

RMOL

0.25.2

Generated by Doxygen 1.7.4

Wed Nov 2 2011 23:53:24

Contents

1	RMOL Documentation	1
1.1	Getting Started	1
1.2	RMOL at SourceForge	1
1.3	RMOL Development	1
1.4	External Libraries	2
1.5	Support RMOL	2
1.6	About RMOL	2
2	People	2
2.1	Project Admins	2
2.2	Developers	2
2.3	Retired Developers	3
2.4	Contributors	3
2.5	Distribution Maintainers	3
3	Coding Rules	3
3.1	Default Naming Rules for Variables	3
3.2	Default Naming Rules for Functions	4
3.3	Default Naming Rules for Classes and Structures	4
3.4	Default Naming Rules for Files	4
3.5	Default Functionality of Classes	4
4	Copyright and License	4
4.1	GNU LESSER GENERAL PUBLIC LICENSE	4
4.1.1	Version 2.1, February 1999	5
4.2	Preamble	5
4.3	TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MOD- IFICATION	6
4.3.1	NO WARRANTY	11
4.3.2	END OF TERMS AND CONDITIONS	12
4.4	How to Apply These Terms to Your New Programs	12
5	Documentation Rules	13
5.1	General Rules	13

5.2	File Header	14
5.3	Grouping Various Parts	14
6	Main features	15
6.1	Optimisation features	15
6.2	Unconstraining	15
6.3	Forecasting features	15
6.4	Overbooking features	15
6.5	Other features	15
7	Make a Difference	15
8	Make a new release	16
8.1	Introduction	16
8.2	Initialisation	16
8.3	Branch creation	16
8.4	Commit and publish the release branch	17
8.5	Update the change-log in the trunk as well	17
8.6	Create distribution packages	17
8.7	Generation the RPM packages	18
8.8	Update distributed change log	18
8.9	Create the binary package, including the documentation	18
8.10	Upload the files to SourceForge	18
8.11	Upload the documentation to SourceForge	18
8.12	Make a new post	19
8.13	Send an email on the announcement mailing-list	19
9	Installation	19
9.1	Table of Contents	19
9.2	Fedora/RedHat Linux distributions	20
9.3	RMOL Requirements	20
9.4	Basic Installation	21
9.5	Compilers and Options	22
9.6	Compiling For Multiple Architectures	22
9.7	Installation Names	23

9.8	Optional Features	24
9.9	Particular systems	24
9.10	Specifying the System Type	25
9.11	Sharing Defaults	25
9.12	Defining Variables	26
9.13	'cmake' Invocation	26
10	Linking with RMOL	30
10.1	Table of Contents	31
10.2	Introduction	31
10.3	Using the pkg-config command	31
10.4	Using the rmol-config script	31
10.5	M4 macro for the GNU Autotools	32
10.6	Using RMOL with dynamic linking	32
11	Test Rules	32
11.1	The Test File	32
11.2	The Reference File	33
11.3	Testing IT++ Library	33
12	Users Guide	33
12.1	Table of Contents	33
12.2	Introduction	34
12.3	Get Started	34
12.3.1	Get the RMOL library	34
12.3.2	Build the RMOL project	34
12.3.3	Build and Run the Tests	34
12.3.4	Install the RMOL Project (Binaries, Documentation)	34
12.4	Exploring the Predefined BOM Tree	34
12.4.1	Forecaster BOM Tree	34
12.4.2	Optimiser BOM Tree	34
12.5	Extending the BOM Tree	35
13	Supported Systems	35
13.1	Table of Contents	35

13.2	Introduction	35
13.3	RMOL 0.23.x	36
13.3.1	Linux Systems	36
13.3.2	Windows Systems	40
13.3.3	Unix Systems	43
14	RMOL Supported Systems (Previous Releases)	44
14.1	RMOL 3.9.1	44
14.2	RMOL 3.9.0	44
14.3	RMOL 3.8.1	44
15	BPSK modulation over an AWGN channel	44
16	Simulation of a convolutional encoder and decoder	45
17	Interleaving and de-interleaving of data	47
18	Writing and reading data from files	48
19	Simulation of LDPC codes the AWGN channel	49
20	Generation of LDPC codes	51
21	Conversion table between IT++ syntax and Matlab/Octave	53
22	MIMO (spatial multiplexing) with convolutional coding	54
23	Using Mixture of Gaussians (MOG) module to model data	59
24	Simulation of QPSK modulation on an AWGN channel	62
25	Generating a correlated Rayleigh fading process	64
26	Simulation of a Reed-Solomon Block Code	65
27	Simulation of a Multicode CDMA system on an AWGN channel	66
28	Using timers to measure execution time	70
29	Tutorials	71
29.1	Table of Contents	71

29.2 Introduction	71
29.2.1 Preparing the StdAir Project for Development	71
29.3 Build a Predefined BOM Tree	72
29.3.1 Instantiate the BOM Root Object	72
29.3.2 Instantiate the (Airline) Inventory Object	72
29.3.3 Link the Inventory Object with the BOM Root	72
29.3.4 Build Another Airline Inventory	73
29.3.5 Dump The BOM Tree Content	73
29.3.6 Result of the Tutorial Program	73
29.4 Extend the Pre-Defined BOM Tree	73
29.4.1 Extend an Airline Inventory Object	73
29.4.2 Build the Specific BOM Objects	74
29.4.3 Result of the Tutorial Program	74
30 A very simple tutorial about vectors and matrixes	75
31 Command-Line Test to Demonstrate How To Test the RMOL Project	76
32 Command-Line Test to Demonstrate How To Test the RMOL Project	80
33 Command-Line Test to Demonstrate How To Test the RMOL Project	81
34 Command-Line Test to Demonstrate How To Test the RMOL Project	86
35 Directory Hierarchy	87
35.1 Directories	87
36 Namespace Index	87
36.1 Namespace List	87
37 Class Index	87
37.1 Class Hierarchy	88
38 Class Index	89
38.1 Class List	89
39 File Index	91
39.1 File List	91

40 Directory Documentation	93
40.1 rmol/basic/ Directory Reference	93
40.2 rmol/batches/ Directory Reference	93
40.3 rmol/bom/ Directory Reference	94
40.4 rmol/command/ Directory Reference	94
40.5 rmol/config/ Directory Reference	94
40.6 doc/ Directory Reference	95
40.7 rmol/factory/ Directory Reference	95
40.8 rmol/bom/old/ Directory Reference	95
40.9 test/rmol/ Directory Reference	95
40.10rmol/ Directory Reference	95
40.11rmol/service/ Directory Reference	96
40.12doc/tutorial/src/ Directory Reference	96
40.13test/ Directory Reference	96
40.14doc/tutorial/ Directory Reference	96
41 Namespace Documentation	97
41.1 RMOL Namespace Reference	97
41.1.1 Typedef Documentation	98
41.1.2 Variable Documentation	99
41.2 stdair Namespace Reference	101
41.2.1 Detailed Description	101
42 Class Documentation	101
42.1 CmdAbstract Class Reference	101
42.2 RMOL::DefaultDCPList Struct Reference	101
42.2.1 Detailed Description	102
42.2.2 Member Function Documentation	102
42.3 RMOL::DefaultMap Struct Reference	102
42.3.1 Detailed Description	102
42.3.2 Member Function Documentation	102
42.4 RMOL::DemandGeneratorList Class Reference	102
42.4.1 Detailed Description	103
42.4.2 Member Typedef Documentation	103
42.4.3 Constructor & Destructor Documentation	103

42.4.4	Member Function Documentation	104
42.5	RMOL::Detruncator Class Reference	104
42.5.1	Detailed Description	104
42.5.2	Member Function Documentation	104
42.6	RMOL::DPOptimiser Class Reference	106
42.6.1	Detailed Description	106
42.6.2	Member Function Documentation	106
42.7	RMOL::EMDetruncator Class Reference	106
42.7.1	Detailed Description	106
42.7.2	Member Function Documentation	107
42.8	RMOL::Emsr Class Reference	107
42.8.1	Detailed Description	107
42.8.2	Member Function Documentation	107
42.9	RMOL::EmsrUtils Class Reference	108
42.9.1	Detailed Description	108
42.9.2	Member Function Documentation	108
42.10	RMOL::FacRmolServiceContext Class Reference	109
42.10.1	Detailed Description	110
42.10.2	Constructor & Destructor Documentation	110
42.10.3	Member Function Documentation	110
42.11	FacServiceAbstract Class Reference	111
42.12	RMOL::Forecaster Class Reference	111
42.12.1	Detailed Description	111
42.12.2	Member Function Documentation	112
42.13	ForecasterTestSuite Class Reference	112
42.13.1	Detailed Description	113
42.13.2	Constructor & Destructor Documentation	113
42.13.3	Member Function Documentation	113
42.13.4	Member Data Documentation	113
42.14	RMOL::ForecastException Class Reference	113
42.14.1	Detailed Description	114
42.14.2	Constructor & Destructor Documentation	114
42.15	RMOL::GuillotineBlockHelper Class Reference	114
42.15.1	Detailed Description	114

42.15.2 Member Function Documentation	114
42.16RMOL::HistoricalBooking Struct Reference	115
42.16.1 Detailed Description	116
42.16.2 Constructor & Destructor Documentation	116
42.16.3 Member Function Documentation	116
42.17RMOL::HistoricalBookingHolder Struct Reference	118
42.17.1 Detailed Description	119
42.17.2 Constructor & Destructor Documentation	119
42.17.3 Member Function Documentation	119
42.18RMOL::InventoryParser Class Reference	122
42.18.1 Detailed Description	123
42.18.2 Member Function Documentation	123
42.19RMOL::MCOptimiser Class Reference	123
42.19.1 Detailed Description	123
42.19.2 Member Function Documentation	124
42.20RMOL::OptimisationException Class Reference	124
42.20.1 Detailed Description	125
42.20.2 Constructor & Destructor Documentation	125
42.21RMOL::Optimiser Class Reference	125
42.21.1 Detailed Description	125
42.21.2 Member Function Documentation	126
42.22OptimiseTestSuite Class Reference	127
42.22.1 Detailed Description	128
42.22.2 Constructor & Destructor Documentation	128
42.22.3 Member Function Documentation	128
42.22.4 Member Data Documentation	128
42.23RMOL::OverbookingException Class Reference	128
42.23.1 Detailed Description	129
42.23.2 Constructor & Destructor Documentation	129
42.24RMOL::RMOL_Service Class Reference	129
42.24.1 Detailed Description	130
42.24.2 Constructor & Destructor Documentation	131
42.24.3 Member Function Documentation	132
42.25RMOL::RMOL_ServiceContext Class Reference	137

42.25.1 Detailed Description	138
42.25.2 Friends And Related Function Documentation	138
42.26RootException Class Reference	138
42.27ServiceAbstract Class Reference	138
42.28StructAbstract Class Reference	139
42.29TestFixture Class Reference	139
42.30UnconstrainerTestSuite Class Reference	139
42.30.1 Detailed Description	140
42.30.2 Constructor & Destructor Documentation	140
42.30.3 Member Function Documentation	140
42.30.4 Member Data Documentation	140
42.31RMOL::UnconstrainingException Class Reference	140
42.31.1 Detailed Description	141
42.31.2 Constructor & Destructor Documentation	141
42.32RMOL::Utilities Class Reference	141
42.32.1 Detailed Description	141
42.32.2 Member Function Documentation	142
43 File Documentation	142
43.1 doc/local/authors.doc File Reference	142
43.2 doc/local/codingrules.doc File Reference	142
43.3 doc/local/copyright.doc File Reference	142
43.4 doc/local/documentation.doc File Reference	143
43.5 doc/local/features.doc File Reference	143
43.6 doc/local/help_wanted.doc File Reference	143
43.7 doc/local/howto_release.doc File Reference	143
43.8 doc/local/index.doc File Reference	143
43.9 doc/local/installation.doc File Reference	143
43.10doc/local/linking.doc File Reference	143
43.11doc/local/test.doc File Reference	143
43.12doc/local/users_guide.doc File Reference	143
43.13doc/local/verification.doc File Reference	143
43.14doc/tutorial/bpsk.doc File Reference	143
43.15doc/tutorial/convcode.doc File Reference	143

43.16doc/tutorial/interleaver.doc File Reference	143
43.17doc/tutorial/itfile.doc File Reference	143
43.18doc/tutorial/ldpc_bersim_awgn.doc File Reference	143
43.19doc/tutorial/ldpc_gen_codes.doc File Reference	143
43.20doc/tutorial/matlab_itpp.doc File Reference	143
43.21doc/tutorial/mimoconv.doc File Reference	143
43.22doc/tutorial/mog.doc File Reference	143
43.23doc/tutorial/qpsk_simulation.doc File Reference	143
43.24doc/tutorial/rayleigh.doc File Reference	143
43.25doc/tutorial/reedsolomon.doc File Reference	143
43.26doc/tutorial/spread.doc File Reference	144
43.27doc/tutorial/src/bpsk.cpp File Reference	144
43.27.1 Function Documentation	144
43.28bpsk.cpp	144
43.29doc/tutorial/src/convcode.cpp File Reference	145
43.29.1 Function Documentation	145
43.30convcode.cpp	145
43.31doc/tutorial/src/interleaver.cpp File Reference	147
43.31.1 Function Documentation	147
43.32interleaver.cpp	147
43.33doc/tutorial/src/ldpc_bersim_awgn.cpp File Reference	148
43.33.1 Function Documentation	148
43.34ldpc_bersim_awgn.cpp	148
43.35doc/tutorial/src/ldpc_gen_codes.cpp File Reference	150
43.35.1 Function Documentation	150
43.36ldpc_gen_codes.cpp	150
43.37doc/tutorial/src/mimoconv.cpp File Reference	152
43.37.1 Function Documentation	152
43.38mimoconv.cpp	152
43.39doc/tutorial/src/mog.cpp File Reference	157
43.39.1 Function Documentation	157
43.40mog.cpp	157
43.41doc/tutorial/src/qpsk_simulation.cpp File Reference	160
43.41.1 Function Documentation	160

43.42qpsk_simulation.cpp	160
43.43doc/tutorial/src/rayleigh.cpp File Reference	162
43.43.1 Function Documentation	162
43.44rayleigh.cpp	162
43.45doc/tutorial/src/read_it_file.cpp File Reference	162
43.45.1 Function Documentation	163
43.46read_it_file.cpp	163
43.47doc/tutorial/src/reedsolomon.cpp File Reference	163
43.47.1 Function Documentation	163
43.48reedsolomon.cpp	164
43.49doc/tutorial/src/spread.cpp File Reference	165
43.49.1 Function Documentation	165
43.50spread.cpp	165
43.51doc/tutorial/src/timer.cpp File Reference	167
43.51.1 Function Documentation	167
43.52timer.cpp	167
43.53doc/tutorial/src/vector_and_matrix.cpp File Reference	168
43.53.1 Function Documentation	168
43.54vector_and_matrix.cpp	168
43.55doc/tutorial/src/write_it_file.cpp File Reference	169
43.55.1 Function Documentation	169
43.56write_it_file.cpp	169
43.57doc/tutorial/timer.doc File Reference	170
43.58doc/tutorial/tutorial.doc File Reference	170
43.59doc/tutorial/vector_and_matrix.doc File Reference	170
43.60rmol/basic/BasConst.cpp File Reference	170
43.61BasConst.cpp	170
43.62rmol/basic/BasConst_Curves.hpp File Reference	172
43.63BasConst_Curves.hpp	172
43.64rmol/basic/BasConst_General.hpp File Reference	172
43.65BasConst_General.hpp	173
43.66rmol/basic/BasConst_RMOL_Service.hpp File Reference	173
43.67BasConst_RMOL_Service.hpp	173
43.68rmol/batches/rmol.cpp File Reference	174

43.68.1 Function Documentation	175
43.68.2 Variable Documentation	175
43.69rmol.cpp	176
43.70rmol/bom/BucketHolderTypes.hpp File Reference	181
43.71BucketHolderTypes.hpp	181
43.72rmol/bom/DistributionParameterList.hpp File Reference	182
43.73DistributionParameterList.hpp	182
43.74rmol/bom/DPOptimiser.cpp File Reference	182
43.75DPOptimiser.cpp	183
43.76rmol/bom/DPOptimiser.hpp File Reference	186
43.77DPOptimiser.hpp	186
43.78rmol/bom/EMDetruncator.cpp File Reference	187
43.79EMDetruncator.cpp	187
43.80rmol/bom/EMDetruncator.hpp File Reference	189
43.81EMDetruncator.hpp	189
43.82rmol/bom/Emsr.cpp File Reference	190
43.83Emsr.cpp	190
43.84rmol/bom/Emsr.hpp File Reference	193
43.85Emsr.hpp	193
43.86rmol/bom/EmsrUtils.cpp File Reference	193
43.87EmsrUtils.cpp	194
43.88rmol/bom/EmsrUtils.hpp File Reference	195
43.89EmsrUtils.hpp	196
43.90rmol/bom/GuillotineBlockHelper.cpp File Reference	196
43.91GuillotineBlockHelper.cpp	197
43.92rmol/bom/GuillotineBlockHelper.hpp File Reference	198
43.93GuillotineBlockHelper.hpp	198
43.94rmol/bom/HistoricalBooking.cpp File Reference	199
43.95HistoricalBooking.cpp	199
43.96rmol/bom/HistoricalBooking.hpp File Reference	200
43.97HistoricalBooking.hpp	201
43.98rmol/bom/HistoricalBookingHolder.cpp File Reference	202
43.99HistoricalBookingHolder.cpp	202
43.100rmol/bom/HistoricalBookingHolder.hpp File Reference	206

43.10HistoricalBookingHolder.hpp	207
43.102mol/bom/MCOptimiser.cpp File Reference	208
43.103MCOptimiser.cpp	209
43.104mol/bom/MCOptimiser.hpp File Reference	214
43.105MCOptimiser.hpp	214
43.106mol/bom/old/DemandGeneratorList.cpp File Reference	215
43.107DemandGeneratorList.cpp	215
43.108mol/bom/old/DemandGeneratorList.hpp File Reference	216
43.109DemandGeneratorList.hpp	216
43.110mol/bom/Utilities.cpp File Reference	217
43.111Utilities.cpp	218
43.112mol/bom/Utilities.hpp File Reference	220
43.113Utilities.hpp	220
43.114mol/command/Detruncator.cpp File Reference	220
43.115Detruncator.cpp	221
43.116mol/command/Detruncator.hpp File Reference	231
43.117Detruncator.hpp	231
43.118mol/command/Forecaster.cpp File Reference	233
43.119Forecaster.cpp	234
43.120mol/command/Forecaster.hpp File Reference	248
43.121Forecaster.hpp	248
43.122mol/command/InventoryParser.cpp File Reference	250
43.123InventoryParser.cpp	250
43.124mol/command/InventoryParser.hpp File Reference	253
43.125InventoryParser.hpp	253
43.126mol/command/Optimiser.cpp File Reference	254
43.127Optimiser.cpp	254
43.128mol/command/Optimiser.hpp File Reference	257
43.129Optimiser.hpp	258
43.130mol/config/rmol-paths.hpp File Reference	258
43.130. Define Documentation	259
43.131rmol-paths.hpp	260
43.132mol/factory/FacRmolServiceContext.cpp File Reference	261
43.133FacRmolServiceContext.cpp	261

43.134	mol/factory/FacRmolServiceContext.hpp File Reference	262
43.135	FacRmolServiceContext.hpp	262
43.136	mol/RMOL_Service.hpp File Reference	263
43.137	RMOL_Service.hpp	264
43.138	mol/RMOL_Types.hpp File Reference	267
43.139	RMOL_Types.hpp	268
43.140	mol/service/RMOL_Service.cpp File Reference	269
43.141	RMOL_Service.cpp	270
43.142	mol/service/RMOL_ServiceContext.cpp File Reference	302
43.143	RMOL_ServiceContext.cpp	302
43.144	mol/service/RMOL_ServiceContext.hpp File Reference	303
43.145	RMOL_ServiceContext.hpp	304
43.146	test/rmol/bomsforforecaster.cpp File Reference	305
43.147	bomsforforecaster.cpp	305
43.148	test/rmol/ForecasterTestSuite.cpp File Reference	310
43.149	ForecasterTestSuite.cpp	310
43.150	test/rmol/ForecasterTestSuite.hpp File Reference	311
43.150.1	Function Documentation	311
43.151	ForecasterTestSuite.hpp	311
43.152	test/rmol/OptimiseTestSuite.cpp File Reference	312
43.153	OptimiseTestSuite.cpp	312
43.154	test/rmol/OptimiseTestSuite.hpp File Reference	316
43.154.1	Function Documentation	316
43.155	OptimiseTestSuite.hpp	316
43.156	test/rmol/UnconstrainerTestSuite.cpp File Reference	317
43.157	UnconstrainerTestSuite.cpp	317
43.158	test/rmol/UnconstrainerTestSuite.hpp File Reference	318
43.158.1	Function Documentation	318
43.159	UnconstrainerTestSuite.hpp	318

1 RMOL Documentation

1.1 Getting Started

- [Main features](#)
- [Installation](#)
- [Linking with RMOL](#)
- [Users Guide](#)
- [Tutorials](#)
- [Copyright and License](#)
- [Make a Difference](#)
- [Make a new release](#)
- [People](#)

1.2 RMOL at SourceForge

- [Project page](#)
- [Download RMOL](#)
- [Open a ticket for a bug or feature](#)
- [Mailing lists](#)
- [Forums](#)
 - [Discuss about Development issues](#)
 - [Ask for Help](#)
 - [Discuss RMOL](#)

1.3 RMOL Development

- [Git Repository](#) (Subversion is deprecated)
- [Coding Rules](#)
- [Documentation Rules](#)
- [Test Rules](#)

1.4 External Libraries

- [Boost](#) (C++ STL extensions)
- [Python](#)
- [MySQL client](#)
- [SOI](#) (C++ DB API)

1.5 Support RMOL

1.6 About RMOL

RMOL is a C++ library of revenue management and optimisation classes and functions. **RMOL** mainly targets simulation purposes. [N](#)

RMOL makes an extensive use of existing open-source libraries for increased functionality, speed and accuracy. In particular [GSL](#) (*GNU Scientific Library*) and [Boost](#) (*C++ Standard Extensions*) libraries are used.

The **RMOL** library originates from the department of Operational Research and Innovation at [Amadeus](#), Sophia Antipolis, France. **RMOL** is released under the terms of the [GNU Lesser General Public License](#) (LGPLv2.1) for you to enjoy.

RMOL should work on [GNU/Linux](#), [Sun Solaris](#), Microsoft Windows (with [Cygwin](#), [MinGW/MSYS](#), or [Microsoft Visual C++ .NET](#)) and [Mac OS X](#) operating systems.

Note

(N) - The **RMOL** library is **NOT** intended, in any way, to be used by airlines for production systems. If you want to report issue, bug or feature request, or if you just want to give feedback, have a look on the right-hand side of this page for the preferred reporting methods. In any case, please do not contact Amadeus directly for any matter related to [RMOL](#).

2 People

2.1 Project Admins

- Denis Arnaud <denis_arnaud@users.sourceforge.net> ([N](#))
- Anh Quan Nguyen <quannaus@users.sourceforge.net> ([N](#))

2.2 Developers

- Anh Quan Nguyen <quannaus@users.sourceforge.net> ([N](#))
- Denis Arnaud <denis_arnaud@users.sourceforge.net> ([N](#))
- Nicolas Bondoux <nbondoux@users.sourceforge.net> ([N](#))

2.3 Retired Developers

- Patrick Grandjean <pgrandjean@users.sourceforge.net> ([N](#))
- Benoit Lardeux <benlardeux@users.sourceforge.net> ([N](#))

- Karim Duval <duvalkarim@users.sourceforge.net> (N)
- Ngoc-Thach Hoang <hoangngocthach@users.sourceforge.net> (N)
- Son Nguyen Kim <snguyenkim@users.sourceforge.net> (N)

2.4 Contributors

- Emmanuel Bastien <ebastien@users.sourceforge.net> (N)
- Christophe Lacombe <ddtoif@users.sourceforge.net> (N)

2.5 Distribution Maintainers

- **Fedora/RedHat**: Denis Arnaud <denis_arnaud@users.sourceforge.net> (N)
- **Debian**: Emmanuel Bastien <ebastien@users.sourceforge.net> (N)

Note

(N) - **Amadeus** employees.

3 Coding Rules

In the following sections we describe the naming conventions which are used for files, classes, structures, local variables, and global variables.

3.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

3.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

- `int myFunctionName (const int& a, int b)`

3.3 Default Naming Rules for Classes and Structures

Each new word in a class or structure name should always start with a capital letter and the words should be separated with an under-score. Abbreviations are written with capital letters. Examples:

- `MyClassName`
- `MyStructName`

3.4 Default Naming Rules for Files

Files are named after the C++ class names.

Source files are named using `.cpp` suffix, whereas header files end with `.hpp` extension. Examples:

- `FlightDate.hpp`
- `SegmentDate.cpp`

3.5 Default Functionality of Classes

All classes that are configured by input parameters should include:

- default empty constructor
- one or more additional constructor(s) that takes input parameters and initializes the class instance
- setup function, preferably named `'setup'` or `'set_parameters'`

Explicit destructor functions are not required, unless they are needed. It shall not be possible to use any of the other member functions unless the class has been properly initiated with the input parameters.

4 Copyright and License

4.1 GNU LESSER GENERAL PUBLIC LICENSE

4.1.1 Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

4.2 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only

if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

4.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control

compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium

does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library,

and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- a) Accompany the combined library with a copy of the same work based on the Library,

uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by

patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

4.3.1 NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4.3.2 END OF TERMS AND CONDITIONS

4.4 How to Apply These Terms to Your New Programs

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.
```

```
This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.
```

```
You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library 'Frob' (a library for tweaking knobs) written by James Random Hacker.
```

```
<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

Source

5 Documentation Rules

5.1 General Rules

All classes in [RMOL](#) should be properly documented with Doxygen comments in include (.hpp) files. Source (.cpp) files should be documented according to a normal standard for well documented C++ code.

An example of how the interface of a class shall be documented in [RMOL](#) is shown here:

```

/*!
 * \brief Brief description of MyClass here
 *
 * Detailed description of MyClass here. With example code if needed.
 */
class MyClass {
public:
    /*! Default constructor
     * MyClass(void) { setup_done = false; }
     */

    /*!
     * \brief Constructor that initializes the class with parameters
     *
     * Detailed description of the constructor here if needed
     *
     * \param[in] param1 Description of \a param1 here
     * \param[in] param2 Description of \a param2 here
     */
    MyClass(TYPE1 param1, TYPE2 param2) { setup(param1, param2); }

    /*!
     * \brief Setup function for MyClass
     *
     * Detailed description of the setup function here if needed
     *
     * \param[in] param1 Description of \a param1 here
     * \param[in] param2 Description of \a param2 here
     */
    void setup(TYPE1 param1, TYPE2 param2);

    /*!
     * \brief Brief description of memberFunction1
     *
     * Detailed description of memberFunction1 here if needed
     *
     * \param[in] param1 Description of \a param1 here
     * \param[in] param2 Description of \a param2 here
     * \param[in,out] param3 Description of \a param3 here
     * \return Description of the return value here
     */
    TYPE4 memberFunction1(TYPE1 param1, TYPE2 param2, TYPE3 &param3);

private:
    bool _setupDone;          /*!< Variable that checks if the class is properly
                               initialized with parameters */
    TYPE1 _privateVariable1; /*!< Short description of _privateVariable1 here
    TYPE2 _privateVariable2; /*!< Short description of _privateVariable2 here
};

```

5.2 File Header

All files should start with the following header, which include Doxygen's `\file`, `\brief` and `\author` tags, `$Date$` and `$Revisions$` CVS tags, and a common copyright note:

```

/*!
 * \file
 * \brief Brief description of the file here
 * \author Names of the authors who contributed to this code
 * \date Date
 *
 * Detailed description of the file here if needed.
 *
 * -----
 *
 * RMOL - C++ Revenue Management Object Library
 *
 * Copyright (C) 2007-2010 (\see authors file for a list of contributors)
 *
 * \see copyright file for license information
 *
 * -----
 */

```

5.3 Grouping Various Parts

All functions must be added to a Doxygen group in order to appear in the documentation. The following code example defines the group 'my_group':

```

/*!
 * \defgroup my_group Brief description of the group here
 *
 * Detailed description of the group here
 */

```

The following example shows how to document the function `myFunction` and how to add it to the group `my_group`:

```

/*!
 * \brief Brief description of myFunction here
 * \ingroup my_group
 *
 * Detailed description of myFunction here
 *
 * \param[in] param1 Description of \a param1 here
 * \param[in] param2 Description of \a param2 here
 * \return Description of the return value here
 */
TYPE3 myFunction(TYPE1 param1, TYPE2 &param2);

```

6 Main features

A short list of the main features of [RMOL](#) is given below sorted in different categories. Many more features and functions exist and for these we refer to the reference documentation.

6.1 Optimisation features

- [Dynamic Programming \(DP\)](#)
- [EMSRa](#) and [EMSRb](#)
- Network optimisation with [Linear Programming \(LP\)](#)

6.2 Unconstraining

- Inventory censorflag and guillotine
- E-M (Expectation Maximisation)

6.3 Forecasting features

- [Exponential Smoothing](#)
- [Moving Average](#)

6.4 Overbooking features

- Cancellations and No-Shows
- Cost-based optimisation
- Service-based optimisation

6.5 Other features

- CSV input file parsing

7 Make a Difference

Do not ask what [RMOL](#) can do for you. Ask what you can do for [RMOL](#).

You can help us to develop the [RMOL](#) library. There are always a lot of things you can do:

- Start using [RMOL](#)
- Tell your friends about [RMOL](#) and help them to get started using it
- If you find a bug, report it to us. Without your help we can never hope to produce a bug free code.
- Help us to improve the documentation by providing information about documentation bugs

- Answer support requests in the [RMOL](#) discussion forums on SourceForge. If you know the answer to a question, help others to overcome their [RMOL](#) problems.
- Help us to improve our algorithms. If you know of a better way (e.g. that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help us to port [RMOL](#) to new platforms. If you manage to compile [RMOL](#) on a new platform, then tell us how you did it.
- Send us your code. If you have a good [RMOL](#) compatible code, which you can release under the LGPL, and you think it should be included in [RMOL](#), then send it to us.
- Become an [RMOL](#) developer. Send us an e-mail and tell what you can do for [RMOL](#).

8 Make a new release

8.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of [RMOL](#) using a Linux development machine and the SourceForge project site.

The following steps are required to make a release of the distribution package.

8.2 Initialisation

Clone locally the full [Git](#) project:

```
cd ~
mkdir -p dev/sim
cd ~/dev/sim
git clone git://rmol.git.sourceforge.net/gitroot/rmol/rmol rmolgit
cd rmolgit
git checkout trunk
```

8.3 Branch creation

Create the branch, on your local clone, corresponding to the new release (say, 0.50.0):

```
cd ~/dev/sim/rmolgit
git checkout trunk
git checkout -b 0.50.0
```

Update the version in the various build system files, replacing 99.99.99 by the correct version number:

```
vi CMakeLists.txt
vi autogen.sh
```

Update the version and add a change-log in the ChangeLog and in the RPM specification files:

```
vi ChangeLog
vi rmol.spec
```

8.4 Commit and publish the release branch

Commit the new release:

```
cd ~/dev/sim/rmolgit
git add -A
git commit -m "[Release 0.50.0] Release of version 0.50.0."
git push
```

8.5 Update the change-log in the trunk as well

Update the change-log in the ChangeLog and RPM specification files:

```
cd ~/dev/sim/rmolgit
git checkout trunk
vi ChangeLog
vi rmol.spec
```

Commit the change-logs and publish the trunk (main development branch):

```
git commit -m "[Doc] Integrated the change-log of the release 0.50.0."
git push
```

8.6 Create distribution packages

Create the distribution packages using the following command:

```
cd ~/dev/sim/rmolgit
git checkout 0.50.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/rmol-0.50.0 \
      -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
make check && make dist
```

This will configure, compile and check the package. The output packages will be named, for instance, `rmol-0.50.0.tar.gz` and `rmol-0.50.0.tar.bz2`.

8.7 Generation the RPM packages

Optionally, generate the RPM package (for instance, for [Fedora/RedHat](#)):

```
cd ~/dev/sim/rmolgit
git checkout 0.50.0
```



```
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/rmol-0.50.0 \
      -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
make dist
```

To perform this step, rpm-build, rpmlint and rpmdevtools have to be available on the system.

```
cp rmol.spec ~/dev/packages/SPECS \
  && cp rmol-0.50.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba rmol.spec
rpmlint -i ../SPECS/rmol.spec ../SRPMS/rmol-0.50.0-1.fc15.src.rpm \
  ../RPMS/noarch/rmol-* ../RPMS/i686/rmol-*
```

8.8 Update distributed change log

Update the NEWS and ChangeLog files with appropriate information, including what has changed since the previous release. Then commit and push the changes into the [RMOL's Git repository](#).

8.9 Create the binary package, including the documentation

Create the binary package, which includes HTML and PDF documentation, using the following command:

```
make package
```

The output binary package will be named, for instance, `rmol-0.50.0-Linux.tar.bz2`. That package contains both the HTML and PDF documentation. The binary package contains also the executables and shared libraries, as well as C++ header files, but all of those do not interest us for now.

8.10 Upload the files to SourceForge

Upload the distribution and documentation packages to the SourceForge server. Check [SourceForge help page on uploading software](#).

8.11 Upload the documentation to SourceForge

In order to update the Web site files, either:

- [synchronise them with rsync and SSH](#):

```
cd ~/dev/sim/rmolgit
git checkout 0.50.0
rsync -aiv doc/html/ doc/latex/refman.pdf joe,rmol@web.sourceforge.net:htdocs/
```

where `-aiv` options mean:

- `-a`: archive/mirror mode; equals `-rlptgoD` (no `-H`, `-A`, `-X`)
 - `-v`: increase verbosity
 - `-i`: output a change-summary for all updates
 - Note the trailing slashes (/) at the end of both the source and target directories. It means that the content of the source directory (`doc/html`), rather than the directory itself, has to be copied into the content of the target directory.
- or use the [SourceForge Shell service](#).

8.12 Make a new post

- submit a new entry in the [SourceForge project-related news feed](#)
- make a new post on the [SourceForge hosted WordPress blog](#)
- and update, if necessary, [Trac tickets](#).

8.13 Send an email on the announcement mailing-list

Finally, you should send an announcement to rmol-announce@lists.sourceforge.net (see <https://lists.sourceforge.net/lists/listinfo/rmol-announce> for the archives)

9 Installation

9.1 Table of Contents

- [Fedora/RedHat Linux distributions](#)
- [RMOL Requirements](#)
- [Basic Installation](#)
- [Compilers and Options](#)
- [Compiling For Multiple Architectures](#)
- [Installation Names](#)
- [Optional Features](#)
- [Particular systems](#)
- [Specifying the System Type](#)
- [Sharing Defaults](#)
- [Defining Variables](#)
- [‘cmake’ Invocation](#)

9.2 Fedora/RedHat Linux distributions

Note that on [Fedora/RedHat](#) Linux distributions, RPM packages are available and can be installed with your usual package manager. For instance:

```
yum -y install rmol-devel rmol-doc
```

RPM packages can also be available on the [SourceForge download site](#).

9.3 RMOL Requirements

[RMOL](#) should compile without errors or warnings on most GNU/Linux systems, on UNIX systems like Solaris SunOS, and on POSIX based environments for Microsoft Windows like Cygwin or MinGW with MSYS. It can be also built on Microsoft Windows NT/2000/XP/Vista/7 using Microsoft's Visual C++ .NET, but our support for this compiler is limited. For GNU/Linux, SunOS, Cygwin and MinGW we assume that you have at least the following GNU software installed on your computer:

- GNU Autotools:
 - [autoconf](#),
 - [automake](#),
 - [libtool](#),
 - [make](#), version 3.72.1 or later (check version with ``make --version``)
- [GCC](#) - GNU C++ Compiler (g++), version 4.3.x or later (check version with ``gcc --version``)
- [Boost](#) - C++ STL extensions, version 1.35 or later (check version with ``grep "define BOOST_LIB_VERSION" /usr/include/boost/version.hpp``)
- [MySQL](#) - Database client libraries, version 5.0 or later (check version with ``mysql --version``)
- [SOI](#) - C++ database client library wrapper, version 3.0.0 or later (check version with ``soci-config --version``)

Optionally, you might need a few additional programs: [Doxygen](#), [LaTeX](#), [Dvips](#) and [Ghostscript](#), to generate the HTML and PDF documentation.

We strongly recommend that you use recent stable releases of the GCC, if possible. We do not actively work on supporting older versions of the GCC, and they may therefore (without prior notice) become unsupported in future releases of [RMOL](#).

9.4 Basic Installation

Briefly, the shell commands ``./cmake .. && make install`` should configure, build and install this package. The following more-detailed instructions are generic; see the ``README`` file for instructions specific to this package. Some packages provide this ``INSTALL`` file but do not implement all of the features documented below.

The lack of an optional feature in a given package is not necessarily a bug. More recommendations for GNU packages can be found in the info page corresponding to "Makefile Conventions: (standards)Makefile Conventions".

The `'cmake'` shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a `'Makefile'` in each directory of the package. It may also create one or more `'h'` files containing system-dependent definitions. Finally, it creates a `'CMakeCache.txt'` cache file that you can refer to in the future to recreate the current configuration, and files `'CMakeFiles'` containing compiler output (useful mainly for debugging `'cmake'`).

It can also use an optional file (typically called `'config.cache'` and enabled with `'--cache-file=config.cache'` or simply `'-C'`) that saves the results of its tests to speed up reconfiguring. Caching is disabled by default to prevent problems with accidental use of stale cache files.

If you need to do unusual things to compile the package, please try to figure out how `'configure'` could check whether to do them, and mail diffs or instructions to the address given in the `'README'` so they can be considered for the next release. If you are using the cache, and at some point `'config.cache'` contains results you don't want to keep, you may remove or edit it.

The file `'CMakeLists.txt'` is used to create the `'Makefile'` files.

The simplest way to compile this package is:

1. `'cd'` to the directory containing the package's source code and type `'./cmake . '` to configure the package for your system. Running `'cmake'` is generally fast. While running, it prints some messages telling which features it is checking for.
2. Type `'make'` to compile the package.
3. Optionally, type `'make check'` to run any self-tests that come with the package, generally using the just-built uninstalled binaries.
4. Type `'make install'` to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the `'make install'` phase executed with root privileges.
5. You can remove the program binaries and object files from the source code directory by typing `'make clean'`. To also remove the files that `'configure'` created (so you can compile the package for a different kind of computer), type `'make distclean'`. There is also a `'make maintainer-clean'` target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.
6. Often, you can also type `'make uninstall'` to remove the installed files again. In practice, not all packages have

tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.

9.5 Compilers and Options

Some systems require unusual options for compilation or linking that the 'cmake' script does not know about. Run './cmake --help' for details on some of the pertinent environment variables.

You can give 'cmake' initial values for configuration parameters by setting variables in the command line or in the environment. Here is an example:

```
./cmake CC=c99 CFLAGS=-g LIBS=-lposix
```

See also

[Defining Variables](#) for more details.

9.6 Compiling For Multiple Architectures

You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you can use GNU 'make'. 'cd' to the directory where you want the object files and executables to go and run the 'configure' script. 'configure' automatically checks for the source code in the directory that 'configure' is in and in '..'. This is known as a "VPATH" build.

With a non-GNU 'make', it is safer to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use 'make distclean' before reconfiguring for another architecture.

On MacOS X 10.5 and later systems, you can create libraries and executables that work on multiple system types--known as "fat" or "universal" binaries--by specifying multiple '-arch' options to the compiler but only a single '-arch' option to the preprocessor. Like this:

```
./configure CC="gcc -arch i386 -arch x86_64 -arch ppc -arch ppc64" \  
           CXX="g++ -arch i386 -arch x86_64 -arch ppc -arch ppc64" \  
           CPP="gcc -E" CXXCPP="g++ -E"
```

This is not guaranteed to produce working output in all cases, you may have to build one architecture at a time and combine the results using the 'lipo' tool if you have problems.

9.7 Installation Names

By default, `'make install'` installs the package's commands under `'/usr/local/bin'`, include files under `'/usr/local/include'`, etc. You can specify an installation prefix other than `'/usr/local'` by giving `'configure'` the option `'--prefix=PREFIX'`, where `PREFIX` must be an absolute file name.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you pass the option `'--exec-prefix=PREFIX'` to `'configure'`, the package uses `PREFIX` as the prefix for installing programs and libraries. Documentation and other data files still use the regular prefix.

In addition, if you use an unusual directory layout you can give options like `'--bindir=DIR'` to specify different values for particular kinds of files. Run `'configure --help'` for a list of the directories you can set and what kinds of files go in them. In general, the default for these options is expressed in terms of `'${prefix}'`, so that specifying just `'--prefix'` will affect all of the other directory specifications that were not explicitly provided.

The most portable way to affect installation locations is to pass the correct locations to `'configure'`; however, many packages provide one or both of the following shortcuts of passing variable assignments to the `'make install'` command line to change installation locations without having to reconfigure or recompile.

The first method involves providing an override variable for each affected directory. For example, `'make install prefix=/alternate/directory'` will choose an alternate location for all directory configuration variables that were expressed in terms of `'${prefix}'`. Any directories that were specified during `'configure'`, but not in terms of `'${prefix}'`, must each be overridden at install time for the entire installation to be relocated. The approach of makefile variable overrides for each directory variable is required by the GNU Coding Standards, and ideally causes no recompilation. However, some platforms have known limitations with the semantics of shared libraries that end up requiring recompilation when using this method, particularly noticeable in packages that use GNU Libtool.

The second method involves providing the `'DESTDIR'` variable. For example, `'make install DESTDIR=/alternate/directory'` will prepend `'/alternate/directory'` before all installation names. The approach of `'DESTDIR'` overrides is not required by the GNU Coding Standards, and does not work on platforms that have drive letters. On the other hand, it does better at avoiding recompilation issues, and works well even when some directory options were not specified in terms of `'${prefix}'` at `'configure'` time.

9.8 Optional Features

If the package supports it, you can cause programs to be installed with an extra prefix or suffix on their names by giving `'cmake'` the option `'--program-prefix=PREFIX'` or `'--program-suffix=SUFFIX'`.

Some packages pay attention to `'--enable-FEATURE'` options to `'configure'`, where `FEATURE` indicates an optional part of the package. They may also pay attention to `'--with-PACKAGE'` options, where `PACKAGE` is something like `'gnu-as'` or `'x'` (for the X Window System). The `'README'` should mention any `'--enable-'` and `'--with-'` options that the package recognizes.

For packages that use the X Window System, `'configure'` can usually find the X include and library files automatically, but if it doesn't, you can use the `'configure'` options `'--x-includes=DIR'` and `'--x-libraries=DIR'` to specify their locations.

Some packages offer the ability to configure how verbose the execution of `'make'` will be. For these packages, running `./configure --enable-silent-rules` sets the default to minimal output, which can be overridden with `'make V=1'`; while running `./configure --disable-silent-rules` sets the default to verbose, which can be overridden with `'make V=0'`.

9.9 Particular systems

On HP-UX, the default C compiler is not ANSI C compatible. If GNU CC is not installed, it is recommended to use the following options in order to use an ANSI C compiler:

```
./configure CC="cc -Ae -D_XOPEN_SOURCE=500"
```

and if that doesn't work, install pre-built binaries of GCC for HP-UX.

On OSF/1 a.k.a. Tru64, some versions of the default C compiler cannot parse its `'<wchar.h>'` header file. The option `'-nodtk'` can be used as a workaround. If GNU CC is not installed, it is therefore recommended to try

```
./configure CC="cc"
```

and if that doesn't work, try

```
./configure CC="cc -nodtk"
```

On Solaris, don't put `'/usr/ucb'` early in your `'PATH'`. This directory contains several dysfunctional programs; working

variants of these programs are available in `'/usr/bin'`. So, if you need `'/usr/ucb'` in your `'PATH'`, put it `_after_` `'/usr/bin'`.

On Haiku, software installed for all users goes in `'/boot/common'`, not `'/usr/local'`. It is recommended to use the following options:

```
./cmake -DCMAKE_INSTALL_PREFIX=/boot/common
```

9.10 Specifying the System Type

There may be some features `'configure'` cannot figure out automatically, but needs to determine by the type of machine the package will run on. Usually, assuming the package is built to be run on the `_same_` architectures, `'configure'` can figure that out, but if it prints a message saying it cannot guess the machine type, give it the `'--build=TYPE'` option. TYPE can either be a short name for the system type, such as `'sun4'`, or a canonical name which has the form CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

- OS
- KERNEL-OS

See the file `'config.sub'` for the possible values of each field. If `'config.sub'` isn't included in this package, then this package doesn't need to know the machine type.

If you are `_building_` compiler tools for cross-compiling, you should use the option `'--target=TYPE'` to select the type of system they will produce code for.

If you want to `_use_` a cross compiler, that generates code for a platform different from the build platform, you should specify the `"host"` platform (i.e., that on which the generated programs will eventually be run) with `'--host=TYPE'`.

9.11 Sharing Defaults

If you want to set default values for `'configure'` scripts to share, you can create a site shell script called `'config.site'` that gives default values for variables like `'CC'`, `'cache_file'`, and `'prefix'`. `'configure'` looks for `'PREFIX/share/config.site'` if it exists, then `'PREFIX/etc/config.site'` if it exists. Or, you can set the `'CONFIG_SITE'` environment variable to the location of the site script. A warning: not all `'configure'` scripts look for a site script.

9.12 Defining Variables

Variables not defined in a site shell script can be set in the environment passed to 'configure'. However, some packages may run configure again during the build, and the customized values of these variables may be lost. In order to avoid this problem, you should set them in the 'configure' command line, using 'VAR=value'. For example:

```
./configure CC=/usr/local2/bin/gcc
```

causes the specified 'gcc' to be used as the C compiler (unless it is overridden in the site shell script).

Unfortunately, this technique does not work for 'CONFIG_SHELL' due to an Autoconf bug. Until the bug is fixed you can use this workaround:

```
CONFIG_SHELL=/bin/bash /bin/bash ./configure CONFIG_SHELL=/bin/bash
```

9.13 'cmake' Invocation

'cmake' recognizes the following options to control how it operates.

- '--help', '-h' print a summary of all of the options to 'configure', and exit.
- '--help=short', '--help=recursive' print a summary of the options unique to this package's 'configure', and exit. The 'short' variant lists options used only in the top level, while the 'recursive' variant lists options also present in any nested packages.
- '--version', '-V' print the version of Autoconf used to generate the 'configure' script, and exit.
- '--cache-file=FILE' enable the cache: use and save the results of the tests in FILE, traditionally 'config.cache'. FILE defaults to '/dev/null' to disable caching.
- '--config-cache', '-C' alias for '--cache-file=config.cache'.
- '--quiet', '--silent', '-q' do not print messages saying which checks are being made. To suppress all normal output, redirect it to '/dev/null' (any error messages will still be shown).
- '--srcdir=DIR' look for the package's source code in directory DIR. Usually 'configure' can determine that directory automatically.
- '--prefix=DIR' use DIR as the installation prefix.

See also

[Installation Names](#) for more details, including other options available for fine-tuning the installation locations.

- '--no-create', '-n' run the configure checks, but stop before creating any output files.

'cmake' also accepts some other, not widely useful, options. Run 'cmake --help' for more details.

The 'cmake' script produces an output like this:

```
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/rmol-99.99.99 -DLIB_SUFFIX=64 -DCMAKE_BUILD
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: 56c6c98cf2cfb4008a0acd35d08075cf5f79e693 trunk
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   program_options
--   date_time
--   iostreams
--   serialization
--   filesystem
--   unit_test_framework
--   python
-- Found Boost version: 1.46.0
-- Found BoostWrapper: /usr/include (Required is at least version "1.41")
-- Requires MySQL without specifying any version
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so
-- Found MySQL version: 5.5.14
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI: /usr/lib64/libsoci_core.so (Required is at least version "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (Required is at least version "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires StdAir-0.35
-- Found StdAir version: 0.37.1
-- Requires Doxygen without specifying any version
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'airraclib' to CXX
-- Had to set the linker language for 'rmollib' to CXX
-- Test 'UnconstrainerTest' to be built with 'UnconstrainerTestSuite.cpp'
-- Test 'ForecasterTest' to be built with 'ForecasterTestSuite.cpp'
-- Test 'OptimiseTest' to be built with 'OptimiseTestSuite.cpp'
-- Test 'BOMsForForecasterTest' to be built with 'bomsforforecaster.cpp'
```

```

--
-- =====
-- -----
-- ---      Project Information      ---
-- -----
-- PROJECT_NAME ..... : rmol
-- PACKAGE_PRETTY_NAME ..... : RMOL
-- PACKAGE ..... : rmol
-- PACKAGE_NAME ..... : RMOL
-- PACKAGE_BRIEF ..... : C++ library of Revenue Management and Optimisation classes and
-- PACKAGE_VERSION ..... : 99.99.99
-- GENERIC_LIB_VERSION ..... : 99.99.99
-- GENERIC_LIB_SOVERSION ..... : 99.99
--
-- -----
-- ---      Build Configuration      ---
-- -----
-- Modules to build ..... : airrac;rmol
-- Libraries to build/install ..... : airraclib;rmollib
-- Binaries to build/install ..... : airrac;rmol
-- Modules to test ..... : rmol
-- Binaries to test ..... : UnconstrainerTesttst;UnconstrainerTesttst;ForecasterTesttst;
--
-- * Module ..... : airrac
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : airraclib
--   + Executables to build/install : airrac
--   + Tests to perform ..... :
-- * Module ..... : rmol
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers : airraclib
--   + Libraries to build/install . : rmollib
--   + Executables to build/install : rmol
--   + Tests to perform ..... : UnconstrainerTesttst;UnconstrainerTesttst;ForecasterTesttst;
--
-- BUILD_SHARED_LIBS ..... : ON
-- CMAKE_BUILD_TYPE ..... : Debug
-- * CMAKE_C_FLAGS ..... :
-- * CMAKE_CXX_FLAGS ..... : -Wall -Werror
-- * BUILD_FLAGS ..... :
-- * COMPILE_FLAGS ..... :
-- CMAKE_MODULE_PATH ..... : /home/user/dev/sim/rmol/rmolgithub/config/
-- CMAKE_INSTALL_PREFIX ..... : /home/user/dev/deliveries/rmol-99.99.99
--
-- * Doxygen:
--   - DOXYGEN_VERSION ..... : 1.7.4
--   - DOXYGEN_EXECUTABLE ..... : /usr/bin/doxygen
--   - DOXYGEN_DOT_EXECUTABLE ..... : /usr/bin/dot
--   - DOXYGEN_DOT_PATH ..... : /usr/bin
--
-- -----
-- ---      Installation Configuration      ---
-- -----
-- INSTALL_LIB_DIR ..... : /home/user/dev/deliveries/rmol-99.99.99/lib64
-- INSTALL_BIN_DIR ..... : /home/user/dev/deliveries/rmol-99.99.99/bin
-- INSTALL_INCLUDE_DIR ..... : /home/user/dev/deliveries/rmol-99.99.99/include
-- INSTALL_DATA_DIR ..... : /home/user/dev/deliveries/rmol-99.99.99/share
-- INSTALL_SAMPLE_DIR ..... : /home/user/dev/deliveries/rmol-99.99.99/share/rmol/samples
-- INSTALL_DOC ..... : ON
--
-- -----

```

```

-- --- Packaging Configuration ---
-- -----
-- CPACK_PACKAGE_CONTACT ..... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot r
-- CPACK_PACKAGE_VENDOR ..... : Denis Arnaud
-- CPACK_PACKAGE_VERSION ..... : 99.99.99
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/rmol/rmolgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/rmol/rmolgithub/COPYING
-- CPACK_GENERATOR ..... : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ..... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : rmol-99.99.99
--
-- -----
-- --- External libraries ---
-- -----
--
-- * Boost:
--   - Boost_VERSION ..... : 104600
--   - Boost_LIB_VERSION ..... : 1_46
--   - Boost_HUMAN_VERSION ..... : 1.46.0
--   - Boost_INCLUDE_DIRS ..... : /usr/include
--   - Boost required components .. : program_options;date_time;iostreams;serialization;filesystem
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_iostreams-mt.so;debug;/usr/lib
--
-- * MySQL:
--   - MYSQL_VERSION ..... : 5.5.14
--   - MYSQL_INCLUDE_DIR ..... : /usr/include/mysql
--   - MYSQL_LIBRARIES ..... : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ..... : 3.0.0
--   - SOCI_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_MYSQL_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_LIBRARIES ..... : /usr/lib64/libsoci_core.so
--   - SOCI_MYSQL_LIBRARIES ..... : /usr/lib64/libsoci_mysql.so
--
-- * StdAir:
--   - STDAIR_VERSION ..... : 0.37.1
--   - STDAIR_BINARY_DIRS ..... : /home/user/dev/deliveries/stdair-0.37.1/bin
--   - STDAIR_EXECUTABLES ..... : stdair
--   - STDAIR_LIBRARY_DIRS ..... : /home/user/dev/deliveries/stdair-0.37.1/lib64
--   - STDAIR_LIBRARIES ..... : stdairlib;stdairuiclib
--   - STDAIR_INCLUDE_DIRS ..... : /home/user/dev/deliveries/stdair-0.37.1/include
--   - STDAIR_SAMPLE_DIR ..... : /home/user/dev/deliveries/stdair-0.37.1/share/stdair/samples
--
-- Change a value with: cmake -D<Variable>=<Value>
-- =====
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/rmol/rmolgithub/build

```

It is recommended that you check if your library has been compiled and linked properly and works as expected. To do so, you should execute the testing process 'make check'. As a result, you should obtain a similar report:

```

[ 0%] Built target hdr_cfg_rmol
[ 0%] Built target hdr_cfg_airrac
[ 30%] Built target airraclib
[ 86%] Built target rmolllib

```

```
[ 90%] Built target BOMsForForecasterTesttst
[ 93%] Built target UnconstrainerTesttst
[ 96%] Built target ForecasterTesttst
[100%] Built target OptimiseTesttst
Scanning dependencies of target check_rmoltst
Test project /home/user/dev/sim/rmol/rmolgithub/build/test/rmol
  Start 1: UnconstrainerTesttst
1/4 Test #1: UnconstrainerTesttst ..... Passed    0.04 sec
  Start 2: ForecasterTesttst
2/4 Test #2: ForecasterTesttst ..... Passed    0.04 sec
  Start 3: OptimiseTesttst
3/4 Test #3: OptimiseTesttst ..... Passed    0.44 sec
  Start 4: BOMsForForecasterTesttst
4/4 Test #4: BOMsForForecasterTesttst ..... Passed    0.02 sec

100% tests passed, 0 tests failed out of 4

Total Test time (real) = 0.78 sec
[100%] Built target check_rmoltst
Scanning dependencies of target check
[100%] Built target check
```

Check if all the executed tests PASSED. If not, please contact us by filling a [bug-report](#).

Finally, you should install the compiled and linked library, include files and (optionally) HTML and PDF documentation by typing:

```
make install
```

Depending on the PREFIX settings during configuration, you might need the root (administrator) access to perform this step.

Eventually, you might invoke the following command

```
make clean
```

to remove all files created during compilation process, or even

```
cd ~/dev/sim/rmolgit
rm -rf build && mkdir build
cd build
```

to remove everything.

10 Linking with RMOL

10.1 Table of Contents

- [Introduction](#)

- [Using the pkg-config command](#)
- [Using the rmol-config script](#)
- [M4 macro for the GNU Autotools](#)
- [Using RMOL with dynamic linking](#)

10.2 Introduction

There are two convenient methods of linking your programs with the [RMOL](#) library. The first one employs the `'pkg-config'` command (see <http://pkgconfig.freedesktop.org/>), whereas the second one uses `'rmol-config'` script. These methods are shortly described below.

10.3 Using the pkg-config command

`'pkg-config'` is a helper tool used when compiling applications and libraries. It helps you insert the correct compiler and linker options. The syntax of the `'pkg-config'` is as follows:

```
pkg-config <options> <library_name>
```

For instance, assuming that you need to compile an [RMOL](#) based program `'my_prog.cpp'`, you should use the following command:

```
g++ `pkg-config --cflags rmol` -o my_prog my_prog.cpp `pkg-config --libs rmol`
```

For more information see the `'pkg-config'` man pages.

10.4 Using the rmol-config script

[RMOL](#) provides a shell script called `'rmol-config'`, which is installed by default in `'$prefix/bin'` (`'/usr/local/bin'`) directory. It can be used to simplify compilation and linking of [RMOL](#) based programs. The usage of this script is quite similar to the usage of the `'pkg-config'` command.

Assuming that you need to compile the program `'my_prog.cpp'` you can now do that with the following command:

```
g++ `rmol-config --cflags` -o my_prog_opt my_prog.cpp `rmol-config --libs`
```

A list of `'rmol-config'` options can be obtained by typing:

```
rmol-config --help
```

If the `'rmol-config'` command is not found by your shell, you should add its location `'$prefix/bin'` to the `PATH` environment variable, e.g.:

```
export PATH=/usr/local/bin:$PATH
```

10.5 M4 macro for the GNU Autotools

A M4 macro file is delivered with [RMOL](#), namely 'rmol.m4', which can be found in, e.g., '/usr/share/aclocal'. When used by a 'configure' script, thanks to the 'AM_PATH_RMOL' macro (specified in the M4 macro file), the following Makefile variables are then defined:

- 'RMOL_VERSION' (e.g., defined to 0.23.0)
- 'RMOL_CFLAGS' (e.g., defined to '-I\${prefix}/include')
- 'RMOL_LIBS' (e.g., defined to '-L\${prefix}/lib -lrmol')

10.6 Using RMOL with dynamic linking

When using static linking some of the library routines in [RMOL](#) are copied into your executable program. This can lead to unnecessary large executables. To avoid having too large executable files you may use dynamic linking instead. Dynamic linking means that the actual linking is performed when the program is executed. This requires that the system is able to locate the shared [RMOL](#) library file during your program execution. If you install the [RMOL](#) library using a non-standard prefix, the 'LD_LIBRARY_PATH' environment variable might be used to inform the linker of the dynamic library location, e.g.:

```
export LD_LIBRARY_PATH=<RMOL installation prefix>/lib:$LD_LIBRARY_PATH
```

11 Test Rules

This section describes rules how the functionality of the IT++ library should be verified. In the 'tests' subdirectory test files are provided. All functionality should be tested using these test files.

11.1 The Test File

Each new IT++ module/class should be accompanied with a test file. The test file is an implementation in C++ that tests the functionality of a function/class or a group of functions/classes called modules. The test file should test relevant parameter settings and input/output relations to guarantee correct functionality of the corresponding classes/functions. The test files should be maintained using version control and updated whenever new functionality is added to the IT++ library.

The test file should print relevant data to a standard output that can be used to verify the functionality. All relevant parameter settings should be tested.

The test file should be placed in the 'tests' subdirectory and should have a name ending with '_test.cpp'.

11.2 The Reference File

Consider a test file named `'module_test.cpp'`. A reference file named `'module_test.ref'` should accompany the test file. The reference file contains a reference printout of the standard output generated when running the test program. The reference file should be maintained using version control and updated according to the test file.

11.3 Testing IT++ Library

One can compile and execute all test programs from `'tests'` subdirectory by typing

```
% make check
```

after successful compilation of the IT++ library.

12 Users Guide

12.1 Table of Contents

- [Introduction](#)
- [Get Started](#)
 - [Get the RMOL library](#)
 - [Build the RMOL project](#)
 - [Build and Run the Tests](#)
 - [Install the RMOL Project \(Binaries, Documentation\)](#)
- [Exploring the Predefined BOM Tree](#)
 - [Forecaster BOM Tree](#)
 - [Optimiser BOM Tree](#)
- [Extending the BOM Tree](#)

12.2 Introduction

The [RMOL](#) library contains classes for revenue management. This document does not cover all the aspects of the [RMOL](#) library. It does however explain the most important things you need to know in order to start using [RMOL](#).

12.3 Get Started

12.3.1 Get the RMOL library

12.3.2 Build the RMOL project

To run the configuration script the first time, go to the top directory (where the [RMOL](#) package has been un-packed), and issue the following command:

- `mkdir -p build && cd build && cmake ..`
- `make`

Note

The [RMOL](#) project can either be cloned from the [Git Repository](#) or downloaded as a tar-ball package from the [Sourceforge Web site](#).

12.3.3 Build and Run the Tests

12.3.4 Install the RMOL Project (Binaries, Documentation)

12.4 Exploring the Predefined BOM Tree

[RMOL](#) predefines a BOM (Business Object Model) tree specific to the airline IT arena.

12.4.1 Forecaster BOM Tree

- [RMOL::EMDetruncator](#)
- [RMOL::Detruncator](#)
- [RMOL::Forecaster](#)

12.4.2 Optimiser BOM Tree

- [RMOL::DPOptimiser](#)
- [RMOL::MCOptimiser](#)
- [RMOL::Optimiser](#)

12.5 Extending the BOM Tree

13 Supported Systems

13.1 Table of Contents

- [Introduction](#)
- [RMOL 0.23.x](#)

- Linux Systems
 - * [Fedora Core 4 with ATLAS](#)
 - * [Gentoo Linux with ACML](#)
 - * [Gentoo Linux with ATLAS](#)
 - * [Gentoo Linux with MKL](#)
 - * [Gentoo Linux with NetLib's BLAS and LAPACK](#)
 - * [Red Hat Enterprise Linux with RMOL External](#)
 - * [SUSE Linux 10.0 with NetLib's BLAS and LAPACK](#)
 - * [SUSE Linux 10.0 with MKL](#)
 - Windows Systems
 - * [Microsoft Windows XP with Cygwin](#)
 - * [Microsoft Windows XP with Cygwin and ATLAS](#)
 - * [Microsoft Windows XP with Cygwin and ACML](#)
 - * [Microsoft Windows XP with MinGW, MSYS and ACML](#)
 - * [Microsoft Windows XP with MinGW, MSYS and RMOL External](#)
 - * [Microsoft Windows XP with MS Visual C++ and Intel MKL](#)
 - Unix Systems
 - * [SunOS 5.9 with RMOL External](#)
- [RMOL 3.9.1](#)
 - [RMOL 3.9.0](#)
 - [RMOL 3.8.1](#)

13.2 Introduction

This page is intended to provide a list of [RMOL](#) supported systems, i.e. the systems on which configuration, installation and testing process of the [RMOL](#) library has been successful. Results are grouped based on minor release number. Therefore, only the latest tests for bug-fix releases are included. Besides, the information on this page is divided into sections dependent on the operating system.

Where necessary, some extra information is given for each tested configuration, e.g. external libraries installed, configuration commands used, etc.

If you manage to compile, install and test the [RMOL](#) library on a system not mentioned below, please let us know, so we could update this database.

13.3 RMOL 0.23.x

13.3.1 Linux Systems

13.3.1.1 Fedora Core 4 with ATLAS

- **Platform:** Intel Pentium 4

- **Operating System:** Fedora Core 4 (x86)
- **Compiler:** g++ (GCC) 4.0.2 20051125
- **RMOL release:** 0.23.0
- **External Libraries:** From FC4 distribution:
 - `fftw3.i386-3.0.1-3`
 - `fftw3-devel.i386-3.0.1-3`
 - `atlas-sse2.i386-3.6.0-8.fc4`
 - `atlas-sse2-devel.i386-3.6.0-8.fc4`
 - `blas.i386-3.0-35.fc4`
 - `lapack.i386-3.0-35.fc4`
- **Tests Status:** All tests PASSED
- **Comments:** RMOL configured with:

```
% CXXFLAGS="-O3 -pipe -march=pentium4" ./configure
```
- **Date:** March 7, 2006
- **Tester:** Tony Ottosson

13.3.1.2 Gentoo Linux with ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s):** g++ (GCC) 3.4.5
- **RMOL release:** 0.23.1
- **External Libraries:** Compiled and installed from portage tree:
 - `sci-libs/acml-3.0.0`
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ACML
% eselect lapack set ACML
```

RMOL configured with:

```
% export CPPFLAGS="-I/usr/include/acml"
% ./configure --with-blas="-lblas"
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.1.3 Gentoo Linux with ATLAS

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86)
- **Compiler:** g++ (GCC) 3.4.5
- **RMOL release:** 0.23.1
- **External Libraries:** Compiled and installed from portage tree:
 - sci-libs/fftw-3.1
 - sci-libs/blas-atlas-3.6.0-r1
 - sci-libs/lapack-atlas-3.6.0
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ATLAS
% eselect lapack set ATLAS
```

RMOL configured with:

```
% ./configure --with-blas="-lblas"
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.1.4 Gentoo Linux with MKL

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86 arch)
- **Compiler:** g++ (GCC) 3.4.5
- **RMOL release:** 0.23.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: /opt/intel/mkl/8.0.1
- **Tests Status:** All tests PASSED
- **Comments:** RMOL configured using the following commands:

```
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/32"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```
- **Date:** February 28, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.1.5 Gentoo Linux with NetLib's BLAS and LAPACK

- **Platform:** Intel Pentium M Centrino
- **Operating System:** Gentoo Linux 2006.0 (x86)
- **Compiler:** g++ (GCC) 3.4.5
- **RMOL release:** 0.23.1
- **External Libraries:** Compiled and installed from portage tree:
 - sci-libs/fftw-3.1
 - sci-libs/blas-reference-19940131-r2
 - sci-libs/cblas-reference-20030223
 - sci-libs/lapack-reference-3.0-r2
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

```
% blas-config reference
% lapack-config reference
```

RMOL configured with:

```
% ./configure --with-blas="-lblas"
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.1.6 Red Hat Enterprise Linux with RMOL External

- **Platform:** Intel Pentium 4
- **Operating System:** Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
- **Compiler:** g++ (GCC) 3.4.4 20050721 (Red Hat 3.4.4-2)
- **RMOL release:** 0.23.0
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from [RMOL External 2.1.1](#) package
- **Tests Status:** All tests PASSED
- **Date:** March 7, 2006
- **Tester:** Erik G. Larsson

13.3.1.7 SUSE Linux 10.0 with NetLib's BLAS and LAPACK

- **Platform:** Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System:** SUSE Linux 10.0 (x86_64)
- **Compiler(s):** g++ (GCC) 4.0.2
- **RMOL release:** 0.23.0
- **External Libraries:** BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:
 - blas-3.0-926
 - lapack-3.0-926
 - fftw3-3.0.1-114
 - fftw3-threads-3.0.1-114
 - fftw3-devel-3.0.1-114
- **Tests Status:** All tests PASSED
- **Comments:** RMOL configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% ./configure --with-lapack="/usr/lib64/liblapack.so.3"
```
- **Date:** March 1, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.1.8 SUSE Linux 10.0 with MKL

- **Platform:** Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System:** SUSE Linux 10.0 (x86_64)
- **Compiler(s):** g++ (GCC) 4.0.2
- **RMOL release:** 0.23.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: /opt/intel/mkl/8.0.1
- **Tests Status:** All tests PASSED
- **Comments:** RMOL configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/em64t"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```
- **Date:** March 1, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.2 Windows Systems

13.3.2.1 Microsoft Windows XP with Cygwin

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **RMOL release:** 0.23.1
- **External Libraries:** Installed from Cygwin's repository:
 - fftw-3.0.1-2
 - fftw-dev-3.0.1-1
 - lapack-3.0-4
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. [RMOL](#) configured with:

```
% ./configure
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.2.2 Microsoft Windows XP with Cygwin and ATLAS

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **RMOL release:** 0.23.1
- **External Libraries:** Installed from Cygwin's repository:
 - fftw-3.0.1-2
 - fftw-dev-3.0.1-1

ATLAS BLAS and LAPACK libraries from [RMOL](#) External 2.1.1 package configured using:

```
% ./configure --enable-atlas --disable-fftw
```
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. [RMOL](#) configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% ./configure
```
- **Date:** March 31, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.2.3 Microsoft Windows XP with Cygwin and ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s):** g++ (GCC) 3.4.4 (cygming special)
- **RMOL release:** 0.23.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. [RMOL](#) configured with:

```
% export LDFLAGS="-L/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```
- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.2.4 Microsoft Windows XP with MinGW, MSYS and ACML

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s):** g++ (GCC) 3.4.4 (mingw special)
- **RMOL release:** 0.23.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. [RMOL](#) configured with:

```
% export LDFLAGS="-L/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```
- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.2.5 Microsoft Windows XP with MinGW, MSYS and RMOL External

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s):** g++ (GCC) 3.4.4 (mingw special)
- **RMOL release:** 0.23.5
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from [RMOL External 2.2.0](#) package
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. [RMOL](#) configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-Wall -O3 -march=athlon-tbird -pipe"
% ./configure --disable-html-doc
```

- **Date:** August 11, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.2.6 Microsoft Windows XP with MS Visual C++ and Intel MKL

- **Platform:** AMD Sempron 3000+
- **Operating System:** Microsoft Windows XP SP2
- **Compiler(s):** Microsoft Visual C++ 2005 .NET
- **RMOL release:** 0.23.5
- **External Libraries:** Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: "C:\Program Files\Intel\MKL\8.1"
- **Tests Status:** Not fully tested. Some [RMOL](#) based programs compiled and run with success.
- **Comments:** Only static library can be built. [RMOL](#) built by opening the "win32\rmol.vcproj" project file in MSVC++ and executing "Build -> Build Solution" command from menu.
- **Date:** August 11, 2006
- **Tester:** Adam Piatyszek (ediap)

13.3.3 Unix Systems

13.3.3.1 SunOS 5.9 with RMOL External

- **Platform:** SUNW, Sun-Blade-100 (SPARC)
- **Operating System:** SunOS 5.9 Generic_112233-10
- **Compiler(s):** g++ (GCC) 3.4.5
- **RMOL release:** 0.23.2
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from [RMOL External 2.1.1](#) package. The following configuration command has been used:

```
% export CFLAGS="-mcpu=ultrasparc -O2 -pipe -funroll-all-loops"
% ./configure
```

- **Tests Status:** All tests PASSED
- **Comments:** [RMOL](#) configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-mcpu=ultrasparc -O2 -pipe"
% ./configure --enable-debug
```

- **Date:** May 15, 2006
- **Tester:** Adam Piatyszek (ediap)

14 RMOL Supported Systems (Previous Releases)

14.1 RMOL 3.9.1

14.2 RMOL 3.9.0

14.3 RMOL 3.8.1

15 BPSK modulation over an AWGN channel

As a first example we will generate a sequence of 500000 random bits {0,1} and BPSK modulate these. Thereafter the BPSK signals will be transmitted over an AWGN channel with a signal-to-noise ratio $E_b/N_0 = 0$ dB. The received signal is then decoded and the number of bit errors are calculated.

```
#include <itpp/itcomm.h>

using namespace itpp;
```

```
//These lines are needed for use of cout and endl
using std::cout;
using std::endl;

int main()
{
    //Scalars
    int N;
    double N0;

    //Vectors
    bvec bits, dec_bits;
    vec symbols, rec;

    //Classes
    BPSK bpsk; //The BPSK modulator/debodulator class
    BEREC berc; //The Bit Error Rate Counter class

    //Init
    N = 500000; //The number of bits to simulate
    N0 = 1;     //0 dB SNR

    //Randomize the random number generator
    RNG_randomize();

    //Generate the bits:
    bits = randb(N);

    //Do the BPSK modulation
    bpsk.modulate_bits(bits, symbols);

    //Add the AWGN
    rec = symbols + sqrt(N0/2)* randn(N);

    //Decode the received bits
    bpsk.demodulate_bits(rec, dec_bits);

    //Count the number of errors
    berc.count(bits,dec_bits);

    //Print the results
    cout << "There were " << berc.get_errors() << " received bits in error." << endl;
    cout << "There were " << berc.get_corrects() << " correctly received bits." << endl;
    cout << "The error probability was " << berc.get_errorrate() << endl;
    cout << "The theoretical error probability is " << 0.5*erfc(1.0) << endl;

    //Exit program:
    return 0;
}
```

When you run this program, the output will look something like this:

```
There were 39224 received bits in error.
There were 460776 correctly received bits.
The error probability was 0.078448
The theoretical error probability is 0.0786496
```

16 Simulation of a convolutional encoder and decoder

In this example we will show how to use the convolutional encoder/decoder class in it++. The Viterbi decoder uses the soft received values.

```
#include <itpp/itcomm.h>

using namespace itpp;

//These lines are needed for use of cout and endl
using std::cout;
using std::endl;

int main()
{
    //Scalars
    int constraint_length, MaxNrofErrors, Nobits, MaxIterations, p, i;
    double Ec, Eb;

    //Vectors
    ivec generators;
    vec EbN0dB, EbN0, N0, ber, trans_symbols, rec_symbols;
    bvec uncoded_bits, coded_bits, decoded_bits;

    //Classes
    BPSK bpsk;
    BEREC berc;
    Convolutional_Code conv_code;
    AWGN_Channel channel;

    /*
    Set up the convolutional encoder/decoder class:
    The generators are given in octal form by adding a zero in front of the numbers
    .
    In this example we will simulate a rate 1/3 code that is listed in J. G. Proakis,
    "Digital communications". The encoder has constraint length 7.
    */
    generators.set_size(3, false);
    generators(0) = 0133;
    generators(1) = 0145;
    generators(2) = 0175;
    constraint_length = 7;
    conv_code.set_generator_polynomials(generators, constraint_length);

    //Init: Calculate some simulation specific parameters:
    Ec = 1.0;
    EbN0dB = linspace(-2, 6, 5);
    EbN0 = inv_dB(EbN0dB);
    Eb = Ec / conv_code.get_rate();
    N0 = Eb * pow(EbN0, -1);
    MaxNrofErrors = 100;
    Nobits = 10000;
    MaxIterations = 10;
    ber.set_size(EbN0dB.length(), false);
    ber.clear();

    //Randomize the random number generators.
    RNG_randomize();

    for (p=0; p<EbN0dB.length(); p++) {
```

```

cout << "Now simulating point " << p+1 << " out of " << EbN0dB.length() << endl;
berc.clear(); //Clear the bit error rate counter.
channel.set_noise(N0(p)/2.0); //Set the noise value of the AWGN channel.

for (i=0; i<MaxIterations; i++) {

    uncoded_bits = randb(Nobits); //The uncoded bits.
    coded_bits = conv_code.encode(uncoded_bits); //The convolutional encoder
        function.
    bpsk.modulate_bits(coded_bits, trans_symbols); //The BPSK modulator.
    rec_symbols = channel( trans_symbols ); //The AWGN channel.
    decoded_bits = conv_code.decode(rec_symbols); //The Viterbi decoder function.
    berc.count(uncoded_bits,decoded_bits); //Count the errors.
    ber(p) = berc.get_errorrate();

    //Break the simulation on this point if sufficient number of bit errors were
    e observed:
    if (berc.get_errors()>MaxNrofErrors) {
        cout << "Breaking on point " << p+1 << " with " << berc.get_errors() << "
        errors." << endl; break;
    }

}

//Print the results:
cout << "BER = " << ber << endl;
cout << "EbN0dB = " << EbN0dB << endl;

//Exit program:
return 0;
}

```

When you run this program, the output will look something like this:

```

Now simulating point 1 out of 5
Breaking on point 1 with 3297 errors.
Now simulating point 2 out of 5
Breaking on point 2 with 781 errors.
Now simulating point 3 out of 5
Breaking on point 3 with 112 errors.
Now simulating point 4 out of 5
Now simulating point 5 out of 5
BER = [0.330858 0.0783743 0.00280983 0 0]
EbN0dB = [-2 0 2 4 6]

```

17 Interleaving and de-interleaving of data

This example shows how to use one of the interleaving classes.

```

#include <itpp/itcomm.h>

using namespace itpp;

```

```

//These lines are needed for use of cout and endl
using std::cout;
using std::endl;

int main()
{
    //Declare scalars and vectors:
    int rows, cols;
    ivec input, output, deinterleaved;

    //Declare the interleaver. The interleaver classes are templated, and therefore
        we must specify
    //the type of the data elements. In this example we are using integers:
    Block_Interleaver<int> my_interleaver;

    //Initialize the interleaver class. Note that this can be done already in the d
        eclaration by writing
    //Block_Interleaver<int> my_interleaver(rows,cols);
    rows = 4;
    cols = 5;
    my_interleaver.set_rows(rows);
    my_interleaver.set_cols(cols);

    //Define the input to the interleaver:
    input = "1:20";

    //Do the interleaving:
    output = my_interleaver.interleave(input);

    //Do the de-interleaving:
    deinterleaved = my_interleaver.deinterleave(output);

    //Print the results:
    cout << "input = " << input << endl;
    cout << "output = " << output << endl;
    cout << "deinterleaved = " << deinterleaved << endl;

    //Exit program:
    return 0;
}

```

When you run this program, the output will look like this:

```

input = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
output = [1 5 9 13 17 2 6 10 14 18 3 7 11 15 19 4 8 12 16 20]
deinterleaved = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

```

18 Writing and reading data from files

Here we will use the `it_file` class to store some data. The program `write_it_file.cpp` looks as follows:

```

#include <itpp/itcomm.h>

using namespace itpp;

```

```

int main()
{
    // Declare the it_file class
    it_file ff;

    // Open a file with the name "it_file_test.it"
    ff.open("it_file_test.it");

    // Create some data to put into the file
    vec a = linspace(1, 20, 20);

    // Put the variable a into the file. The Name("a") tells the file class
    // that the next variable shall be named "a".
    ff << Name("a") << a;

    // Force the file to be written to disc. This is useful when performing
    // iterations and ensures that the information is not stored in any cache
    // memory. In this simple example it is not necessary to flush the file.
    ff.flush();

    // Close the file
    ff.close();

    // Exit program
    return 0;
}

```

When you run this program you will obtain a file called `it_file_test.it` in your current directory. You can read the file into Matlab/Octave to view the data by using the following commands:

```

itload('it_file_test.it')
figure(1); clf;
plot(a)

```

Note: Make sure that `$PREFIX/share/itpp` is in your Matlab/Octave path and that you run the code above from the directory where `it_file_test.it` is located (`$PREFIX` is the IT++ installation prefix; `/usr/local` by default).

The IT++ program `read_it_file.cpp` that reads the file and prints its content can look like this:

```

#include <itpp/itcomm.h>

using namespace itpp;

int main()
{
    // Declare the it_file class
    it_file ff;

    // Open the file "it_file_test.it" for reading
    ff.open("it_file_test.it");

    // Read the variable a from the file. Put result in vector a.
    vec a;
    ff >> Name("a") >> a;
}

```

```

// Print the result
std::cout << "a = " << a << std::endl;

// Exit the program
return 0;
}

```

Here is the output of the program:

```
a = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
```

19 Simulation of LDPC codes the AWGN channel

This program simulates the performance of LDPC codes on the AWGN channel. Since the channel is symmetric, the zero codeword is used.

```

#include <itpp/itcomm.h>

using namespace std;
using namespace itpp;

extern int main(int argc, char **argv)
{
    int Nbits=1000*1000*5000;           // maximum number of bits simulated for any
        SNR point
    int Nbers=2000;                     // target number of bit errors per SNR point
    double BERmin=1e-6;                 // BER at which to terminate simulation
    vec EbN0db = "0.6:0.2:5";

    LDPC_Code C(argv[1]);
    bool single_snr_mode=false;
    if (argc==3) {
        double x;
        sscanf(argv[2], "%lf", &x);
        EbN0db.set_size(1);
        EbN0db(0)=x;
        single_snr_mode=true;
    }

    cout << "Running with Eb/N0: " << EbN0db << endl;

    // High performance: 2500 iterations, high resolution LLR algebra
    C.setup_decoder("bp", "2500 1 0");

    // Alt. setting -- High speed: 50 iterations, logmax approximation
    // C.setup_decoder("bp", "50 1 0", LLR_calc_unit(12,0,7));

    cout << C << endl;

    int N = C.get_nvar();                // number of bits per codeword
    BPSK Mod;
    bvec bitsin = zeros_b(N);
    vec s = Mod.modulate_bits(bitsin);

    RNG_randomize();
    for (int j=0; j<length(EbN0db); j++) {
        // noise variance is N0/2 per dimension

```



```

double N0 = pow(10.0, -EbN0db(j)/10.0) / C.get_rate();
AWGN_Channel chan(N0/2);
BERC berc; // counters for coded and uncoded BER
BLERC ferc; // counter for coded FER
ferc.set_blocksize(N);
for (long int i=0; i<Nbits; i+=C.get_nvar()) {
    // Received data
    vec x = chan(s);

    // Demodulate
    vec softbits=Mod.demodulate_soft_bits(x,N0);

    //Decode the received bits
    bvec bitsout=C.decode(softbits);

    //Count the number of errors
    berc.count(bitsin,bitsout);
    ferc.count(bitsin,bitsout);

    if (single_snr_mode) {
        cout << "Eb/N0=" << EbN0db(j) << " Simulated "
              << ferc.get_total_blocks() << " frames and "
              << berc.get_total_bits() << " bits. "
              << "Obtained " << berc.get_errors() << " bit errors. "
              << " BER: " << berc.get_errorrate()
              << " FER: " << ferc.get_errorrate() << endl;
        cout.flush();
    } else {
        if (berc.get_errors()>Nbers) { break;}
    }
}

cout << "Eb/N0=" << EbN0db(j) << " Simulated "
      << ferc.get_total_blocks() << " frames and "
      << berc.get_total_bits() << " bits. "
      << "Obtained " << berc.get_errors() << " bit errors. "
      << " BER: " << berc.get_errorrate()
      << " FER: " << ferc.get_errorrate() << endl;
cout.flush();
if (berc.get_errorrate()<BERmin) { break; }
}
return 0;
}

```

To simulate the code "RU_10000.it" over a range of SNR (see [Generation of LDPC codes](#) for how to generate codec)

```
./ldpc_bersim_awgn RU_10000.it
```

To simulate at $E_b/N_0=1.1$ dB

```
./ldpc_bersim_awgn RU_10000.it 1.1
```

20 Generation of LDPC codes

```
// Generate some example LDPC codes
```

```

#include <itpp/itcomm.h>

using namespace itpp;
using namespace std;

extern int main(int argc, char **argv)
{
    { // This generates a random regular (3,6) code with 500 bits
      cout << "===== RANDOM (3,6) CODE =====" << endl;
      LDPC_Parity_Matrix H;
      H.generate_regular_ldpc(500,3,6,
                             "rand", // random unstructured matrix
                             "500 10"); // optimize girth

      H.display_stats();
      LDPC_Code C1(H);
      C1.save_to_file("random_3_6_code.it");
    }

    { // This is the code "204.33.484 (N=204,K=102,M=102,R=0.5)" from
      // David MacKay's database over sparse-graph code. It can be
      // obtained with "wget
      // http://www.inference.phy.cam.ac.uk/mackay/codes/EN/C/204.33.484"
      cout << "===== MACKAY CODE =====" << endl;
      LDPC_Parity_Matrix H("204.33.484","alist");
      H.display_stats();
      LDPC_Generator_Matrix G(H);
      LDPC_Code C(H,G);
      C.save_to_file("mackay_204.33.484.it");

      // Now produce a girth-optimized version of this code by removing
      // cycles. This slightly improves the performance at high SNR.
      H.cycle_removal_MGW(12);
      LDPC_Generator_Matrix G1(H);
      LDPC_Code C1(H,G1);
      C1.save_to_file("mackay_204.33.484_opt.it");
    }

    // Irregular 1/2-rate codes optimized for the AWGN channel. The
    // degree distributions are taken from Richardson & Urbanke,
    // Trans. IT 2001.

    { // 1000 bits
      cout << "===== IRREGULAR CODE 1000 BITS =====" << endl;
      LDPC_Parity_Matrix H;
      H.generate_irregular_ldpc(1000,
                              "0 0.27684 0.28342 0 0 0 0 0 0.43974",
                              "0 0 0 0 0 0.01568 0.85244 0.13188",
                              "rand", // random unstructured matrix
                              "500 8"); // optimize girth

      LDPC_Code C(H);
      C.save_to_file("RU_1000.it");
    }

    { // 10000 bits (takes a few minutes to run)
      cout << "===== IRREGULAR CODE 10000 BITS =====" << endl;
      LDPC_Parity_Matrix H;
      H.generate_irregular_ldpc(10000,"0 0.21991 0.23328 0.02058 0 0.08543 0.06540
        0.04767 \
                                0.01912 0 0 0 0 0 0 0 0 0.08064 0.22798",
                                "0 0 0 0 0 0 0.64854 0.34747 0.00399",
                                "rand", // random unstructured matrix
                                "150 8"); // optimize
    }
}

```

```

LDPC_Code C(H);
C.save_to_file("RU_10000.it");
}

{ // 100000 bits (takes a while to run)
cout << "===== IRREGULAR CODE 100000 BITS =====" << endl;
LDPC_Parity_Matrix H;
H.generate_irregular_ldpc(100000,"0 0.1712 0.21053 0.00273 0 0 0.00009 0.1526
9 0.09227 \
0.07212 0 0 0 0 \
0.02802 0 0 0 0 0.01206 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0.25830",
"0 0 0 0 0 0 0 0 0.33620 0.08883 0.57497",
"rand",
"40 4"); // less aggressive optimization

LDPC_Code C(H);
C.save_to_file("RU_100000.it");
}

exit(0);

{ // 1000000 bits (THIS CODE REQUIRES ABOUT 450 MB TO STORE AND 2GB
// INTERNAL MEMORY TO GENERATE)
cout << "===== IRREGULAR CODE 1000000 BITS =====" << endl;
LDPC_Parity_Matrix H;
H.generate_irregular_ldpc(1000000,"0 0.1712 0.21053 0.00273 0 0 0.00009 0.152
69 0.09227 \
0.07212 0 0 0 0 \
0.02802 0 0 0 0 0.01206 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0.25830",
"0 0 0 0 0 0 0 0 0.33620 0.08883 0.57497",
"rand",
"0 0"); // no optimization here

LDPC_Code C(H);
C.save_to_file("RU_1000000.it");
}
}

```

21 Conversion table between IT++ syntax and Matlab/Octave

Here we provide a conversion table between Matlab/Octave and IT++ syntax. This table is intended to help with the transition from Matlab/Octave programming to IT++, but it is not an exhaustive list of possible operations.

In what follows,

- a, b denote vectors (assumed to be column vectors in Matlab/Octave notation),
- A, B denote matrices,
- x is a scalar,
- k, l, n are indices

Vector indexing and manipulation:

```

a.length()           // length(a)
a(0)                 // a(1)
a(1)                 // a(2)
a(k-1)               // a(k)
a(k-1)=x;  or a.set(k-1,x); // a(k)=x;
a.left(k)            // a(1:k)
a.right(k)           // a(end-k+1:end)
a.mid(k,l)           // a(k:k+l-1)
a.del(k);            // a=[a(1:k-1); a(k+1:end)];
concat(a,b)          // [a; b] (or [a.' b.'].')
a.clear();           // a=zeros(size(a)); (or a=complex(zeros(si
                      ze(a)));)
to_cmat(a)           // complex(a) (assuming a real-valued)

```

Note that indexing in IT++ starts at 0, whereas indexing in Matlab starts at 1. Also note that Matlab/Octave does distinguish between column and row vectors, whereas IT++ does not.

Matrix indexing and manipulation:

```

A.rows()             // size(A,1)
A.cols()             // size(A,2)
A(k-1,l-1)           // A(k,l)
A.set(k-1,l-1,x);    // A(k,l)=x;
A.get_col(k-1)       // A(:,k)
A.get_row(k-1)       // A(k,:)
A.set_col(k-1,a);    // A(:,k)=a;
A.set_row(k-1,a);    // A(k,:)=a;
A.append_row(a);     // A=[A; a.'];
A.append_col(a);     // A=[A a];
A.transpose()        // A.'
A.hermitian_transpose() // A'
A.clear();           // A=zeros(size(A)); (or A=complex(zeros(si
                      ze(A)));)
to_cmat(A)           // complex(A) (assuming a real-valued)

```

Some vector and matrix algebra:

```

a+b                 // a+b
a-b                 // a-b
elem_mult(a,b)      // a.*b (elementwise product)
a*b                 // a.'*b (inner product)
conj(a)*b           // a'*b (inner product)
outer_product(a,b)  // a*b.' (outer product)
elem_div(a,b)       // a./b
A+B;                // A+B;
A-B;                // A-B;
A*B;                // A*B;
elem_mul(A,B)       // A.*B
elem_div(A,B)       // A./B
A\b;                // ls_solve_od(A,b); (assuming the system
                      is overdetermined)

```

Special matrices and vectors:

```

zeros(n,n)          // zeros(n,n)
zeros_c(n,n)        // complex(zeros(n,n))
eye(n)              // eye(n)
eye_c(n)            // complex(eye(n))
linspace(alpha,beta,n) // linspace(alpha,beta,n)

```

Hardcoded initializations:

```

mat X="1.1 1.2; 2.1; 2.2";           // X=[1.1 1.2; 2.1 2.2];
ivec a="1 2 3 4 5";                 // a=[1; 2; 3; 4; 5]; (or a=[1 2 3 4 5].';)
ivec a="1:-3:-8";                   // a=1:-3:-8;

```

22 MIMO (spatial multiplexing) with convolutional coding

This example demonstrates how to use the `Modulator_ND` (MIMO) class for soft-output demodulation. The program simulates a simple convolutionally coded spatial-multiplexing (V-BLAST style) MIMO system with maximum-likelihood, alternatively zero-forcing, demodulation and soft Viterbi decoding, but no iteration between the demodulator and the decoder.

```

#include <itpp/itcomm.h>

using std::cout;
using std::endl;
using namespace itpp;
using namespace std;

/* Zero-forcing detector with ad hoc soft information. This function
   applies the ZF (pseudoinverse) linear filter to the received
   data. This results in effective noise with covariance matrix
   inv(H'H)*sigma2. The diagonal elements of this noise covariance
   matrix are taken as noise variances per component in the processed
   received data but the noise correlation is ignored.

*/

void ZF_demod(ND_UQAM &channel, ivec &LLR_apr, ivec &LLR_apost, double sigma2, c
    mat &H, cvec &y)
{
    it_assert(H.rows()>=H.cols(), "ZF_demod() - underdetermined systems not tolerate
        d");
    cvec shat=ls_solve_od(H,y); // the ZF solution
    vec Sigma2=real(diag(inv(H.hermitian_transpose()*H)))*sigma2; // noise covaria
        nce of shat
    cvec h(length(shat));
    for (int i=0; i<length(shat); i++) {
        shat(i) = shat(i)/sqrt(Sigma2(i));
        h(i) = 1.0/sqrt(Sigma2(i));
    }
    channel.map_demod(LLR_apr,LLR_apost,1.0,h,shat);
}

extern int main(int argc, char **argv)
{
    // -- modulation and channel parameters (taken from command line input) --
    int nC; // type of constellation (1=QPSK, 2=16-QAM, 3=64-QA
        M)
    int nRx; // number of receive antennas
    int nTx; // number of transmit antennas
    int TC; // coherence time (number of channel vectors with sa
        me H)

```

```

if (argc!=5) {
    cout << "Usage: cm nTx nRx nC Tc" << endl << "Example: cm 2 2 1 100000 (2x2 Q
        PSK MIMO on slow fading channel)" << endl;
    exit(1);
} else {
    sscanf(argv[1],"%i",&nTx);
    sscanf(argv[2],"%i",&nRx);
    sscanf(argv[3],"%i",&nC);
    sscanf(argv[4],"%i",&Tc);
}

cout << "Initializing.. " << nTx << " TX antennas, " << nRx << " RX antennas, "
    << (1<<nC) << "-PAM per dimension, coherence time " << Tc << endl;

// -- simulation control parameters --
const vec EbN0db = "-5:0.5:50";           // SNR range
const int Nmethods =2;                     // number of demodulators to try
const long int Nbitsmax=50*1000*1000;      // maximum number of bits to ever simula
    te per SNR point
const int Nu = 1000;                       // length of data packet (before applyin
    g channel coding)

int Nbers, Nfers;                          // target number of bit/frame errors per SNR poi
    nt
double BERmin, FERmin;                     // BER/FER at which to terminate simulation
if (Tc==1) {                               // Fast fading channel, BER is of primary interest
    BERmin = 0.001;                        // stop simulating a given method if BER<this value
    FERmin = 1.0e-10;                     // stop simulating a given method if FER<this value
    Nbers = 1000;                         // move to next SNR point after counting 1000 bit errors

    Nfers = 200;                          // do not stop on this condition
} else {                                   // Slow fading channel, FER is of primary interest here
    BERmin = 1.0e-15;                     // stop simulating a given method if BER<this value
    FERmin = 0.01;                        // stop simulating a given method if FER<this value
    Nbers = -1;                           // do not stop on this condition
    Nfers = 200;                          // move to next SNR point after counting 200 frame error
    s
}

// -- Channel code parameters --
Convolutional_Code code;
ivec generator(3);
generator(0)=0133; // use rate 1/3 code
generator(1)=0165;
generator(2)=0171;
double rate=1.0/3.0;
code.set_generator_polynomials(generator, 7);
bvec dummy;
code.encode_tail(randb(Nu),dummy);
const int Nc = length(dummy); // find out how long the coded blocks are

// ===== Initialize =====

const int Nctx = (int) (2*nC*nTx*ceil(double(Nc)/double(2*nC*nTx))); // Total
    number of bits to transmit
const int Nvec = Nctx/(2*nC*nTx); // Number of channel vectors to tran
    smit
const int Nbitspvec = 2*nC*nTx; // Number of bits per channel vector

```

```

// initialize MIMO channel with uniform QAM per complex dimension and Gray coding
ND_UQAM chan;
chan.set_Gray_QAM(nTx, 1 << (2*nC));
cout << chan << endl;

// initialize interleaver
Sequence_Interleaver<bin> sequence_interleaver_b(Nctx);
Sequence_Interleaver<int> sequence_interleaver_i(Nctx);
sequence_interleaver_b.randomize_interleaver_sequence();
sequence_interleaver_i.set_interleaver_sequence(sequence_interleaver_b.get_interleaver_sequence());

// RNG_randomize();

Array<cvec> Y(Nvec); // received data
Array<cmat> H(Nvec/Tc+1); // channel matrix (new matrix for each coherence interval)

ivec Contflag = ones_i(Nmethods); // flag to determine whether to run a given demodulator
if (pow(2.0, nC*2.0*nTx) > 256) { // ML decoder too complex..
    Contflag(1)=0;
}
if (nTx > nRx) {
    Contflag(0)=0; // ZF not for underdetermined systems
}
cout << "Running methods: " << Contflag << endl;

cout.setf(ios::fixed, ios::floatfield);
cout.setf(ios::showpoint);
cout.precision(5);

// ===== Run simulation =====
for (int nsnr=0; nsnr<length(EbN0db); nsnr++) {
    const double Eb=1.0; // transmitted energy per information bit
    const double N0 = pow(10.0, -EbN0db(nsnr)/10.0);
    const double sigma2=N0; // Variance of each scalar complex noise sample
    const double Es=rate*2*nC*Eb; // Energy per complex scalar symbol
    // (Each transmitted scalar complex symbol contains rate*2*nC
    // information bits.)
    const double Ess=sqrt(Es);

    Array<BERC> berc(Nmethods); // counter for coded BER
    Array<BERC> bercu(Nmethods); // counter for uncoded BER
    Array<BLERC> ferc(Nmethods); // counter for coded FER

    for (int i=0; i<Nmethods; i++) {
        ferc(i).set_blocksize(Nu);
    }

    long int nbits=0;
    while (nbits<Nbitsmax) {
        nbits += Nu;

        // generate and encode random data
        bvec inputbits = randb(Nu);
        bvec txbits;
        code.encode_tail(inputbits, txbits);
        // coded block length is not always a multiple of the number of
        // bits per channel vector
        txbits=concat(txbits, randb(Nctx-Nc));
    }
}

```

```

txbits = sequence_interleaver_b.interleave(txbits);

// -- generate channel and data ----
for (int k=0; k<Nvec; k++) {
    /* A complex valued channel matrix is used here. An
       alternative (with equivalent result) would be to use a
       real-valued (structured) channel matrix of twice the
       dimension.
    */
    if (k%Tc==0) {          // generate a new channel realization every Tc inter
vals
        H(k/Tc) = Ess*randn_c(nRx,nTx);
    }

    // modulate and transmit bits
    bvec bitstamp = txbits(k*2*nTx*nC, (k+1)*2*nTx*nC-1);
    cvec x=chan.modulate_bits(bitstamp);
    cvec e=sqrt(sigma2)*randn_c(nRx);
    Y(k) = H(k/Tc)*x+e;
}

// -- demodulate --
Array<QLLRvec> LLRin(Nmethods);
for (int i=0; i<Nmethods; i++) {
    LLRin(i) = zeros_i(Nctx);
}

QLLRvec llr_apr=zeros_i(nC*2*nTx); // no a priori input to demodulator
QLLRvec llr_apost=zeros_i(nC*2*nTx);
for (int k=0; k<Nvec; k++) {
    // zero forcing demodulation
    if (Contflag(0)) {
        ZF_demod(chan,llr_apr,llr_apost,sigma2,H(k/Tc),Y(k));
        LLRin(0).set_subvector(k*Nbitspvec, (k+1)*Nbitspvec-1,llr_apost);
    }

    // ML demodulation
    if (Contflag(1)) {
        chan.map_demod(llr_apr, llr_apost, sigma2, H(k/Tc), Y(k));
        LLRin(1).set_subvector(k*Nbitspvec, (k+1)*Nbitspvec-1,llr_apost);
    }
}

// -- decode and count errors --
for (int i=0; i<Nmethods; i++) {
    bvec decoded_bits;
    if (Contflag(i)) {
        bercu(i).count(txbits(0,Nc-1),LLRin(i)(0,Nc-1)<0); // uncoded BER
        LLRin(i) = sequence_interleaver_i.deinterleave(LLRin(i),0);
        // QLLR values must be converted to real numbers since the convolutiona
l decoder wants this
        vec llr=chan.get_llrcalc().to_double(LLRin(i).left(Nc));
        //      llr=-llr; // UNCOMMENT THIS LINE IF COMPILING WITH 3.10.5 OR EA
RLIER (BEFORE HARMONIZING LLR CONVENTIONS)
        code.decode_tail(llr,decoded_bits);
        berc(i).count(inputbits(0,Nu-1),decoded_bits(0,Nu-1)); // coded BER
        ferc(i).count(inputbits(0,Nu-1),decoded_bits(0,Nu-1)); // coded FER
    }
}

/* Check whether it is time to terminate the simulation.
   Terminate when all demodulators that are still running have

```



```

        counted at least Nbers or Nfers bit/frame errors. */
int minber=1000000;
int minfer=1000000;
for (int i=0; i<Nmethods; i++) {
    if (Contflag(i)) {
        minber=min(minber,round_i(berc(i).get_errors()));
        minfer=min(minfer,round_i(ferc(i).get_errors()));
    }
}
if (Nbers>0 && minber>Nbers) { break;}
if (Nfers>0 && minfer>Nfers) { break;}
}

cout << "-----" << endl;
cout << "Eb/NO: " << EbN0db(nsnr) << " dB. Simulated " << nbits << " bits." <
< endl;
cout << " Uncoded BER: " << bercu(0).get_errorrate() << " (ZF);      " << berc
u(1).get_errorrate() << " (ML)" << endl;
cout << " Coded BER:   " << berc(0).get_errorrate() << " (ZF);      " << berc
(1).get_errorrate() << " (ML)" << endl;
cout << " Coded FER:   " << ferc(0).get_errorrate() << " (ZF);      " << ferc
(1).get_errorrate() << " (ML)" << endl;
cout.flush();

/* Check wheter it is time to terminate simulation. Stop when all
methods have reached the min BER/FER of interest. */
int contflag=0;
for (int i=0; i<Nmethods; i++) {
    if (Contflag(i)) {
        if (berc(i).get_errorrate()>BERmin) { contflag=1; } else { Contflag(i)
= 0; }
        if (ferc(i).get_errorrate()>FERmin) { contflag=1; } else { Contflag(i)
= 0; }
    }
}
if (contflag) { continue; } else {break; }
}

return 0;
}

```

To run the program,

```
mimoconv nTx nRx nC Tc
```

where

- nTx=number of transmit antennas
- nRx=number of receive antennas
- nC=constellation (1=QPSK, 2=16-QAM, 3=64-QAM)
- Tc=coherence time (channel uses)

23 Using Mixture of Gaussians (MOG) module to model data

This example demonstrates how to find the parameters of a MOG model via using the kmeans and EM based optimisers. Synthetic data is utilised.

```
#include <itpp/itstat.h>

#include <fstream>
#include <iostream>
#include <iomanip>
#include <ios>

using std::cout;
using std::endl;
using std::fixed;
using std::setprecision;

using namespace itpp;

int main() {

    bool print_progress = false;

    //
    // first, let's generate some synthetic data

    int N = 100000; // number of vectors
    int D = 3;      // number of dimensions
    int K = 5;      // number of Gaussians

    Array<vec> X(N); for(int n=0;n<N;n++) { X(n).set_size(D); X(n) = 0.0; }

    // the means

    Array<vec> mu(K);
    mu(0) = "-6, -4, -2";
    mu(1) = "-4, -2, 0";
    mu(2) = "-2, 0, 2";
    mu(3) = " 0, +2, +4";
    mu(4) = "+2, +4, +6";

    // the diagonal variances

    Array<vec> var(K);
    var(0) = "0.1, 0.2, 0.3";
    var(1) = "0.2, 0.3, 0.1";
    var(2) = "0.3, 0.1, 0.2";
    var(3) = "0.1, 0.2, 0.3";
    var(4) = "0.2, 0.3, 0.1";

    cout << fixed << setprecision(3);
    cout << "user configured means and variances:" << endl;
    cout << "mu = " << mu << endl;
    cout << "var = " << var << endl;

    // randomise the order of Gaussians "generating" the vectors
    I_Uniform_RNG rnd_uniform(0, K-1);
    ivec gaus_id = rnd_uniform(N);

    ivec gaus_count(K); gaus_count = 0;
```

```

Array<vec> mu_test(K); for(int k=0;k<K;k++) { mu_test(k).set_size(D); mu_test(
    k) = 0.0; }
Array<vec> var_test(K); for(int k=0;k<K;k++) { var_test(k).set_size(D); var_te
    st(k) = 0.0; }

Normal_RNG rnd_normal;
for(int n=0;n<N;n++) {

    int k = gaus_id(n);
    gaus_count(k)++;

    for(int d=0;d<D;d++) {
        rnd_normal.setup( mu(k)(d), var(k)(d) );
        double tmp = rnd_normal();
        X(n)(d) = tmp;
        mu_test(k)(d) += tmp;
    }
}

//
// find the stats for the generated data

for(int k=0;k<K;k++) mu_test(k) /= gaus_count(k);

for(int n=0;n<N;n++) {
    int k = gaus_id(n);

    for(int d=0;d<D;d++) {
        double tmp = X(n)(d) - mu_test(k)(d);
        var_test(k)(d) += tmp*tmp;
    }
}

for(int k=0;k<K;k++) var_test(k) /= (gaus_count(k)-1.0);

cout << endl << endl;
cout << fixed << setprecision(3);
cout << "stats for X:" << endl;

for(int k=0;k<K;k++) {
    cout << "k = " << k << " count = " << gaus_count(k) << " weight = " << gaus
        _count(k)/double(N) << endl;
    for(int d=0;d<D;d++) cout << " d = " << d << " mu_test = " << mu_test(k)(d)
        << " var_test = " << var_test(k)(d) << endl;
    cout << endl;
}

// make a model with initial values (zero mean and unit variance)
// the number of gaussians and dimensions of the model is specified here

MOG_diag mog(K,D);

cout << endl;
cout << fixed << setprecision(3);
cout << "mog.avg_log_lhood(X) = " << mog.avg_log_lhood(X) << endl;

//
// find initial parameters via k-means (which are then used as seeds for EM bas
    ed optimisation)

cout << endl << endl;

```

```

cout << "running kmeans optimiser" << endl << endl;

MOG_diag_kmeans(mog, X, 10, 0.5, true, print_progress);

cout << fixed << setprecision(3);
cout << "mog.get_means() = " << endl << mog.get_means() << endl;
cout << "mog.get_diag_covs() = " << endl << mog.get_diag_covs() << endl;
cout << "mog.get_weights() = " << endl << mog.get_weights() << endl;

cout << endl;
cout << "mog.avg_log_lhood(X) = " << mog.avg_log_lhood(X) << endl;

//
// EM ML based optimisation

cout << endl << endl;
cout << "running ML optimiser" << endl << endl;

MOG_diag_ML(mog, X, 10, 0.0, 0.0, print_progress);

cout << fixed << setprecision(3);
cout << "mog.get_means() = " << endl << mog.get_means() << endl;
cout << "mog.get_diag_covs() = " << endl << mog.get_diag_covs() << endl;
cout << "mog.get_weights() = " << endl << mog.get_weights() << endl;

cout << endl;
cout << "mog.avg_log_lhood(X) = " << mog.avg_log_lhood(X) << endl;

return 0;
}

```

24 Simulation of QPSK modulation on an AWGN channel

In this example we will introduce a few new features compared to the BPSK example. We will use the `it_ifile` class to store the results of the simulation. We will use the `Real_Timer` class to measure the execution time of our program. Furthermore we will use one of the channel classes, the `AWGN_Channel`. We will also show how to read the results in to Matlab with the `load_it` function. The program is as follows:

```

#include <itpp/itcomm.h>

using namespace itpp;

//These lines are needed for use of cout and endl
using std::cout;
using std::endl;

int main()
{
    //Declarations of scalars and vectors:
    int i, Number_of_bits;
    double Ec, Eb;
    vec EbN0dB, EbN0, N0, noise_variance, bit_error_rate; //vec is a vector contain
        ing double
    bvec transmitted_bits, received_bits; //bvec is a vector contain
        ing bits
    cvec transmitted_symbols, received_symbols; //cvec is a vector contain

```

```

        ning double_complex

//Declarations of classes:
QPSK qpsk;                //The QPSK modulator class
AWGN_Channel awgn_channel; //The AWGN channel class
it_file ff;               //For saving the results to file
BERC berc;                //Used to count the bit errors
Real_Timer tt;            //The timer used to measure the execution time

//Reset and start the timer:
tt.tic();

//Init:
Ec = 1.0;                //The transmitted energy per QPSK symbol is 1.
Eb = Ec / 2.0;           //The transmitted energy per bit is 0.5.
EbN0dB = linspace(0.0,9.0,10); //Simulate for 10 Eb/N0 values from 0 to 9 dB.
EbN0 = inv_dB(EbN0dB);   //Calculate Eb/N0 in a linear scale instead of d
                           B.
N0 = Eb * pow(EbN0,-1.0); //N0 is the variance of the (complex valued) noi
                           se.
Number_of_bits = 100000; //One hundred thousand bits is transmitted for e
                           ach Eb/N0 value

//Allocate storage space for the result vector.
//The "false" argument means "Do not copy the old content of the vector to the
                           new storage area."
bit_error_rate.set_size(EbN0dB.length(),false);

//Randomize the random number generators in it++:
RNG_randomize();

//Iterate over all EbN0dB values:
for (i=0; i<EbN0dB.length(); i++) {

    //Show how the simulation progresses:
    cout << "Now simulating Eb/N0 value number " << i+1 << " of " << EbN0dB.lengt
        h() << endl;

    //Generate a vector of random bits to transmit:
    transmitted_bits = randb(Number_of_bits);

    //Modulate the bits to QPSK symbols:
    transmitted_symbols = qpsk.modulate_bits(transmitted_bits);

    //Set the noise variance of the AWGN channel:
    awgn_channel.set_noise(N0(i));

    //Run the transmitted symbols through the channel using the () operator:
    received_symbols = awgn_channel(transmitted_symbols);

    //Demodulate the received QPSK symbols into received bits:
    received_bits = qpsk.demodulate_bits(received_symbols);

    //Calculate the bit error rate:
    berc.clear(); //Clear the bit error rate counte
        r
    berc.count(transmitted_bits,received_bits); //Count the bit errors
    bit_error_rate(i) = berc.get_errrorrate(); //Save the estimated BER in the r
        esult vector
}

```

```

tt.toc();

//Print the results:
cout << endl;
cout << "EbN0dB = " << EbN0dB << " [dB]" << endl;
cout << "BER = " << bit_error_rate << endl;
cout << "Saving results to ./qpsk_result_file.it" << endl;
cout << endl;

//Save the results to file:
ff.open("qpsk_result_file.it");
ff << Name("EbN0dB") << EbN0dB;
ff << Name("ber") << bit_error_rate;
ff.close();

//Exit program:
return 0;
}

```

When you run this program, the output will look something like this:

```

Now simulating Eb/N0 value number 1 of 10
Now simulating Eb/N0 value number 2 of 10
Now simulating Eb/N0 value number 3 of 10
Now simulating Eb/N0 value number 4 of 10
Now simulating Eb/N0 value number 5 of 10
Now simulating Eb/N0 value number 6 of 10
Now simulating Eb/N0 value number 7 of 10
Now simulating Eb/N0 value number 8 of 10
Now simulating Eb/N0 value number 9 of 10
Now simulating Eb/N0 value number 10 of 10
Elapsed time = 0.460899 seconds

EbN0dB = [0 1 2 3 4 5 6 7 8 9] [dB]
BER = [0.07968 0.0559 0.03729 0.02294 0.01243 0.0058 0.0025 0.00076 0.00013 6e-05
      ]
Saving results to ./qpsk_result_file.it

```

Now it is time to plot the simulation results in Matlab and compare with theory using the Matlab code below. The results will be stored in a file called "qpsk_result_file.it". Make sure load_it.m is in your Matlab path (look in /matlab) and that you run the example code below from the directory where qpsk_result_file.it is located

```

figure(1); clf;
load_it qpsk_result_file.it
h1 = semilogy(EbN0dB,ber,'*-r'); hold on
ebn0db = 0:.1:10;
ebn0 = 10.^(ebn0db/10);
Pb = 0.5 * erfc(sqrt(ebn0));
h2 = semilogy(ebn0db,Pb);
set(gca,'fontname','times','fontsize',16);
xlabel('\it E_b / \it N_0 [dB]');
ylabel('BER')
title('QPSK on an AWGN Channel');
legend([h1 h2],'Simulation','Theory');
grid on;

```

25 Generating a correlated Rayleigh fading process

In this example we will generate a correlated Rayleigh fading process with a normalized Doppler frequency equal to 0.1. The normalized Doppler is defined as the multiplication of the maximum Doppler frequency by the sampling time (i.e. $f_d = F_d T_s$).

```
#include <itpp/itcomm.h>

using namespace itpp;

int main()
{
    // Declare my_channel variable as an instance of the Rayleigh_Channel
    // class
    TDL_Channel my_channel;

    // The normalized Doppler frequency is set to 0.1
    double norm_dopp = 0.1;
    my_channel.set_norm_doppler(norm_dopp);

    // Generate nrof_samples of the fading process and store them in ch_coeffs
    // matrix
    int nrof_samples = 10000;
    cmatrix ch_coeffs;
    my_channel.generate(nrof_samples, ch_coeffs);

    // Open an output file "rayleigh_test.it"
    it_file ff("rayleigh_test.it");

    // Save channel coefficients to the output file
    ff << Name("ch_coeffs") << ch_coeffs;

    // Close the output file
    ff.close();

    // Exit program
    return 0;
}
```

You can use Matlab or Octave to examine the channel fading process that is stored in the output file `rayleigh_test.it`. Try the following code to view a part of the fading process:

```
itload("rayleigh_test.it")
figure(1); clf;
semilogy(abs(ch_coeffs(1:200)))
```

Note: Make sure that the folder `$PREFIX/share/itpp` is included your Matlab/Octave path variable (`$PREFIX` is the IT++ installation prefix: `/usr/local` by default).

26 Simulation of a Reed-Solomon Block Code

A Reed-Solomon code is a q^m -ary BCH code of length $q^m - 1$. The generator polynomial for a t -error correcting code is $g(x) = (x - \alpha)(x - \alpha^1) \dots (x - \alpha^{2t-1})$. The decoder

uses the Berlekamp-Massey algorithm for decoding as described in: S. B. Wicker, "Error Control Systems for digital communication and storage," Prentice Hall. The following example simulates a binary (i.e. $q = 2$) Reed-Solomon code with parameters m and t :

```
#include <itpp/itcomm.h>

using namespace itpp;

//These lines are needed for use of cout and endl
using std::cout;
using std::endl;

int main()
{
    //Scalars and vectors:
    int m, t, n, k, q, NumBits, NumCodeWords;
    double p;
    bvec uncoded_bits, coded_bits, received_bits, decoded_bits;

    //Set parameters:
    NumCodeWords = 1000; //Number of Reed-Solomon code-words to simulate
    p = 0.01;             //BSC Error probability
    m = 3;                //Reed-Solomon parameter m
    t = 2;                //Reed-Solomon parameter t

    cout << "Number of Reed-Solomon code-words to simulate: " << NumCodeWords << endl;
    cout << "BSC Error probability : " << p << endl;
    cout << "RS m: " << m << endl;
    cout << "RS t: " << t << endl;

    //Classes:
    Reed_Solomon reed_solomon(m,t);
    BSC bsc(p);
    BEREC berc;

    RNG_randomize();

    //Calculate parameters for the Reed-Solomon Code:
    n = round_i(pow(2.0,m)-1);
    k = round_i(pow(2.0,m)-1-2*t);
    q = round_i(pow(2.0,m));

    cout << "Simulating an Reed-Solomon code with the following parameters:" << endl;
    cout << "n = " << n << endl;
    cout << "k = " << k << endl;
    cout << "q = " << q << endl;

    NumBits = m * k * NumCodeWords;
    uncoded_bits = randb(NumBits);
    coded_bits = reed_solomon.encode(uncoded_bits);
    received_bits = bsc(coded_bits);
    decoded_bits = reed_solomon.decode(received_bits);

    berc.count(uncoded_bits,decoded_bits);
    cout << "The bit error probability after decoding is " << berc.get_errorrte()
        << endl;

    //Exit program:
    return 0;
}
```



```
}
```

A typical run of this program can look like this:

```
Number of Reed-Solomon code-words to simulate: 1000
BSC Error probability : 0.01
RS m: 3
RS t: 2
Simulating an Reed-Solomon code with the following parameters:
n = 7
k = 3
q = 8
The bit error probability after decoding is 0.000333333
```

27 Simulation of a Multicode CDMA system on an AWGN channel

In this example we will introduce the multi-code spreading class `Multicode_Spread_2d`. This is the most general spreading class available in `it++`. Different spreading codes may be assigned to the I and Q branches. The number of multiple spreading codes and the spreading factor is determined by the number of rows and columns respectively that is used when calling the member function `set_codes`. In this example we will use four Hadamard sequenced of length four, and the same spreading sequences will be used for the I and Q branches.

```
#include <itpp/itcomm.h>

using namespace itpp;

//These lines are needed for use of cout and endl
using std::cout;
using std::endl;

int main()
{
    //Scalars:
    int SF, Ncode, sc, i, j, NumOfBits, MaxIterations, MaxNrOfErrors, MinNrOfErrors;
    double Eb;

    //Vectors and matrixes:
    vec EbN0dB, EbN0, N0, ber;
    smat all_codes, spread_codesI, spread_codesQ;
    bvec transmitted_bits, received_bits;
    cvec transmitted_symbols, received_symbols, transmitted_chips, received_chips;

    //Classes:
    Multicode_Spread_2d mc_spread;
    AWGN_Channel channel;
    QPSK qpsk;
    BEREC ber;

    //Initialize the spreading:
    SF = 4; //The spreading factor is 4
    Ncode = 4; //Use all four codes in the multi-code spreader
    der
```

```

all_codes = to_smat(hadamard(SF)); //Calculate the spreading codes

//Set the spreading codes:
spread_codesI.set_size(Ncode, SF, false);
spread_codesQ.set_size(Ncode, SF, false);
for (sc = 0; sc<Ncode; sc++) {
    spread_codesI.set_row(sc, all_codes.get_row(sc));
    spread_codesQ.set_row(sc, all_codes.get_row(sc));
}
mc_spread.set_codes(to_mat(spread_codesI), to_mat(spread_codesQ));

//Specify simulation specific variables:
EbN0dB = linspace(-2,14,17);
EbN0 = pow(10,EbN0dB/10);
Eb = 1.0;
N0 = Eb * pow(EbN0,-1.0);
NumOfBits = 50000;
MaxIterations = 10;
MaxNrOfErrors = 200;
MinNrOfErrors = 5;
berc.set_size(EbN0dB.size(), false);
berc.clear();

//Randomize the random number generators:
RNG_randomize();

for (i=0; i<EbN0dB.length(); i++) {

    cout << endl << "Simulating point nr " << i+1 << endl;
    berc.clear();
    channel.set_noise(N0(i)/2.0);

    for (j=0; j<MaxIterations; j++) {

        transmitted_bits = randb(NumOfBits);
        transmitted_symbols = qpsk.modulate_bits(transmitted_bits);

        //This is where we do the multi-code spreading:
        transmitted_chips = mc_spread.spread(transmitted_symbols);

        received_chips = channel(transmitted_chips);

        //The multi-code despreading:
        //The second argument tells the despreader that the offset is zero chips.
        //This offset is usefull on channels with delay.
        received_symbols = mc_spread.despread(received_chips, 0);

        received_bits = qpsk.demodulate_bits(received_symbols);

        berc.count(transmitted_bits,received_bits);
        ber(i) = berc.get_errorrate();

        cout << "  Iteration " << j+1 << ": ber = " << berc.get_errorrate() << endl;
        if (berc.get_errors()>MaxNrOfErrors) {
            cout << "Breaking on point " << i+1 << " with " << berc.get_errors() << " errors." << endl; break;
        }
    }

    if (berc.get_errors() < MinNrOfErrors) {

```

```
        cout << "Exiting Simulation on point " << i+1 << endl; break;
    }

}

//Print results:
cout << endl;
cout << "EbN0dB = " << EbN0dB << endl;
cout << "ber = " << ber << endl;

//Exit program:
return 0;
}
```

A typical run of this program will look like this:

```
Simulating point nr 1
    Iteration 1: ber = 0.13016
Breaking on point 1 with 6508 errors.

Simulating point nr 2
    Iteration 1: ber = 0.103
Breaking on point 2 with 5150 errors.

Simulating point nr 3
    Iteration 1: ber = 0.07886
Breaking on point 3 with 3943 errors.

Simulating point nr 4
    Iteration 1: ber = 0.05534
Breaking on point 4 with 2767 errors.

Simulating point nr 5
    Iteration 1: ber = 0.03822
Breaking on point 5 with 1911 errors.

Simulating point nr 6
    Iteration 1: ber = 0.02388
Breaking on point 6 with 1194 errors.

Simulating point nr 7
    Iteration 1: ber = 0.01316
Breaking on point 7 with 658 errors.

Simulating point nr 8
    Iteration 1: ber = 0.00586
Breaking on point 8 with 293 errors.

Simulating point nr 9
    Iteration 1: ber = 0.0027
    Iteration 2: ber = 0.00247
Breaking on point 9 with 247 errors.

Simulating point nr 10
    Iteration 1: ber = 0.00094
    Iteration 2: ber = 0.00086
    Iteration 3: ber = 0.00076
    Iteration 4: ber = 0.00077
```

```

    Iteration 5: ber = 0.0008
    Iteration 6: ber = 0.000796667
    Breaking on point 10 with 239 errors.

Simulating point nr 11
    Iteration 1: ber = 0.00016
    Iteration 2: ber = 0.00016
    Iteration 3: ber = 0.000186667
    Iteration 4: ber = 0.000175
    Iteration 5: ber = 0.000172
    Iteration 6: ber = 0.000173333
    Iteration 7: ber = 0.000162857
    Iteration 8: ber = 0.00017
    Iteration 9: ber = 0.000166667
    Iteration 10: ber = 0.000172

Simulating point nr 12
    Iteration 1: ber = 4e-05
    Iteration 2: ber = 2e-05
    Iteration 3: ber = 2.66667e-05
    Iteration 4: ber = 2.5e-05
    Iteration 5: ber = 2e-05
    Iteration 6: ber = 2e-05
    Iteration 7: ber = 3.14286e-05
    Iteration 8: ber = 2.75e-05
    Iteration 9: ber = 2.44444e-05
    Iteration 10: ber = 2.6e-05

Simulating point nr 13
    Iteration 1: ber = 0
    Iteration 2: ber = 0
    Iteration 3: ber = 0
    Iteration 4: ber = 0
    Iteration 5: ber = 0
    Iteration 6: ber = 0
    Iteration 7: ber = 0
    Iteration 8: ber = 2.5e-06
    Iteration 9: ber = 2.22222e-06
    Iteration 10: ber = 2e-06
    Exiting Simulation on point 13

EbN0dB = [-2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]
ber = [0.13016 0.103 0.07886 0.05534 0.03822 0.02388 0.01316 0.00586 0.00247 0.00
       0796667 0.000172 2.6e-05 2e-06 0 0 0 0]

```

Use copy-and-paste to insert the variables `EbN0dB` and `ber` into Matlab and plot the result.

28 Using timers to measure execution time

In this example we are using the `Real_Timer` class to measure the execution time of a simple program. The `Real_Timer` class is included in the `itmisc` library.

```

#include <itpp/itbase.h>

using namespace itpp;

//These lines are needed for use of cout and endl

```

```
using std::cout;
using std::endl;

int main()
{
    //Declare the scalars used:
    long i, sum, N;

    //Declare tt as an instance of the timer class:
    Real_Timer tt;

    //Initiate the variables:
    N = 1000000;
    sum = 0;

    //Start and reset the timer:
    tt.tic();

    //Do some processing
    for (i=0; i<N; i++) {
        sum += i;
    }

    // Print the elapsed time
    tt.toc_print();

    //Print the result of the processing:
    cout << "The sum of all integers from 0 to " << N-1 << " equals " << sum << endl;

    //Exit program:
    return 0;
}
```

When you run this program, the output will look something like this:

```
Elapsed time = 0.000797055 seconds
The sum of all integers from 0 to 999999 equals 1783293664
```

29 Tutorials

29.1 Table of Contents

- [Introduction](#)
 - [Preparing the StdAir Project for Development](#)
- [Build a Predefined BOM Tree](#)
 - [Instantiate the BOM Root Object](#)
 - [Instantiate the \(Airline\) Inventory Object](#)
 - [Link the Inventory Object with the BOM Root](#)
 - [Build Another Airline Inventory](#)

- [Dump The BOM Tree Content](#)
 - [Result of the Tutorial Program](#)
- [Extend the Pre-Defined BOM Tree](#)
 - [Extend an Airline Inventory Object](#)
 - [Build the Specific BOM Objects](#)
 - [Result of the Tutorial Program](#)

29.2 Introduction

This page contains some tutorial examples that will help you getting started using StdAir. Most examples show how to construct some simple business objects, i.e., instances of the so-named Business Object Model (BOM).

29.2.1 Preparing the StdAir Project for Development

The source code for these examples can be found in the `batches` and `test/stdair` directories. They are compiled along with the rest of the StdAir project. See the User Guide ([Users Guide](#)) for more details on how to build the StdAir project.

29.3 Build a Predefined BOM Tree

A few steps:

- [Instantiate the BOM Root Object](#)
- [Instantiate the \(Airline\) Inventory Object](#)
- [Link the Inventory Object with the BOM Root](#)

29.3.1 Instantiate the BOM Root Object

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `stdair::STDAIR_ServiceContext` context object, when the `stdair::STDAIR_Service` is itself instantiated. The corresponding StdAir type (class) is `stdair::BomRoot`.

In the following sample, that object is named `ioBomRoot`, and is given as input/output parameter of the `stdair::CmdBomManager::buildSampleBom()` method:

29.3.2 Instantiate the (Airline) Inventory Object

An airline inventory object can then be instantiated. Let us give it the "BA" airline code (corresponding to [British Airways](#)) as the object key. That is, an object (let us name it `lBAKey`) of type (class) `stdair::InventoryKey` has first to be instantiated.

Thanks to that key, an airline inventory object, i.e. of type (class) `stdair::Inventory`, can be instantiated. Let us name that airline inventory object `lBAInv`.

29.3.3 Link the Inventory Object with the BOM Root

Then, both objects have to be linked: the airline inventory object (`stdair::Inventory`) has to be linked with the root of the BOM tree (`stdair::BomRoot`). That operation is as simple as using the `stdair::FacBomManager::addToListAndMap()` method:

29.3.4 Build Another Airline Inventory

Another airline inventory object, corresponding to the Air France ([Air France](#)) company, is instantiated the same way:

See the corresponding full program (`cmd_bom_manager_cpp`) for more details.

29.3.5 Dump The BOM Tree Content

From the `BomRoot` (of type `stdair::BomRoot`) object instance, the list of airline inventories (of type `stdair::Inventory`) can then be retrieved...

... and browsed:

See the corresponding full program (`bom_display_cpp`) for more details.

29.3.6 Result of the Tutorial Program

When the `stdair.cpp` program is run (with the `-b` option), the output should look like:

See the corresponding full program (`batch_stdair_cpp`) for more details.

29.4 Extend the Pre-Defined BOM Tree

Now that we master how to instantiate the pre-defined StdAir classes, let us see how to extend that BOM.

29.4.1 Extend an Airline Inventory Object

For instance, let us assume that some (IT) provider (e.g., you) would like to have a specific implementation of the `Inventory` object. The corresponding class has just to extend the `stdair::Inventory` class:

The STL containers have to be defined accordingly too:

See the full class definition (`test_archi_inv.hpp`) and implementation (`test_archi_inv.cpp`) for more details.

29.4.2 Build the Specific BOM Objects

The BOM root object (`stdair::BomRoot`) is instantiated the classical way:

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) can be instantiated the same way as a standard `Inventory` (`stdair::Inventory`) would be:

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) is linked to the root of the BOM tree (`stdair::BomRoot`) the same way as the standard `Inventory` (`stdair::Inventory`) would be:

Another specific airline inventory object is instantiated the same way:

From the `BomRoot` (of type `stdair::BomRoot`) object instance, the list of specific airline inventories (of type `stdair::Inventory`) can then be retrieved...

... and browsed:

29.4.3 Result of the Tutorial Program

When this program is run, the output should look like:

See the corresponding full program (StandardAirlineITTestSuite_cpp) for more details.

30 A very simple tutorial about vectors and matrixes

Let's start with a really simple example. Try to compile the following program:

```
#include <itpp/itbase.h>

using namespace itpp;

//These lines are needed for use of cout and endl
using std::cout;
using std::endl;

int main()
{
    //Declare vectors and matrices:
    vec a, b, c;
    mat A, B;

    //Use the function linspace to define a vector:
    a = linspace(1.0,2.0,10);

    //Use a string of values to define a vector:
    b = "0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0";

    //Add two vectors:
    c = a + b;

    //Print results:
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "c = " << c << endl;

    //Use a string to define a matrix:
    A = "1.0 2.0;3.0 4.0";

    //Calculate the inverse of matrix A:
    B = inv(A);

    //Print results:
    cout << "A = " << A << endl;
    cout << "B = " << B << endl;

    //Exit program:
    return 0;
}
```

When you run this program, the output shall look like this

```

a = [1 1.11111 1.22222 1.33333 1.44444 1.55556 1.66667 1.77778 1.88889 2]
b = [0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]
c = [1.1 1.31111 1.52222 1.73333 1.94444 2.15556 2.36667 2.57778 2.78889 3]
A = [[1 2]
      [3 4]]
B = [[-2 1]
      [1.5 -0.5]]

```

If this is what you see, then congratulations! You have managed to compile your first it++ program!

31 Command-Line Test to Demonstrate How To Test the RMOL Project

```

*/
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <cassert>
#include <limits>
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE OptimiseTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/service/Logger.hpp>
// RMOL
#include <rmol/RMOL_Service.hpp>
#include <rmol/config/rmol-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("bomsforforecaster_utfresults.xml");

struct UnitTestConfig {
    UnitTestConfig() {
        boost_utf::unit_test_log.set_stream (utfReportStream);
        boost_utf::unit_test_log.set_format (boost_utf::XML);
        boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
        //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
    }

    ~UnitTestConfig() {
    }
};

namespace RMOL {

    struct BookingClassData {

        // Attributes

```

```

double _bookingCount;
double _fare;
double _sellupFactor;
bool _censorshipFlag;

// Constructor
BookingClassData (const double iBookingCount, const double iFare,
                  const double iSellupFactor, const bool iCensorshipFlag)
    : _bookingCount(iBookingCount), _fare(iFare),
      _sellupFactor(iSellupFactor), _censorshipFlag(iCensorshipFlag) {
}

// Getters
double getFare () const {
    return _fare;
}

bool getCensorshipFlag () const {
    return _censorshipFlag;
}

// Display
std::string toString() const {
    std::ostringstream oStr;
    oStr << std::endl
        << "[Booking class data information]" << std::endl
        << "Booking counter: " << _bookingCount << std::endl
        << "Fare: " << _fare << std::endl
        << "Sell-up Factor: " << _sellupFactor << std::endl
        << "censorshipFlag: " << _censorshipFlag << std::endl;
    return oStr.str();
}

};

struct BookingClassDataSet {

    typedef std::vector<BookingClassData*> BookingClassDataList_T;

    // Attributes
    int _numberOfClass;
    double _minimumFare;
    bool _censorshipFlag; // true if any of the classes is censored
    BookingClassDataList_T _bookingClassDataList;

    // Constructor
    BookingClassDataSet ()
        : _numberOfClass(0), _minimumFare(0),
          _censorshipFlag(false) {
    }

    // Add BookingClassData
    void addBookingClassData (BookingClassData& ioBookingClassData) {
        _bookingClassDataList.push_back (&ioBookingClassData);
    }

    // Getters
    unsigned int getNumberOfClass () const {
        return _bookingClassDataList.size();
    }

    double getMinimumFare () const {

```

```

        return _minimumFare;
    }

    bool getCensorshipFlag () const {
        return _censorshipFlag;
    }

    // Setters
    void setMinimumFare (const double iMinFare) {
        _minimumFare = iMinFare;
    }

    void setCensorshipFlag (const bool iCensorshipFlag) {
        _censorshipFlag = iCensorshipFlag;
    }

    // compute minimum fare
    void updateMinimumFare() {
        double minFare = std::numeric_limits<double>::max();
        BookingClassDataList_T::iterator itBookingClassDataList;
        for (itBookingClassDataList = _bookingClassDataList.begin();
            itBookingClassDataList != _bookingClassDataList.end();
            ++itBookingClassDataList) {
            BookingClassData* lBookingClassData = *itBookingClassDataList;
            assert (lBookingClassData != NULL);

            const double lFare = lBookingClassData->getFare();
            if (lFare < minFare) {
                minFare = lFare;
            }
        }
        //
        setMinimumFare(minFare);
    }

    // compute censorship flag for the data set
    void updateCensorshipFlag () {
        bool censorshipFlag = false;
        BookingClassDataList_T::iterator itBookingClassDataList;
        for (itBookingClassDataList = _bookingClassDataList.begin();
            itBookingClassDataList != _bookingClassDataList.end();
            ++itBookingClassDataList) {
            BookingClassData* lBookingClassData = *itBookingClassDataList;
            assert (lBookingClassData != NULL);

            const bool lCensorshipFlagOfAClass =
                lBookingClassData->getCensorshipFlag();
            if (lCensorshipFlagOfAClass) {
                censorshipFlag = true;
                break;
            }
        }
        //
        setCensorshipFlag(censorshipFlag);
    }

    // Display
    std::string toString() const {
        std::ostringstream oStr;
        oStr << std::endl
            << "[Booking class data set information]" << std::endl
            << "Number of classes: " << _numberOfClass << std::endl

```

```

        << "Minimum fare: " << _minimumFare << std::endl
        << "The data of the class set are sensed: " << _censorshipFlag
        << std::endl;
        return ostr.str();
    }

};

// /**----- BOM : Q-Forecaster ----- */
// struct QForecaster {

//     // Function focused BOM

//     // 1. calculate sell up probability for Q-eq

//     // 2. calculate Q-Equivalent Booking
//     double calculateQEqBooking (BookingClassDataSet& iBookingClassDataSet) {
//         double lQEqBooking = 0.0;
//         double lMinFare = iBookingClassDataSet.getMinimumFare();

//         return lQEqBooking;
//     }

//     /* Calculate Q-equivalent demand
//        [<- performed by unconstrainer if necessary (Using ExpMax BOM)]
//     */

//     // 3. Partition to each class

//     //

// };

}

// ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestFixture);

BOOST_AUTO_TEST_SUITE (master_test_suite)

BOOST_AUTO_TEST_CASE (rmol_forecaster) {

    // Output log File
    std::string lLogFilename ("bomsforforecaster.log");
    std::ofstream logOutputFile;

    // Open and clean the log outputfile
    logOutputFile.open (lLogFilename.c_str());
    logOutputFile.clear();

    // Initialise the RMOL service
    const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);

    // Initialise the RMOL service
    RMOL::RMOL_Service rmolService (lLogParams);

    // Build a sample BOM tree

```

```

rmolService.buildSampleBom();

// Register BCDataSet
RMOL::BookingClassDataSet lBookingClassDataSet;

// Register BookingClassData
RMOL::BookingClassData QClassData (10, 100, 1, false);
RMOL::BookingClassData MClassData (5, 150, 0.8, true);
RMOL::BookingClassData BClassData (0, 200, 0.6, false);
RMOL::BookingClassData YClassData (0, 300, 0.3, false);

// Display
STDAIR_LOG_DEBUG (QClassData.toString());
STDAIR_LOG_DEBUG (MClassData.toString());
STDAIR_LOG_DEBUG (BClassData.toString());
STDAIR_LOG_DEBUG (YClassData.toString());

// Add BookingClassData into the BCDataSet
lBookingClassDataSet.addBookingClassData (QClassData);
lBookingClassDataSet.addBookingClassData (MClassData);
lBookingClassDataSet.addBookingClassData (BClassData);
lBookingClassDataSet.addBookingClassData (YClassData);

// DEBUG
STDAIR_LOG_DEBUG (lBookingClassDataSet.toString());

// Number of classes
const unsigned int lNoOfClass = lBookingClassDataSet.getNumberOfClass();

// DEBUG
STDAIR_LOG_DEBUG ("Number of Classes: " << lNoOfClass);

// Minimum fare
BOOST_CHECK_NO_THROW (lBookingClassDataSet.updateMinimumFare());
const double lMinFare = lBookingClassDataSet.getMinimumFare();

// DEBUG
STDAIR_LOG_DEBUG ("Minimum fare: " << lMinFare);

// Censorship flag
BOOST_CHECK_NO_THROW (lBookingClassDataSet.updateCensorshipFlag());
const bool lCensorshipFlag = lBookingClassDataSet.getCensorshipFlag();

// DEBUG
STDAIR_LOG_DEBUG ("Censorship Flag: " << lCensorshipFlag);

// Close the log output file
logOutputFile.close();
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!

```

32 Command-Line Test to Demonstrate How To Test the RMOL Project

```

*/
// //////////////////////////////////////
// Import section

```

```

// //////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
#include <vector>
#include <cmath>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE ForecasterTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
// RMOL
#include <rmol/RMOL_Service.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("ForecasterTestSuite_utfresults.xml");

struct UnitTestConfig {
    UnitTestConfig() {
        boost_utf::unit_test_log.set_stream (utfReportStream);
        boost_utf::unit_test_log.set_format (boost_utf::XML);
        boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
        //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
    }

    ~UnitTestConfig() {
    }
};

// ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

BOOST_AUTO_TEST_SUITE (master_test_suite)

BOOST_AUTO_TEST_CASE (rmol_forecaster_q_forecasting) {
    const bool lTestFlag = true; //testForecasterHelper(0);
    BOOST_CHECK_EQUAL (lTestFlag, true);
    BOOST_CHECK_MESSAGE (lTestFlag == true,
        "The test has failed. Please see the log file for "
        "<< \"more details\"");
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!

```

33 Command-Line Test to Demonstrate How To Test the RMOL Project

```

*/
// ////////////////////////////////////////
// Import section
// ////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE OptimiseTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
// RMOL
#include <rmol/RMOL_Service.hpp>
#include <rmol/config/rmol-paths.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("OptimiseTestSuite_utfresults.xml");

struct UnitTestConfig {
    UnitTestConfig() {
        boost_utf::unit_test_log.set_stream (utfReportStream);
        boost_utf::unit_test_log.set_format (boost_utf::XML);
        boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
        //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tests);
    }

    ~UnitTestConfig() {
    }
};

// ////////////////////////////////////////
int testOptimiseHelper (const unsigned short optimisationMethodFlag,
                        const bool isBuiltin) {

    // Return value
    int oExpectedBookingLimit = 0;

    // Output log File
    std::ostringstream oStr;
    oStr << "OptimiseTestSuite_" << optimisationMethodFlag << ".log";
    const stdair::Filename_T lLogFilename (oStr.str());

    // Number of random draws to be generated (best if greater than 100)
    const int K = 100000;

    // Methods of optimisation (0 = Monte-Carlo, 1 = Dynamic Programming,
    // 2 = EMSR, 3 = EMSR-a, 4 = EMSR-b, 5 = EMSR-a with sellup prob.)
    const unsigned short METHOD_FLAG = optimisationMethodFlag;

```



```
// Cabin Capacity (it must be greater then 100 here)
const double cabinCapacity = 100.0;

// Set the log parameters
std::ofstream logOutputFile;
// Open and clean the log outputfile
logOutputFile.open (lLogFilename.c_str());
logOutputFile.clear();

// Initialise the RMOL service
const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
RMOL::RMOL_Service rmolService (lLogParams);

// Check whether or not a (CSV) input file should be read
if (isBuiltin == true) {

    // Build the default sample BOM tree and build a dummy BOM tree.
    rmolService.buildSampleBom();

} else {

    // Parse the optimisation data and build a dummy BOM tree
    const stdair::Filename_T lRMInputFileName (STDAIR_SAMPLE_DIR "/rm02.csv");
    rmolService.parseAndLoad (cabinCapacity, lRMInputFileName);
}

switch (METHOD_FLAG) {
case 0: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Optimisation by Monte-Carlo (MC)");

    // Calculate the optimal protections by the Monte Carlo
    // Integration approach
    rmolService.optimalOptimisationByMCIntegration (K);
    break;
}

case 1: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Optimisation by Dynamic Programming (DP)");

    // Calculate the optimal protections by DP.
    rmolService.optimalOptimisationByDP ();
    break;
}

case 2: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Calculate the Bid-Price Vectors (BPV) by EMSR");

    // Calculate the Bid-Price Vector by EMSR
    rmolService.heuristicOptimisationByEmsr ();
    break;
}

case 3: {
    // DEBUG
    STDAIR_LOG_DEBUG ("Calculate the Authorisation Levels (AUs) by EMSRa");

    // Calculate the protections by EMSR-a
    // Test the EMSR-a algorithm implementation
    rmolService.heuristicOptimisationByEmsrA ();
}
```

```

        // Return a cumulated booking limit value to test
        // oExpectedBookingLimit = static_cast<int> (lBookingLimitVector.at(2));
        break;
    }

    case 4: {
        // DEBUG
        STDAIR_LOG_DEBUG ("Calculate the Authorisation Levels (AUs) by EMSRb");

        // Calculate the protections by EMSR-b
        rmolService.heuristicOptimisationByEmsrB ();
        break;
    }

    default: rmolService.optimalOptimisationByMCIntegration (K);
}

// Close the log file
logOutputFile.close();

return oExpectedBookingLimit;
}

// ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestFixture);

// ////////////////////////////////////////
// Tests are based on the following input values
// price; mean; standard deviation;
// 1050; 17.3; 5.8;
// 567; 45.1; 15.0;
// 534; 39.6; 13.2;
// 520; 34.0; 11.3;
// ////////////////////////////////////////

BOOST_AUTO_TEST_SUITE (master_test_suite)

BOOST_AUTO_TEST_CASE (rmol_optimisation_monte_carlo) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = false;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(0, isBuiltin));
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_dynamic_programming) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = false;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(1, isBuiltin));
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_bpv) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = false;

```

```
    BOOST_CHECK_NO_THROW (testOptimiseHelper(2, isBuiltin));
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_a) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = false;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(3, isBuiltin));
    // const int lBookingLimit = testOptimiseHelper(3);
    // const int lExpectedBookingLimit = 61;
    // BOOST_CHECK_EQUAL (lBookingLimit, lExpectedBookingLimit);
    // BOOST_CHECK_MESSAGE (lBookingLimit == lExpectedBookingLimit,
    //                       "The booking limit is " << lBookingLimit
    //                       << ", but it is expected to be "
    //                       << lExpectedBookingLimit);
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_b) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = false;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(4, isBuiltin));
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_monte_carlo_built_in) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = true;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(5, isBuiltin));
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_dynamic_programming_built_in) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = true;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(6, isBuiltin));
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_bpv_built_in) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = true;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(7, isBuiltin));
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_a_built_in) {

    // State whether the BOM tree should be built-in or parsed from an input file
    const bool isBuiltin = true;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(8, isBuiltin));
}

BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_b_built_in) {

    // State whether the BOM tree should be built-in or parsed from an input file
```

```

    const bool isBuiltin = true;

    BOOST_CHECK_NO_THROW (testOptimiseHelper(9, isBuiltin));
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!

```

34 Command-Line Test to Demonstrate How To Test the RMOL Project

```

*/
// //////////////////////////////////////
// Import section
// //////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE UnconstrainerTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
// RMOL
#include <rmol/RMOL_Service.hpp>

namespace boost_utf = boost::unit_test;

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("UnconstrainerTestSuite_utfresults.xml");

struct UnitTestConfig {
    UnitTestConfig() {
        boost_utf::unit_test_log.set_stream (utfReportStream);
        boost_utf::unit_test_log.set_format (boost_utf::XML);
        boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
        //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_test);
    }

    ~UnitTestConfig() {
    }
};

// ////////////////////////////////////// Main: Unit Test Suite //////////////////////////////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

BOOST_AUTO_TEST_SUITE (master_test_suite)

```

```
BOOST_AUTO_TEST_CASE (rmol_unconstraining_em) {  
    const bool lTestFlag = true; // testUnconstrainerHelper(0);  
    BOOST_CHECK_EQUAL (lTestFlag, true);  
    BOOST_CHECK_MESSAGE (lTestFlag == true,  
        "The test has failed. Please see the log file for "  
        << "more details");  
}  
  
// End the test suite  
BOOST_AUTO_TEST_SUITE_END()  
  
/*!
```

35 Directory Hierarchy

35.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

doc	95
tutorial	96
src	96
rmol	95
basic	93
batches	93
bom	94
old	95
command	94
config	94
factory	95
service	96
test	96
rmol	95

36 Namespace Index

36.1 Namespace List

Here is a list of all namespaces with brief descriptions:

RMOL	97
stdair	101

37 Class Index

37.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

```
std::basic_fstream< char >
std::basic_fstream< wchar_t >
std::basic_ifstream< char >
std::basic_ifstream< wchar_t >
std::basic_ios< char >
std::basic_ios< wchar_t >
std::basic_iostream< char >
std::basic_iostream< wchar_t >
std::basic_istream< char >
std::basic_istream< wchar_t >
std::basic_istreamstream< char >
std::basic_istreamstream< wchar_t >
std::basic_ofstream< char >
std::basic_ofstream< wchar_t >
std::basic_ostream< char >
std::basic_ostream< wchar_t >
std::basic_ostreamstream< char >
std::basic_ostreamstream< wchar_t >
std::basic_string< char >
std::basic_string< wchar_t >
std::basic_stringstream< char >
std::basic_stringstream< wchar_t >
```

CmdAbstract	101
RMOL::InventoryParser	122
RMOL::DefaultDCPList	101
RMOL::DefaultMap	102
RMOL::DemandGeneratorList	102
RMOL::Detruncator	104
RMOL::DPOptimiser	106

RMOL::EMDetruncator	106
RMOL::Emsr	107
RMOL::EmsrUtils	108
FacServiceAbstract	111
RMOL::FacRmolServiceContext	109
RMOL::Forecaster	111
RMOL::GuillotineBlockHelper	114
RMOL::MCOptimiser	123
RMOL::Optimiser	125
RMOL::RMOL_Service	129
RootException	138
RMOL::ForecastException	113
RMOL::OptimisationException	124
RMOL::OverbookingException	128
RMOL::UnconstrainingException	140
ServiceAbstract	138
RMOL::RMOL_ServiceContext	137
StructAbstract	139
RMOL::HistoricalBooking	115
RMOL::HistoricalBookingHolder	118
TestFixture	139
ForecasterTestSuite	112
OptimiseTestSuite	127
UnconstrainerTestSuite	139
RMOL::Utilities	141

38 Class Index

38.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

CmdAbstract	101
RMOL::DefaultDCPList	101
RMOL::DefaultMap	102
RMOL::DemandGeneratorList	102
RMOL::Detruncator	104
RMOL::DPOptimiser	106
RMOL::EMDetruncator	106
RMOL::Emsr	107
RMOL::EmsrUtils	108
RMOL::FacRmolServiceContext (Factory for the service context)	109
FacServiceAbstract	111
RMOL::Forecaster	111
ForecasterTestSuite	112
RMOL::ForecastException (Forecast-related exception)	113
RMOL::GuillotineBlockHelper	114
RMOL::HistoricalBooking (Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag)	115
RMOL::HistoricalBookingHolder	118
RMOL::InventoryParser (Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory)	122
RMOL::MCOptimiser	123
RMOL::OptimisationException (Optimisation-related exception)	124
RMOL::Optimiser	125
OptimiseTestSuite	127
RMOL::OverbookingException (Overbooking-related exception)	128
RMOL::RMOL_Service (Interface for the RMOL Services)	129

RMOL::RMOL_ServiceContext (Inner class holding the context for the RMOL Service object)	137
RootException	138
ServiceAbstract	138
StructAbstract	139
TestFixture	139
UnconstrainerTestSuite	139
RMOL::UnconstrainingException (Unconstraining-related exception)	140
RMOL::Utilities	141

39 File Index

39.1 File List

Here is a list of all files with brief descriptions:

doc/tutorial/src/bpsk.cpp	144
doc/tutorial/src/convcode.cpp	145
doc/tutorial/src/interleaver.cpp	147
doc/tutorial/src/ldpc_bersim_awgn.cpp	148
doc/tutorial/src/ldpc_gen_codes.cpp	150
doc/tutorial/src/mimoconv.cpp	152
doc/tutorial/src/mog.cpp	157
doc/tutorial/src/qpsk_simulation.cpp	160
doc/tutorial/src/rayleigh.cpp	162
doc/tutorial/src/read_it_file.cpp	163
doc/tutorial/src/reedsolomon.cpp	164
doc/tutorial/src/spread.cpp	165
doc/tutorial/src/timer.cpp	167
doc/tutorial/src/vector_and_matrix.cpp	168

doc/tutorial/src/write_it_file.cpp	169
rmol/RMOL_Service.hpp	264
rmol/RMOL_Types.hpp	268
rmol/basic/BasConst.cpp	170
rmol/basic/BasConst_Curves.hpp	172
rmol/basic/BasConst_General.hpp	173
rmol/basic/BasConst_RMOL_Service.hpp	173
rmol/batches/rmol.cpp	176
rmol/bom/BucketHolderTypes.hpp	181
rmol/bom/DistributionParameterList.hpp	182
rmol/bom/DPOptimiser.cpp	183
rmol/bom/DPOptimiser.hpp	186
rmol/bom/EMDetruncator.cpp	187
rmol/bom/EMDetruncator.hpp	189
rmol/bom/Emsr.cpp	190
rmol/bom/Emsr.hpp	193
rmol/bom/EmsrUtils.cpp	194
rmol/bom/EmsrUtils.hpp	196
rmol/bom/GuillotineBlockHelper.cpp	197
rmol/bom/GuillotineBlockHelper.hpp	198
rmol/bom/HistoricalBooking.cpp	199
rmol/bom/HistoricalBooking.hpp	201
rmol/bom/HistoricalBookingHolder.cpp	202
rmol/bom/HistoricalBookingHolder.hpp	207
rmol/bom/MCOptimiser.cpp	209
rmol/bom/MCOptimiser.hpp	214
rmol/bom/Utilities.cpp	218

rmol/bom/Utilities.hpp	220
rmol/bom/old/DemandGeneratorList.cpp	215
rmol/bom/old/DemandGeneratorList.hpp	216
rmol/command/Detruncator.cpp	221
rmol/command/Detruncator.hpp	231
rmol/command/Forecaster.cpp	234
rmol/command/Forecaster.hpp	248
rmol/command/InventoryParser.cpp	250
rmol/command/InventoryParser.hpp	253
rmol/command/Optimiser.cpp	254
rmol/command/Optimiser.hpp	258
rmol/config/rmol-paths.hpp	260
rmol/factory/FacRmolServiceContext.cpp	261
rmol/factory/FacRmolServiceContext.hpp	262
rmol/service/RMOL_Service.cpp	270
rmol/service/RMOL_ServiceContext.cpp	302
rmol/service/RMOL_ServiceContext.hpp	304
test/rmol/bomsforforecaster.cpp	305
test/rmol/ForecasterTestSuite.cpp	310
test/rmol/ForecasterTestSuite.hpp	311
test/rmol/OptimiseTestSuite.cpp	312
test/rmol/OptimiseTestSuite.hpp	316
test/rmol/UnconstrainerTestSuite.cpp	317
test/rmol/UnconstrainerTestSuite.hpp	318

40 Directory Documentation

40.1 rmol/basic/ Directory Reference

Files

- file [BasConst.cpp](#)
- file [BasConst_Curves.hpp](#)
- file [BasConst_General.hpp](#)
- file [BasConst_RMOL_Service.hpp](#)

40.2 rmol/batches/ Directory Reference

Files

- file [rmol.cpp](#)

40.3 rmol/bom/ Directory Reference

Directories

- directory [old](#)

Files

- file [BucketHolderTypes.hpp](#)
- file [DistributionParameterList.hpp](#)
- file [DPOptimiser.cpp](#)
- file [DPOptimiser.hpp](#)
- file [EMDetruncator.cpp](#)
- file [EMDetruncator.hpp](#)
- file [Emsr.cpp](#)
- file [Emsr.hpp](#)
- file [EmsrUtils.cpp](#)
- file [EmsrUtils.hpp](#)
- file [GuillotineBlockHelper.cpp](#)
- file [GuillotineBlockHelper.hpp](#)
- file [HistoricalBooking.cpp](#)
- file [HistoricalBooking.hpp](#)
- file [HistoricalBookingHolder.cpp](#)
- file [HistoricalBookingHolder.hpp](#)
- file [MCOptimiser.cpp](#)
- file [MCOptimiser.hpp](#)
- file [Utilities.cpp](#)
- file [Utilities.hpp](#)

40.4 rmol/command/ Directory Reference

Files

- file [Detruncator.cpp](#)
- file [Detruncator.hpp](#)
- file [Forecaster.cpp](#)
- file [Forecaster.hpp](#)
- file [InventoryParser.cpp](#)
- file [InventoryParser.hpp](#)
- file [Optimiser.cpp](#)
- file [Optimiser.hpp](#)

40.5 rmol/config/ Directory Reference

Files

- file [rmol-paths.hpp](#)

40.6 doc/ Directory Reference

Directories

- directory [tutorial](#)

40.7 rmol/factory/ Directory Reference

Files

- file [FacRmolServiceContext.cpp](#)
- file [FacRmolServiceContext.hpp](#)

40.8 rmol/bom/old/ Directory Reference

Files

- file [DemandGeneratorList.cpp](#)
- file [DemandGeneratorList.hpp](#)

40.9 test/rmol/ Directory Reference

Files

- file [bomsforforecaster.cpp](#)

- file [ForecasterTestSuite.cpp](#)
- file [ForecasterTestSuite.hpp](#)
- file [OptimiseTestSuite.cpp](#)
- file [OptimiseTestSuite.hpp](#)
- file [UnconstrainerTestSuite.cpp](#)
- file [UnconstrainerTestSuite.hpp](#)

40.10 rmol/ Directory Reference

Directories

- directory [basic](#)
- directory [batches](#)
- directory [bom](#)
- directory [command](#)
- directory [config](#)
- directory [factory](#)
- directory [service](#)

Files

- file [RMOL_Service.hpp](#)
- file [RMOL_Types.hpp](#)

40.11 rmol/service/ Directory Reference

Files

- file [RMOL_Service.cpp](#)
- file [RMOL_ServiceContext.cpp](#)
- file [RMOL_ServiceContext.hpp](#)

40.12 doc/tutorial/src/ Directory Reference

Files

- file [bpsk.cpp](#)
- file [convcode.cpp](#)
- file [interleaver.cpp](#)
- file [ldpc_bersim_awgn.cpp](#)
- file [ldpc_gen_codes.cpp](#)
- file [mimoconv.cpp](#)
- file [mog.cpp](#)
- file [qpsk_simulation.cpp](#)
- file [rayleigh.cpp](#)

- file [read_it_file.cpp](#)
- file [reedsolomon.cpp](#)
- file [spread.cpp](#)
- file [timer.cpp](#)
- file [vector_and_matrix.cpp](#)
- file [write_it_file.cpp](#)

40.13 test/ Directory Reference

Directories

- directory [rmol](#)

40.14 doc/tutorial/ Directory Reference

Directories

- directory [src](#)

41 Namespace Documentation

41.1 RMOL Namespace Reference

Classes

- struct [DefaultMap](#)
- struct [DefaultDCPList](#)
- class [DPOptimiser](#)
- class [EMDetruncator](#)
- class [Emsr](#)
- class [EmsrUtils](#)
- class [GuillotineBlockHelper](#)
- struct [HistoricalBooking](#)

Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag.

- struct [HistoricalBookingHolder](#)
- class [MCOptimiser](#)
- class [DemandGeneratorList](#)
- class [Utilities](#)
- class [Detruncator](#)
- class [Forecaster](#)
- class [InventoryParser](#)

Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory.

- class [Optimiser](#)
- class [FacRmolServiceContext](#)
Factory for the service context.
- class [RMOL_Service](#)
Interface for the [RMOL](#) Services.
- class [OverbookingException](#)
Overbooking-related exception.
- class [UnconstrainingException](#)
Unconstraining-related exception.
- class [ForecastException](#)
Forecast-related exception.
- class [OptimisationException](#)
Optimisation-related exception.
- class [RMOL_ServiceContext](#)
Inner class holding the context for the [RMOL](#) Service object.

Typedefs

- typedef std::list< BucketHolder * > [BucketHolderList_T](#)
- typedef std::list< FldDistributionParameters > [DistributionParameterList_T](#)
- typedef std::vector< [HistoricalBooking](#) > [HistoricalBookingVector_T](#)
- typedef boost::shared_ptr< [RMOL_Service](#) > [RMOL_ServicePtr_T](#)
- typedef std::vector< stdair::NbOfRequests_T > [UnconstrainedDemandVector_T](#)
- typedef std::vector< stdair::NbOfBookings_T > [BookingVector_T](#)
- typedef std::vector< stdair::Flag_T > [FlagVector_T](#)
- typedef std::map< stdair::BookingClass *, [UnconstrainedDemandVector_T](#) > [BookingClassUnconstrainedDemandMap_T](#)
- typedef std::map< stdair::BookingClass *, stdair::NbOfRequests_T > [BookingClassUnconstrainedDemandMap_T](#)
- typedef std::map< const stdair::DTD_T, double > [FRAT5Curve_T](#)

Variables

- const stdair::AirlineCode_T [DEFAULT_RMOL_SERVICE_AIRLINE_CODE](#) = "BA"
- const double [DEFAULT_RMOL_SERVICE_CAPACITY](#) = 1.0
- const int [DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION](#) = 100000
- const int [DEFAULT_PRECISION](#) = 10
- const double [DEFAULT_EPSILON](#) = 0.0001
- const double [DEFAULT_STOPPING_CRITERION](#) = 0.01
- const double [DEFAULT_INITIALIZER_DOUBLE_NEGATIVE](#) = -10.0
- const [FRAT5Curve_T](#) [DEFAULT_CUMULATIVE_FRAT5_CURVE](#)
- const stdair::DCPList_T [DEFAULT_DCP_LIST](#) = DefaultDCPList::init()

41.1.1 Typedef Documentation

41.1.1.1 `typedef std::list<BucketHolder*> RMOL::BucketHolderList_T`

Define a vector (ordered list) of N bucket/classe holders.

Definition at line 16 of file [BucketHolderTypes.hpp](#).

41.1.1.2 `typedef std::list<FldDistributionParameters> RMOL::DistributionParameterList_T`

Define the set of parameters, each of one wrapping a pair of distribution parameters (i.e., mean and standard deviation).

Definition at line 16 of file [DistributionParameterList.hpp](#).

41.1.1.3 `typedef std::vector<HistoricalBooking> RMOL::HistoricalBookingVector_T`

Define a vector (ordered list) of N HistoricalBookings.

Definition at line 16 of file [HistoricalBookingHolder.hpp](#).

41.1.1.4 `typedef boost::shared_ptr<RMOL_Service> RMOL::RMOL_ServicePtr_T`

Pointer on the [RMOL](#) Service handler.

Definition at line 73 of file [RMOL_Types.hpp](#).

41.1.1.5 `typedef std::vector<stdair::NbOfRequests_T> RMOL::UnconstrainedDemandVector_T`

Define the vector of historical unconstrained demand.

Definition at line 76 of file [RMOL_Types.hpp](#).

41.1.1.6 `typedef std::vector<stdair::NbOfBookings_T> RMOL::BookingVector_T`

Define the vector of historical bookings.

Definition at line 79 of file [RMOL_Types.hpp](#).

41.1.1.7 `typedef std::vector<stdair::Flag_T> RMOL::FlagVector_T`

Define the vector of censorship flags.

Definition at line 82 of file [RMOL_Types.hpp](#).

41.1.1.8 `typedef std::map<stdair::BookingClass*, UnconstrainedDemandVector_T> RMOL::BookingClassUnconstrainedDemandVectorMap_T`

Define the map between the booking class and it's corresponding unconstrained demand vector.

Definition at line 86 of file [RMOL_Types.hpp](#).

41.1.1.9 `typedef std::map<stdair::BookingClass*, stdair::NbOfRequests_T>
RMOL::BookingClassUnconstrainedDemandMap_T`

Define the map between the booking class and its corresponding unconstrained demand.

Definition at line 90 of file [RMOL_Types.hpp](#).

41.1.1.10 `typedef std::map<const stdair::DTD_T, double> RMOL::FRAT5Curve_T`

Define the FRAT5 curve.

Definition at line 93 of file [RMOL_Types.hpp](#).

41.1.2 Variable Documentation

41.1.2.1 `const stdair::AirlineCode_T RMOL::DEFAULT_RMOL_SERVICE_AIRLINE_CODE = "BA"`

Default airline code for the [RMOL_Service](#).

Definition at line 11 of file [BasConst.cpp](#).

41.1.2.2 `const double RMOL::DEFAULT_RMOL_SERVICE_CAPACITY = 1.0`

Default capacity for the [RMOL_Service](#).

Definition at line 14 of file [BasConst.cpp](#).

41.1.2.3 `const int RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 100000`

Default value for the number of draws within the Monte-Carlo Integration algorithm.

Definition at line 18 of file [BasConst.cpp](#).

41.1.2.4 `const int RMOL::DEFAULT_PRECISION = 10`

Default value for the precision of the integral computation in the Dynamic Programming algorithm (100 means that the precision will be 0.01).

Default value for the precision of the integral computation in the Dynamic Programming algorithm.

Definition at line 23 of file [BasConst.cpp](#).

41.1.2.5 `const double RMOL::DEFAULT_EPSILON = 0.0001`

Default epsilon value to qualify a denominator

Definition at line 26 of file [BasConst.cpp](#).

41.1.2.6 `const double RMOL::DEFAULT_STOPPING_CRITERION = 0.01`

Default stopping value for an iterative algorithm.

Definition at line 29 of file [BasConst.cpp](#).

41.1.2.7 `const double RMOL::DEFAULT_INITIALIZER_DOUBLE_NEGATIVE = -10.0`

Default negative value used to initialize a double variable.

Definition at line 32 of file [BasConst.cpp](#).

41.1.2.8 `const FRAT5Curve_T RMOL::DEFAULT_CUMULATIVE_FRAT5_CURVE`

Initial value:

```
DefaultMap::createCumulativeFRAT5Curve()
```

Default cumulative[for the remaining period] FRAT5 curve for forecasting and optimisation.

Default cumulative (for the remaining period) FRAT5 curve for forecasting and optimisation.

Definition at line 36 of file [BasConst.cpp](#).

41.1.2.9 `const stdair::DCPList_T RMOL::DEFAULT_DCP_LIST = DefaultDCPList::init()`

Default data collection point list.

Definition at line 69 of file [BasConst.cpp](#).

Referenced by [RMOL::Utilities::buildRemainingDCPList\(\)](#), [RMOL::Utilities::buildRemainingDCPList2\(\)](#), [RMOL::RMOL_Service::forecastOnD\(\)](#), [RMOL::RMOL_Service::optimiseOnD\(\)](#), [RMOL::RMOL_Service::optimiseOnDUsingAdvancedRMCooperation\(\)](#), [RMOL::RMOL_Service::optimiseOnDUsingRMCooperation\(\)](#), [RMOL::RMOL_Service::projectAggregatedDemandOnLegCabins\(\)](#), [RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingDYP\(\)](#), [RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingYP\(\)](#), [RMOL::RMOL_Service::resetDemandInformation\(\)](#), and [RMOL::RMOL_Service::updateBidPrice\(\)](#).

41.2 stdair Namespace Reference

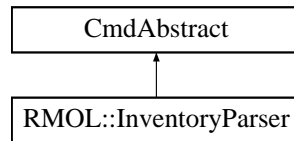
41.2.1 Detailed Description

Forward declarations.

42 Class Documentation

42.1 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract:



The documentation for this class was generated from the following file:

- [rmol/command/InventoryParser.hpp](#)

42.2 RMOL::DefaultDCPList Struct Reference

```
#include <rmol/basic/BasConst_General.hpp>
```

Static Public Member Functions

- static `stdair::DCPList_T` [init](#) ()

42.2.1 Detailed Description

Definition at line 31 of file [BasConst_General.hpp](#).

42.2.2 Member Function Documentation

42.2.2.1 `stdair::DCPList_T RMOL::DefaultDCPList::init ()` [static]

Definition at line 70 of file [BasConst.cpp](#).

The documentation for this struct was generated from the following files:

- [rmol/basic/BasConst_General.hpp](#)
- [rmol/basic/BasConst.cpp](#)

42.3 RMOL::DefaultMap Struct Reference

```
#include <rmol/basic/BasConst_Curves.hpp>
```

Static Public Member Functions

- static `FRAT5Curve_T` [createCumulativeFRAT5Curve](#) ()

42.3.1 Detailed Description

Default PoS probability mass.

Definition at line 17 of file [BasConst_Curves.hpp](#).

42.3.2 Member Function Documentation

42.3.2.1 FRAT5Curve_T RMOL::DefaultMap::createCumulativeFRAT5Curve ()
[static]

Definition at line 38 of file [BasConst.cpp](#).

The documentation for this struct was generated from the following files:

- [rmol/basic/BasConst_Curves.hpp](#)
- [rmol/basic/BasConst.cpp](#)

42.4 RMOL::DemandGeneratorList Class Reference

```
#include <rmol/bom/old/DemandGeneratorList.hpp>
```

Public Member Functions

- [DemandGeneratorList](#) ()
- [DemandGeneratorList](#) (const [DemandGeneratorList](#) &)
- [DemandGeneratorList](#) (const [DistributionParameterList_T](#) &)
- virtual [~DemandGeneratorList](#) ()
- void [generateVariateList](#) ([VariateList_T](#) &) const

Protected Types

- typedef std::list< Gaussian > [DemandGeneratorList_T](#)

42.4.1 Detailed Description

Wrapper around a set of Gaussian Random Generators.

Definition at line 17 of file [DemandGeneratorList.hpp](#).

42.4.2 Member Typedef Documentation

42.4.2.1 typedef std::list<Gaussian> RMOL::DemandGeneratorList::DemandGeneratorList_
T [protected]

Define a (ordered) set of Gaussian Random Generators.

Definition at line 20 of file [DemandGeneratorList.hpp](#).

42.4.3 Constructor & Destructor Documentation

42.4.3.1 RMOL::DemandGeneratorList::DemandGeneratorList ()

Constructors.

Definition at line 10 of file [DemandGeneratorList.cpp](#).

42.4.3.2 RMOL::DemandGeneratorList::DemandGeneratorList (const DemandGeneratorList & iDemandGeneratorList)

Definition at line 17 of file [DemandGeneratorList.cpp](#).

42.4.3.3 RMOL::DemandGeneratorList::DemandGeneratorList (const DistributionParameterList_T & iDistributionParameterList)

List of distribution parameters (mean, standard deviation).

Definition at line 25 of file [DemandGeneratorList.cpp](#).

42.4.3.4 RMOL::DemandGeneratorList::~~DemandGeneratorList () [virtual]

Destructors.

Definition at line 30 of file [DemandGeneratorList.cpp](#).

42.4.4 Member Function Documentation

42.4.4.1 void RMOL::DemandGeneratorList::generateVariateList (VariateList_T & ioVariateList) const

Definition at line 50 of file [DemandGeneratorList.cpp](#).

The documentation for this class was generated from the following files:

- [rmol/bom/old/DemandGeneratorList.hpp](#)
- [rmol/bom/old/DemandGeneratorList.cpp](#)

42.5 RMOL::Detruncator Class Reference

```
#include <rmol/command/Detruncator.hpp>
```

Static Public Member Functions

- static void [unconstrainUsingAdditivePickUp](#) (const stdair::SegmentCabin &, [BookingClassUnconstrainedDemandVector_T](#) &, [UnconstrainedDemandVector_T](#) &, const stdair::DCP_T &, const stdair::DCP_T &, const stdair::Date_T &)
- static void [unconstrainUsingMultiplicativePickUp](#) (const stdair::SegmentCabin &, [BookingClassUnconstrainedDemandVectorMap_T](#) &, [UnconstrainedDemandVector_T](#) &, const stdair::DCP_T &, const stdair::DCP_T &, const stdair::Date_T &, const stdair::NbOfSegments_T &)

- static void [retrieveUnconstrainedDemandForFirstDCP](#) (const stdair::SegmentCabin &, [BookingClassUnconstrainedDemandVectorMap_T](#) &, [UnconstrainedDemandVector_T](#) &, const stdair::DCP_T &, const stdair::NbOfSegments_T &, const stdair::NbOfSegments_T &)
- static void [unconstrainUsingMultiplicativePickUp](#) ([HistoricalBookingHolder](#) &)

42.5.1 Detailed Description

Class wrapping the principal unconstraining algorithms and some accessory algorithms.

Definition at line 24 of file [Detruncator.hpp](#).

42.5.2 Member Function Documentation

42.5.2.1 void RMOL::Detruncator::unconstrainUsingAdditivePickUp
 (const stdair::SegmentCabin & *iSegmentCabin*,
BookingClassUnconstrainedDemandVectorMap_T &
ioBkgClassUncDemMap, **UnconstrainedDemandVector_T** &
ioQEquivalentDemandVector, const stdair::DCP_T & *iDCPBegin*, const stdair::DCP_T &
iDCPEnd, const stdair::Date_T & *iCurrentDate*) [static]

Unconstrain booking figures between two DCP's.

Definition at line 25 of file [Detruncator.cpp](#).

References [RMOL::GuillotineBlockHelper::getNbOfSegmentAlreadyPassedThisDTD\(\)](#).

42.5.2.2 void RMOL::Detruncator::unconstrainUsingMultiplicativePickUp
 (const stdair::SegmentCabin & *iSegmentCabin*,
BookingClassUnconstrainedDemandVectorMap_T &
ioBkgClassUncDemMap, **UnconstrainedDemandVector_T** &
ioQEquivalentDemandVector, const stdair::DCP_T & *iDCPBegin*, const stdair::DCP_T &
iDCPEnd, const stdair::Date_T & *iCurrentDate*, const stdair::NbOfSegments_T &
iNbOfDepartedSegments) [static]

Unconstrain booking figures between two DCP's.

Definition at line 317 of file [Detruncator.cpp](#).

References [RMOL::GuillotineBlockHelper::getNbOfSegmentAlreadyPassedThisDTD\(\)](#).

42.5.2.3 void RMOL::Detruncator::retrieveUnconstrainedDemandForFirstDCP
 (const stdair::SegmentCabin & *iSegmentCabin*,
BookingClassUnconstrainedDemandVectorMap_T &
ioBkgClassUncDemVectorMap, **UnconstrainedDemandVector_T**
 & *ioQEquivalentDemandVector*, const stdair::DCP_T & *iFirstDCP*, const
 stdair::NbOfSegments_T & *iNbOfSegments*, const stdair::NbOfSegments_T &
iNbOfUsedSegments) [static]

Retrieve unconstrained demand figures for the first DCP.

Definition at line 239 of file [Detruncator.cpp](#).

42.5.2.4 void RMOL::Detruncator::unconstrainUsingMultiplicativePickUp (HistoricalBookingHolder & ioHBHolder) [static]

Unconstrain the product-oriented booking figures for a given class ou Q-equivalent class.

Definition at line 558 of file [Detruncator.cpp](#).

References [RMOL::HistoricalBookingHolder::getCensorshipFlag\(\)](#), [RMOL::HistoricalBookingHolder::getHistoricalBookingHolder\(\)](#), [RMOL::HistoricalBookingHolder::getNbOfFlights\(\)](#), [RMOL::HistoricalBookingHolder::getNbOfUncensoredData\(\)](#), and [RMOL::HistoricalBookingHolder::setUnconstrainedDemand\(\)](#).

The documentation for this class was generated from the following files:

- [rmol/command/Detruncator.hpp](#)
- [rmol/command/Detruncator.cpp](#)

42.6 RMOL::DPOptimiser Class Reference

```
#include <rmol/bom/DPOptimiser.hpp>
```

Static Public Member Functions

- static void [optimalOptimisationByDP](#) (stdair::LegCabin &)
- static double [cdfGaussianQ](#) (const double, const double)

42.6.1 Detailed Description

Utility methods for the Dynamic Programming algorithms.

Definition at line 17 of file [DPOptimiser.hpp](#).

42.6.2 Member Function Documentation

42.6.2.1 void RMOL::DPOptimiser::optimalOptimisationByDP (stdair::LegCabin & ioLegCabin) [static]

Dynamic Programming to compute the cumulative protection levels and booking limits (described in the book Revenue Management - Talluri & Van Ryzin, p.41-42).

Definition at line 22 of file [DPOptimiser.cpp](#).

42.6.2.2 static double RMOL::DPOptimiser::cdfGaussianQ (const double , const double) [static]

Compute the cdf_Q of a gaussian.

The documentation for this class was generated from the following files:

- [rmol/bom/DPOptimiser.hpp](#)
- [rmol/bom/DPOptimiser.cpp](#)

42.7 RMOL::EMDetruncator Class Reference

```
#include <rmol/bom/EMDetruncator.hpp>
```

Static Public Member Functions

- static void [unconstrainUsingEMMethod](#) ([HistoricalBookingHolder](#) &)

42.7.1 Detailed Description

Utility for the Expectation-Maximisation algorithm.

Definition at line 12 of file [EMDetruncator.hpp](#).

42.7.2 Member Function Documentation

42.7.2.1 void [RMOL::EMDetruncator::unconstrainUsingEMMethod](#) (
[HistoricalBookingHolder](#) & *ioHistoricalBookingHolder*) [static]

Unconstrain the censored booking data using the Expection-Maximisation algorithm.

Definition at line 20 of file [EMDetruncator.cpp](#).

References [RMOL::HistoricalBookingHolder::getDemandMean\(\)](#), [RMOL::HistoricalBookingHolder::getListOfToBeUncor](#), [RMOL::HistoricalBookingHolder::getNbOfFlights\(\)](#), [RMOL::HistoricalBookingHolder::getNbOfUncensoredBookings\(\)](#), [RMOL::HistoricalBookingHolder::getNbOfUncensoredData\(\)](#), [RMOL::HistoricalBookingHolder::getStandardDeviation\(\)](#), [RMOL::HistoricalBookingHolder::getUncensoredStandardDeviation\(\)](#), [RMOL::HistoricalBookingHolder::getUnconstrainedDemand\(\)](#) and [RMOL::HistoricalBookingHolder::setUnconstrainedDemand\(\)](#).

The documentation for this class was generated from the following files:

- [rmol/bom/EMDetruncator.hpp](#)
- [rmol/bom/EMDetruncator.cpp](#)

42.8 RMOL::Emsr Class Reference

```
#include <rmol/bom/Emsr.hpp>
```

Static Public Member Functions

- static void [heuristicOptimisationByEmsr](#) (stdair::LegCabin &)
- static void [heuristicOptimisationByEmsrA](#) (stdair::LegCabin &)
- static void [heuristicOptimisationByEmsrB](#) (stdair::LegCabin &)

42.8.1 Detailed Description

Class Implementing the EMSR algorithm for Bid-Price Vector computing.

Definition at line 18 of file [Emsr.hpp](#).

42.8.2 Member Function Documentation

42.8.2.1 `void RMOL::Emsr::heuristicOptimisationByEmsr (stdair::LegCabin & ioLegCabin)`
[static]

Compute the Bid-Price Vector using the EMSR algorithm. Then compute the protection levels and booking limits by using the BPV.

For each class/bucket j with yield p_j and demand D_j , compute $p_j * \Pr(D_j \geq x)$ with x the capacity index. This value is called the EMSR (Expected Marginal Seat Revenue) of the class/bucket j with the remaining capacity of x . Thus, we have for each class/bucket a list of EMSR values. We merge all these lists and sort the values from high to low in order to obtain the BPV.

Definition at line 108 of file [Emsr.cpp](#).

References [RMOL::EmsrUtils::computeEmsrValue\(\)](#).

42.8.2.2 `void RMOL::Emsr::heuristicOptimisationByEmsrA (stdair::LegCabin & ioLegCabin)`
[static]

Calculate the optimal protections for the set of buckets/classes given in input, and update those buckets accordingly.

Definition at line 21 of file [Emsr.cpp](#).

References [RMOL::EmsrUtils::computeProtectionLevel\(\)](#).

42.8.2.3 `void RMOL::Emsr::heuristicOptimisationByEmsrB (stdair::LegCabin & ioLegCabin)`
[static]

Compute the protection levels and booking limits by using the EMSR-b algorithm.

Definition at line 64 of file [Emsr.cpp](#).

References [RMOL::EmsrUtils::computeAggregatedVirtualClass\(\)](#), and [RMOL::EmsrUtils::computeProtectionLevel\(\)](#).

The documentation for this class was generated from the following files:

- [rmol/bom/Emsr.hpp](#)
- [rmol/bom/Emsr.cpp](#)

42.9 RMOL::EmsrUtils Class Reference

```
#include <rmol/bom/EmsrUtils.hpp>
```

Static Public Member Functions

- static void [computeAggregatedVirtualClass](#) (stdair::VirtualClassStruct &, stdair::VirtualClassStruct &)
- static const stdair::ProtectionLevel_T [computeProtectionLevel](#) (stdair::VirtualClassStruct &, stdair::VirtualClassStruct &)
- static const double [computeEmsrValue](#) (double, stdair::VirtualClassStruct &)

42.9.1 Detailed Description

Forward declarations.

Definition at line 19 of file [EmsrUtils.hpp](#).

42.9.2 Member Function Documentation

42.9.2.1 void RMOL::EmsrUtils::computeAggregatedVirtualClass (stdair::VirtualClassStruct & *ioAggregatedVirtualClass*, stdair::VirtualClassStruct & *ioCurrentVirtualClass*)
[static]

Compute the aggregated class/bucket of classes/buckets 1,...j for EMSR-b algorithm.

Definition at line 19 of file [EmsrUtils.cpp](#).

Referenced by [RMOL::Emsr::heuristicOptimisationByEmsrB\(\)](#).

42.9.2.2 const stdair::ProtectionLevel_T RMOL::EmsrUtils::computeProtectionLevel (stdair::VirtualClassStruct & *ioAggregatedVirtualClass*, stdair::VirtualClassStruct & *ioNextVirtualClass*) [static]

Compute the protection level using the Little-Wood formular.

Definition at line 53 of file [EmsrUtils.cpp](#).

Referenced by [RMOL::Emsr::heuristicOptimisationByEmsrA\(\)](#), and [RMOL::Emsr::heuristicOptimisationByEmsrB\(\)](#).

42.9.2.3 const double RMOL::EmsrUtils::computeEmsrValue (double *iCapacity*, stdair::VirtualClassStruct & *ioVirtualClass*) [static]

Compute the EMSR value of a class/bucket.

Definition at line 80 of file [EmsrUtils.cpp](#).

Referenced by [RMOL::Emsr::heuristicOptimisationByEmsr\(\)](#).

The documentation for this class was generated from the following files:

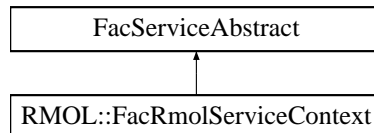
- [rmol/bom/EmsrUtils.hpp](#)
- [rmol/bom/EmsrUtils.cpp](#)

42.10 RMOL::FacRmolServiceContext Class Reference

Factory for the service context.

```
#include <rmol/factory/FacRmolServiceContext.hpp>
```

Inheritance diagram for RMOL::FacRmolServiceContext:



Public Member Functions

- [~FacRmolServiceContext\(\)](#)
- [RMOL_ServiceContext & create\(\)](#)

Static Public Member Functions

- static [FacRmolServiceContext & instance\(\)](#)

Protected Member Functions

- [FacRmolServiceContext\(\)](#)

42.10.1 Detailed Description

Factory for the service context.

Definition at line 22 of file [FacRmolServiceContext.hpp](#).

42.10.2 Constructor & Destructor Documentation

42.10.2.1 RMOL::FacRmolServiceContext::~~FacRmolServiceContext()

Destructor.

The Destruction put the `_instance` to NULL in order to be clean for the next `FacSimfqt-ServiceContext::instance()`.

Definition at line 17 of file [FacRmolServiceContext.cpp](#).

42.10.2.2 RMOL::FacRmolServiceContext::FacRmolServiceContext() [inline, protected]

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

Definition at line 57 of file [FacRmolServiceContext.hpp](#).

Referenced by [instance\(\)](#).

42.10.3 Member Function Documentation

42.10.3.1 FacRmolServiceContext & RMOL::FacRmolServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns

FacServiceContext&

Definition at line 22 of file [FacRmolServiceContext.cpp](#).

References [FacRmolServiceContext\(\)](#).

42.10.3.2 RMOL_ServiceContext & RMOL::FacRmolServiceContext::create ()

Create a new ServiceContext object.

This new object is added to the list of instantiated objects.

Returns

ServiceContext& The newly created object.

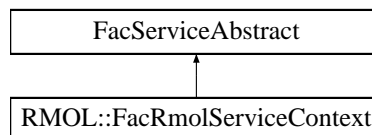
Definition at line 34 of file [FacRmolServiceContext.cpp](#).

The documentation for this class was generated from the following files:

- [rmol/factory/FacRmolServiceContext.hpp](#)
- [rmol/factory/FacRmolServiceContext.cpp](#)

42.11 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract:



The documentation for this class was generated from the following file:

- [rmol/factory/FacRmolServiceContext.hpp](#)

42.12 RMOL::Forecaster Class Reference

```
#include <rmol/command/Forecaster.hpp>
```

Static Public Member Functions

- static bool [forecastUsingAdditivePickUp](#) (stdair::FlightDate &, const stdair::DateTime_
T &)
- static bool [forecastUsingMultiplicativePickUp](#) (stdair::FlightDate &, const stdair::DateTime_
T &)

42.12.1 Detailed Description

Class wrapping the optimisation algorithms.

Definition at line 23 of file [Forecaster.hpp](#).

42.12.2 Member Function Documentation

42.12.2.1 bool RMOL::Forecaster::forecastUsingAdditivePickUp (stdair::FlightDate &
ioFlightDate, const stdair::DateTime_T & *iEventTime*) [static]

Forecast demand for a flight-date using additive pick-up method.

Definition at line 35 of file [Forecaster.cpp](#).

References [RMOL::Utilities::buildRemainingDCPList\(\)](#), and [RMOL::Utilities::buildRemainingDCPList2\(\)](#).

Referenced by [RMOL::RMOL_Service::optimise\(\)](#).

42.12.2.2 bool RMOL::Forecaster::forecastUsingMultiplicativePickUp (stdair::FlightDate &
ioFlightDate, const stdair::DateTime_T & *iEventTime*) [static]

Forecast demand for a flight-date using multiplicative pick-up method.

Definition at line 276 of file [Forecaster.cpp](#).

Referenced by [RMOL::RMOL_Service::optimise\(\)](#).

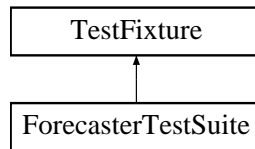
The documentation for this class was generated from the following files:

- [rmol/command/Forecaster.hpp](#)
- [rmol/command/Forecaster.cpp](#)

42.13 ForecasterTestSuite Class Reference

```
#include <test/rmol/ForecasterTestSuite.hpp>
```

Inheritance diagram for ForecasterTestSuite:



Public Member Functions

- void [testQForecaster](#) ()
- [ForecasterTestSuite](#) ()

Protected Attributes

- `std::stringstream` [_describeKey](#)

42.13.1 Detailed Description

Definition at line 6 of file [ForecasterTestSuite.hpp](#).

42.13.2 Constructor & Destructor Documentation

42.13.2.1 `ForecasterTestSuite::ForecasterTestSuite ()`

Constructor.

42.13.3 Member Function Documentation

42.13.3.1 `void ForecasterTestSuite::testQForecaster ()`

Test Q-forecaster.

42.13.4 Member Data Documentation

42.13.4.1 `std::stringstream ForecasterTestSuite::_describeKey` [protected]

Definition at line 19 of file [ForecasterTestSuite.hpp](#).

The documentation for this class was generated from the following file:

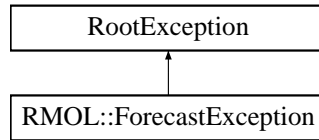
- [test/rmol/ForecasterTestSuite.hpp](#)

42.14 RMOL::ForecastException Class Reference

Forecast-related exception.

```
#include <rmol/RMOL_Types.hpp>
```

Inheritance diagram for RMOL::ForecastException:



Public Member Functions

- [ForecastException](#) (const std::string &iWhat)

42.14.1 Detailed Description

Forecast-related exception.

Definition at line 51 of file [RMOL_Types.hpp](#).

42.14.2 Constructor & Destructor Documentation

42.14.2.1 RMOL::ForecastException::ForecastException (const std::string & *iWhat*)
[inline]

Constructor.

Definition at line 54 of file [RMOL_Types.hpp](#).

The documentation for this class was generated from the following file:

- [rmol/RMOL_Types.hpp](#)

42.15 RMOL::GuillotineBlockHelper Class Reference

```
#include <rmol/bom/GuillotineBlockHelper.hpp>
```

Static Public Member Functions

- static stdair::NbOfSegments_T [getNbOfSegmentAlreadyPassedThisDTD](#) (const stdair::GuillotineBlock &, const stdair::DTD_T &, const stdair::Date_T &)
- static bool [hasPassedThisDTD](#) (const stdair::SegmentCabin &, const stdair::DTD_T &, const stdair::Date_T &)

42.15.1 Detailed Description

Class representing the actual business functions for an airline guillotine block.

Definition at line 23 of file [GuillotineBlockHelper.hpp](#).

42.15.2 Member Function Documentation

42.15.2.1 `stdair::NbOfSegments_T RMOL::GuillotineBlockHelper::getNbOfSegmentAlreadyPassedThisDTD (const stdair::GuillotineBlock & iGB, const stdair::DTD_T & iDTD, const stdair::Date_T & iCurrentDate) [static]`

Retrieve the number of similar segments which already passed the given DTD.

Definition at line 20 of file [GuillotineBlockHelper.cpp](#).

References [hasPassedThisDTD\(\)](#).

Referenced by [RMOL::Utilities::getNbOfDepartedSimilarSegments\(\)](#), [RMOL::Detruncator::unconstrainUsingAdditivePickUp\(\)](#) and [RMOL::Detruncator::unconstrainUsingMultiplicativePickUp\(\)](#).

42.15.2.2 `bool RMOL::GuillotineBlockHelper::hasPassedThisDTD (const stdair::SegmentCabin & iSegmentCabin, const stdair::DTD_T & iDTD, const stdair::Date_T & iCurrentDate) [static]`

Check if the given segment has passed the given DTD.

Definition at line 42 of file [GuillotineBlockHelper.cpp](#).

Referenced by [getNbOfSegmentAlreadyPassedThisDTD\(\)](#).

The documentation for this class was generated from the following files:

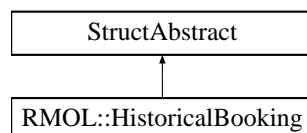
- [rmol/bom/GuillotineBlockHelper.hpp](#)
- [rmol/bom/GuillotineBlockHelper.cpp](#)

42.16 RMOL::HistoricalBooking Struct Reference

Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag.

```
#include <rmol/bom/HistoricalBooking.hpp>
```

Inheritance diagram for RMOL::HistoricalBooking:



Public Member Functions

- const stdair::NbOfBookings_T & [getNbOfBookings](#) () const
- const stdair::NbOfBookings_T & [getUnconstrainedDemand](#) () const
- const stdair::Flag_T & [getFlag](#) () const
- void [setUnconstrainedDemand](#) (const stdair::NbOfBookings_T &iDemand)
- void [setParameters](#) (const stdair::NbOfBookings_T, const stdair::Flag_T)
- void [toStream](#) (std::ostream &ioOut) const
- const std::string [describe](#) () const
- void [display](#) () const
- [HistoricalBooking](#) (const stdair::NbOfBookings_T, const stdair::Flag_T)
- [HistoricalBooking](#) ()
- [HistoricalBooking](#) (const [HistoricalBooking](#) &)
- virtual [~HistoricalBooking](#) ()

42.16.1 Detailed Description

Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag.

Definition at line 17 of file [HistoricalBooking.hpp](#).

42.16.2 Constructor & Destructor Documentation

42.16.2.1 RMOL::HistoricalBooking::HistoricalBooking (const stdair::NbOfBookings_T *iNbOfBookings*, const stdair::Flag_T *iFlag*)

Main constructor.

Definition at line 21 of file [HistoricalBooking.cpp](#).

42.16.2.2 RMOL::HistoricalBooking::HistoricalBooking ()

Default constructor.

Definition at line 15 of file [HistoricalBooking.cpp](#).

42.16.2.3 RMOL::HistoricalBooking::HistoricalBooking (const [HistoricalBooking](#) & *iHistoricalBooking*)

Copy constructor.

Definition at line 29 of file [HistoricalBooking.cpp](#).

42.16.2.4 RMOL::HistoricalBooking::~~HistoricalBooking () [virtual]

Destructor.

Definition at line 36 of file [HistoricalBooking.cpp](#).

42.16.3 Member Function Documentation

42.16.3.1 `const stdair::NbOfBookings_T& RMOL::HistoricalBooking::getNbOfBookings ()`
`const [inline]`

Getter for the booking.

Definition at line 22 of file [HistoricalBooking.hpp](#).

Referenced by [RMOL::HistoricalBookingHolder::calculateExpectedDemand\(\)](#), [RMOL::HistoricalBookingHolder::getHistoricalBooking\(\)](#), [RMOL::HistoricalBookingHolder::getNbOfUncensoredBookings\(\)](#), [RMOL::HistoricalBookingHolder::getUncensoredStar\(\)](#), [RMOL::HistoricalBookingHolder::toStream\(\)](#), and [toStream\(\)](#).

42.16.3.2 `const stdair::NbOfBookings_T& RMOL::HistoricalBooking::getUnconstrainedDemand ()`
`const [inline]`

Getter for the unconstrained bookings.

Definition at line 26 of file [HistoricalBooking.hpp](#).

Referenced by [RMOL::HistoricalBookingHolder::getDemandMean\(\)](#), [RMOL::HistoricalBookingHolder::getStandardDeviation\(\)](#), [RMOL::HistoricalBookingHolder::getUnconstrainedDemand\(\)](#), [RMOL::HistoricalBookingHolder::toStream\(\)](#), and [toStream\(\)](#).

42.16.3.3 `const stdair::Flag_T& RMOL::HistoricalBooking::getFlag ()` `const [inline]`

Getter for the flag of censorship: "false" means that the bookings are not censored.

Definition at line 31 of file [HistoricalBooking.hpp](#).

Referenced by [RMOL::HistoricalBookingHolder::getCensorshipFlag\(\)](#), [RMOL::HistoricalBookingHolder::getListOfToBeUncensored\(\)](#), [RMOL::HistoricalBookingHolder::getNbOfUncensoredBookings\(\)](#), [RMOL::HistoricalBookingHolder::toStream\(\)](#), and [toStream\(\)](#).

42.16.3.4 `void RMOL::HistoricalBooking::setUnconstrainedDemand (const stdair::NbOfBookings_T & iDemand)` `[inline]`

Setter for the unconstraining demand.

Definition at line 38 of file [HistoricalBooking.hpp](#).

42.16.3.5 `void RMOL::HistoricalBooking::setParameters (const stdair::NbOfBookings_T iNbOfBookings, const stdair::Flag_T iFlag)`

Setter for all parameters.

Definition at line 41 of file [HistoricalBooking.cpp](#).

42.16.3.6 `void RMOL::HistoricalBooking::toStream (std::ostream & ioOut)` `const`

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream

Returns

ostream& the output stream.

Definition at line 57 of file [HistoricalBooking.cpp](#).

References [getFlag\(\)](#), [getNbOfBookings\(\)](#), and [getUnconstrainedDemand\(\)](#).

Referenced by [display\(\)](#).

42.16.3.7 const std::string RMOL::HistoricalBooking::describe () const

Give a description of the structure (for display purposes).

Definition at line 48 of file [HistoricalBooking.cpp](#).

42.16.3.8 void RMOL::HistoricalBooking::display () const

Display on standard output.

Definition at line 66 of file [HistoricalBooking.cpp](#).

References [toStream\(\)](#).

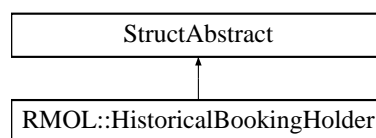
The documentation for this struct was generated from the following files:

- [rmol/bom/HistoricalBooking.hpp](#)
- [rmol/bom/HistoricalBooking.cpp](#)

42.17 RMOL::HistoricalBookingHolder Struct Reference

```
#include <rmol/bom/HistoricalBookingHolder.hpp>
```

Inheritance diagram for RMOL::HistoricalBookingHolder:

**Public Member Functions**

- const short [getNbOfFlights](#) () const
- const short [getNbOfUncensoredData](#) () const
- const stdair::NbOfBookings_T [getNbOfUncensoredBookings](#) () const
- const double [getUncensoredStandardDeviation](#) (const double &iMeanOfUncensoredBookings, const short iNbOfUncensoredData) const
- const double [getDemandMean](#) () const
- const double [getStandardDeviation](#) (const double) const
- const std::vector< bool > [getListOfToBeUnconstrainedFlags](#) () const

- const stdair::NbOfBookings_T & [getHistoricalBooking](#) (const short i) const
- const stdair::NbOfBookings_T & [getUnconstrainedDemand](#) (const short i) const
- const stdair::Flag_T & [getCensorshipFlag](#) (const short i) const
- const stdair::NbOfBookings_T & [getUnconstrainedDemandOnFirstElement](#) () const
- const stdair::NbOfBookings_T [calculateExpectedDemand](#) (const double, const double, const short, const stdair::NbOfBookings_T) const
- void [setUnconstrainedDemand](#) (const stdair::NbOfBookings_T &iExpectedDemand, const short i)
- void [addHistoricalBooking](#) (const [HistoricalBooking](#) &iHistoricalBooking)
- void [toStream](#) (std::ostream &ioOut) const
- const std::string [describe](#) () const
- void [display](#) () const
- virtual [~HistoricalBookingHolder](#) ()
- [HistoricalBookingHolder](#) ()

42.17.1 Detailed Description

Holder of a [HistoricalBookingList](#) object (for memory allocation and recollection purposes).

Definition at line [23](#) of file [HistoricalBookingHolder.hpp](#).

42.17.2 Constructor & Destructor Documentation

42.17.2.1 RMOL::HistoricalBookingHolder::~~HistoricalBookingHolder () [virtual]

Destructor.

Definition at line [23](#) of file [HistoricalBookingHolder.cpp](#).

42.17.2.2 RMOL::HistoricalBookingHolder::HistoricalBookingHolder ()

Constructor.

Protected to force the use of the Factory.

Definition at line [19](#) of file [HistoricalBookingHolder.cpp](#).

42.17.3 Member Function Documentation

42.17.3.1 const short RMOL::HistoricalBookingHolder::getNbOfFlights () const

Get number of flights.

Definition at line [28](#) of file [HistoricalBookingHolder.cpp](#).

Referenced by [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#), and [RMOL::Detruncator::unconstrainUsingMultip](#)

42.17.3.2 `const short RMOL::HistoricalBookingHolder::getNbOfUncensoredData () const`

Get number of uncensored booking data.

Definition at line 33 of file [HistoricalBookingHolder.cpp](#).

Referenced by [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#), and [RMOL::Detruncator::unconstrainUsingMultip](#)

42.17.3.3 `const std::pair<NbOfBookings,T> RMOL::HistoricalBookingHolder::getNbOfUncensoredBookings () const`

Get number of uncensored bookings.

Definition at line 49 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getFlag\(\)](#), and [RMOL::HistoricalBooking::getNbOfBookings\(\)](#).

Referenced by [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#).

42.17.3.4 `const double RMOL::HistoricalBookingHolder::getUncensoredStandardDeviation (const double & iMeanOfUncensoredBookings, const short iNbOfUncensoredData) const`

Get standard deviation of uncensored bookings.

Definition at line 69 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getNbOfBookings\(\)](#).

Referenced by [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#).

42.17.3.5 `const double RMOL::HistoricalBookingHolder::getDemandMean () const`

Get mean of historical demand.

Definition at line 95 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getUnconstrainedDemand\(\)](#).

Referenced by [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#).

42.17.3.6 `const double RMOL::HistoricalBookingHolder::getStandardDeviation (const double iDemandMean) const`

Get standard deviation of demand.

Definition at line 116 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getUnconstrainedDemand\(\)](#).

Referenced by [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#).

42.17.3.7 `const std::vector<bool> RMOL::HistoricalBookingHolder::getListOfToBeUnconstrainedFlags () const`

Get the list of flags of need to be unconstrained.

Definition at line 140 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getFlag\(\)](#).

Referenced by [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#).

42.17.3.8 `const stdair::NbOfBookings_T & RMOL::HistoricalBookingHolder::getHistoricalBooking
(const short i) const`

Get the historical booking of the (i+1)-th flight.

Definition at line 161 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getNbOfBookings\(\)](#).

Referenced by [RMOL::Detruncator::unconstrainUsingMultiplicativePickUp\(\)](#).

42.17.3.9 `const stdair::NbOfBookings_T & RMOL::HistoricalBookingHolder::getUnconstrainedDemand
(const short i) const`

Get the unconstraining demand of the (i+1)-th flight.

Definition at line 169 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getUnconstrainedDemand\(\)](#).

Referenced by [getUnconstrainedDemandOnFirstElement\(\)](#), and [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#).

42.17.3.10 `const stdair::Flag_T & RMOL::HistoricalBookingHolder::getCensorshipFlag (const
short i) const`

Get the flag of the (i+1)-th flight.

Definition at line 177 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getFlag\(\)](#).

Referenced by [RMOL::Detruncator::unconstrainUsingMultiplicativePickUp\(\)](#).

42.17.3.11 `const stdair::NbOfBookings_T& RMOL::HistoricalBookingHolder::getUnconstrainedDemandOnFirstElement
() const [inline]`

Get the unconstraining demand of the first flight.

Definition at line 60 of file [HistoricalBookingHolder.hpp](#).

References [getUnconstrainedDemand\(\)](#).

42.17.3.12 `const stdair::NbOfBookings_T RMOL::HistoricalBookingHolder::calculateExpectedDemand
(const double iMean, const double iSD, const short i, const stdair::NbOfBookings_T
iDemand) const`

Calculate the expected demand.

Definition at line 191 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getNbOfBookings\(\)](#).

42.17.3.13 void RMOL::HistoricalBookingHolder::setUnconstrainedDemand (const
stdair::NbOfBookings_T & *iExpectedDemand*, const short *i*)

Set the expected historical demand of the (i+1)-th flight.

Definition at line 185 of file [HistoricalBookingHolder.cpp](#).

Referenced by [RMOL::EMDetruncator::unconstrainUsingEMMethod\(\)](#), and [RMOL::Detruncator::unconstrainUsingMultipleEMMethod\(\)](#).

42.17.3.14 void RMOL::HistoricalBookingHolder::addHistoricalBooking (const
HistoricalBooking & *iHistoricalBooking*)

Add a [HistoricalBooking](#) object to the holder.

Definition at line 236 of file [HistoricalBookingHolder.cpp](#).

42.17.3.15 void RMOL::HistoricalBookingHolder::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters

<i>ostream&</i> the output stream

Returns

ostream& the output stream.

Definition at line 241 of file [HistoricalBookingHolder.cpp](#).

References [RMOL::HistoricalBooking::getFlag\(\)](#), [RMOL::HistoricalBooking::getNbOfBookings\(\)](#), and [RMOL::HistoricalBooking::getUnconstrainedDemand\(\)](#).

Referenced by [display\(\)](#).

42.17.3.16 const std::string RMOL::HistoricalBookingHolder::describe () const

Give a description of the structure (for display purposes).

Definition at line 265 of file [HistoricalBookingHolder.cpp](#).

42.17.3.17 void RMOL::HistoricalBookingHolder::display () const

Display on standard output.

Definition at line 273 of file [HistoricalBookingHolder.cpp](#).

References [toStream\(\)](#).

The documentation for this struct was generated from the following files:

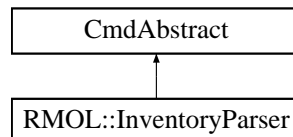
- [rmol/bom/HistoricalBookingHolder.hpp](#)
- [rmol/bom/HistoricalBookingHolder.cpp](#)

42.18 RMOL::InventoryParser Class Reference

Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory.

```
#include <rmol/command/InventoryParser.hpp>
```

Inheritance diagram for RMOL::InventoryParser:



Static Public Member Functions

- static bool [parseInputFileAndBuildBom](#) (const std::string &iInputFileName, stdair::BomRoot &)

42.18.1 Detailed Description

Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory.

Definition at line 25 of file [InventoryParser.hpp](#).

42.18.2 Member Function Documentation

42.18.2.1 bool RMOL::InventoryParser::parseInputFileAndBuildBom (const std::string &iInputFileName, stdair::BomRoot & ioBomRoot) [static]

Parse the input values from a CSV-formatted inventory file.

Parameters

<i>const</i>	std::string& iInputFileName Inventory file to be parsed.
<i>stdair::BomRoot</i>	The BOM tree.

Returns

bool Whether or not the parsing was successful.

Definition at line 36 of file [InventoryParser.cpp](#).

Referenced by [RMOL::RMOL_Service::parseAndLoad\(\)](#).

The documentation for this class was generated from the following files:

- [rmol/command/InventoryParser.hpp](#)

- [rmol/command/InventoryParser.cpp](#)

42.19 RMOL::MCOptimiser Class Reference

```
#include <rmol/bom/MCOptimiser.hpp>
```

Static Public Member Functions

- static void [optimalOptimisationByMCIntegration](#) (stdair::LegCabin &)
- static stdair::GeneratedDemandVector_T [generateDemandVector](#) (const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const unsigned int &)
- static void [optimisationByMCIntegration](#) (stdair::LegCabin &)

42.19.1 Detailed Description

Utility methods for the Monte-Carlo algorithms.

Definition at line 19 of file [MCOptimiser.hpp](#).

42.19.2 Member Function Documentation

42.19.2.1 void RMOL::MCOptimiser::optimalOptimisationByMCIntegration (stdair::LegCabin & *ioLegCabin*) [static]

Calculate the optimal protections for the set of buckets/classes given in input, and update those buckets accordingly.

The Monte Carlo Integration algorithm (see The Theory and Practice of Revenue Management, by Kalyan T. Talluri and Garret J. van Ryzin, Kluwer Academic Publishers, for the details) is used.

Definition at line 28 of file [MCOptimiser.cpp](#).

42.19.2.2 stdair::GeneratedDemandVector_T RMOL::MCOptimiser::generateDemandVector (const stdair::MeanValue_T & *iMean*, const stdair::StdDevValue_T & *iStdDev*, const unsigned int & *K*) [static]

Monte-Carlo

Definition at line 154 of file [MCOptimiser.cpp](#).

Referenced by [optimisationByMCIntegration\(\)](#).

42.19.2.3 void RMOL::MCOptimiser::optimisationByMCIntegration (stdair::LegCabin & *ioLegCabin*) [static]

Definition at line 175 of file [MCOptimiser.cpp](#).

References [generateDemandVector\(\)](#).

Referenced by [RMOL::Optimiser::optimiseUsingOnDForecast\(\)](#).

The documentation for this class was generated from the following files:

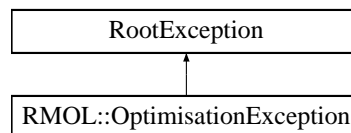
- [rmol/bom/MCOptimiser.hpp](#)
- [rmol/bom/MCOptimiser.cpp](#)

42.20 RMOL::OptimisationException Class Reference

Optimisation-related exception.

```
#include <rmol/RMOL_Types.hpp>
```

Inheritance diagram for RMOL::OptimisationException:



Public Member Functions

- [OptimisationException](#) (const std::string &iWhat)

42.20.1 Detailed Description

Optimisation-related exception.

Definition at line 61 of file [RMOL_Types.hpp](#).

42.20.2 Constructor & Destructor Documentation

42.20.2.1 RMOL::OptimisationException::OptimisationException (const std::string & iWhat) [inline]

Constructor.

Definition at line 64 of file [RMOL_Types.hpp](#).

The documentation for this class was generated from the following file:

- [rmol/RMOL_Types.hpp](#)

42.21 RMOL::Optimiser Class Reference

```
#include <rmol/command/Optimiser.hpp>
```

Static Public Member Functions

- static void [optimalOptimisationByMCIntegration](#) (const int K, stdair::LegCabin &)
- static void [optimalOptimisationByDP](#) (stdair::LegCabin &)
- static void [heuristicOptimisationByEmsr](#) (stdair::LegCabin &)
- static void [heuristicOptimisationByEmsrA](#) (stdair::LegCabin &)
- static void [heuristicOptimisationByEmsrB](#) (stdair::LegCabin &)
- static void [optimise](#) (stdair::FlightDate &)
- static void [buildVirtualClassListForLegBasedOptimisation](#) (stdair::LegCabin &)
- static double [optimiseUsingOnDForecast](#) (stdair::FlightDate &, const bool &iReduceFluctuations=false)

42.21.1 Detailed Description

Class wrapping the optimisation algorithms.

Definition at line 18 of file [Optimiser.hpp](#).

42.21.2 Member Function Documentation

42.21.2.1 void RMOL::Optimiser::optimalOptimisationByMCIntegration (const int K, stdair::LegCabin & *ioLegCabin*) [static]

Monte Carlo Integration algorithm.

Calculate the optimal protections for the set of buckets/classes given in input, and update those buckets accordingly.

The Monte Carlo Integration algorithm (see The Theory and Practice of Revenue Management, by Kalyan T. Talluri and Garret J. van Ryzin, Kluwer Academic Publishers, for the details) is used. Hence, K is the number of random draws to perform. 100 is a minimum for K, as statistics must be drawn from those random generations.

Definition at line 29 of file [Optimiser.cpp](#).

Referenced by [optimise\(\)](#).

42.21.2.2 void RMOL::Optimiser::optimalOptimisationByDP (stdair::LegCabin & *ioLegCabin*) [static]

Dynamic Programming.

Definition at line 63 of file [Optimiser.cpp](#).

42.21.2.3 void RMOL::Optimiser::heuristicOptimisationByEmsr (stdair::LegCabin & *ioLegCabin*) [static]

EMRS algorithm.

Definition at line 68 of file [Optimiser.cpp](#).

42.21.2.4 `void RMOL::Optimiser::heuristicOptimisationByEmsrA (stdair::LegCabin & ioLegCabin) [static]`

EMRS-a algorithm.

Definition at line 73 of file [Optimiser.cpp](#).

42.21.2.5 `void RMOL::Optimiser::heuristicOptimisationByEmsrB (stdair::LegCabin & ioLegCabin) [static]`

EMRS-b algorithm.

Definition at line 78 of file [Optimiser.cpp](#).

42.21.2.6 `void RMOL::Optimiser::optimise (stdair::FlightDate & ioFlightDate) [static]`

Optimise a flight-date using leg-based Monte Carlo Integration.

Definition at line 83 of file [Optimiser.cpp](#).

References [buildVirtualClassListForLegBasedOptimisation\(\)](#), and [optimalOptimisation-ByMCIntegration\(\)](#).

42.21.2.7 `void RMOL::Optimiser::buildVirtualClassListForLegBasedOptimisation (stdair::LegCabin & ioLegCabin) [static]`

Build the virtual class list for the given leg-cabin.

Definition at line 112 of file [Optimiser.cpp](#).

Referenced by [optimise\(\)](#).

42.21.2.8 `double RMOL::Optimiser::optimiseUsingOnDForecast (stdair::FlightDate & ioFlightDate, const bool & iReduceFluctuations = false) [static]`

[Optimiser](#)

Definition at line 156 of file [Optimiser.cpp](#).

References [RMOL::MCOptimiser::optimisationByMCIntegration\(\)](#).

Referenced by [RMOL::RMOL_Service::optimiseOnD\(\)](#), [RMOL::RMOL_Service::optimiseOnDUsingAdvancedRMCooperation\(\)](#) and [RMOL::RMOL_Service::optimiseOnDUsingRMCooperation\(\)](#).

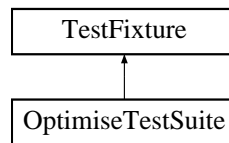
The documentation for this class was generated from the following files:

- [rmol/command/Optimiser.hpp](#)
- [rmol/command/Optimiser.cpp](#)

42.22 OptimiseTestSuite Class Reference

```
#include <test/rmol/OptimiseTestSuite.hpp>
```

Inheritance diagram for OptimiseTestSuite:



Public Member Functions

- void [testOptimiseMC](#) ()
- void [testOptimiseDP](#) ()
- void [testOptimiseEMSR](#) ()
- void [testOptimiseEMSRa](#) ()
- void [testOptimiseEMSRb](#) ()
- [OptimiseTestSuite](#) ()

Protected Attributes

- `std::stringstream` [_describeKey](#)

42.22.1 Detailed Description

Definition at line 6 of file [OptimiseTestSuite.hpp](#).

42.22.2 Constructor & Destructor Documentation

42.22.2.1 `OptimiseTestSuite::OptimiseTestSuite ()`

Test some error detection functionalities. Constructor.

42.22.3 Member Function Documentation

42.22.3.1 `void OptimiseTestSuite::testOptimiseMC ()`

Test the Monte-Carlo (MC) Optimisation functionality.

42.22.3.2 `void OptimiseTestSuite::testOptimiseDP ()`

Test the Dynamic Programming (DP) Optimisation functionality.

42.22.3.3 `void OptimiseTestSuite::testOptimiseEMSR ()`

Test the Expected Marginal Seat Revenue (EMSR) Optimisation functionality.

42.22.3.4 `void OptimiseTestSuite::testOptimiseEMSRa ()`

Test the Expected Marginal Seat Revenue, variant a (EMSR-a), Optimisation functionality.

42.22.3.5 void OptimiseTestSuite::testOptimiseEMSRb ()

Test the Expected Marginal Seat Revenue, variant b (EMSR-b), Optimisation functionality.

42.22.4 Member Data Documentation

42.22.4.1 std::stringstream OptimiseTestSuite::_describeKey [protected]

Definition at line 43 of file [OptimiseTestSuite.hpp](#).

The documentation for this class was generated from the following file:

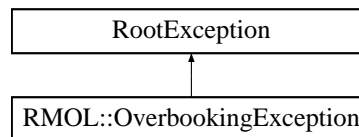
- test/rmol/[OptimiseTestSuite.hpp](#)

42.23 RMOL::OverbookingException Class Reference

Overbooking-related exception.

```
#include <rmol/RMOL_Types.hpp>
```

Inheritance diagram for RMOL::OverbookingException:



Public Member Functions

- [OverbookingException](#) (const std::string &iWhat)

42.23.1 Detailed Description

Overbooking-related exception.

Definition at line 31 of file [RMOL_Types.hpp](#).

42.23.2 Constructor & Destructor Documentation

42.23.2.1 RMOL::OverbookingException::OverbookingException (const std::string &iWhat)
[inline]

Constructor.

Definition at line 34 of file [RMOL_Types.hpp](#).

The documentation for this class was generated from the following file:

- [rmol/RMOL_Types.hpp](#)

42.24 RMOL::RMOL_Service Class Reference

Interface for the [RMOL](#) Services.

```
#include <rmol/RMOL_Service.hpp>
```

Public Member Functions

- [RMOL_Service](#) (const stdair::BasLogParams &, const stdair::BasDBParams &)
- [RMOL_Service](#) (const stdair::BasLogParams &)
- [RMOL_Service](#) (stdair::STDAIR_ServicePtr_T)
- void [parseAndLoad](#) (const stdair::CabinCapacity_T &iCabinCapacity, const stdair::Filename_T &iDemandAndClassDataFile)
- void [setUpStudyStatManager](#) ()
- [~RMOL_Service](#) ()
- void [buildSampleBom](#) ()
- void [optimalOptimisationByMCIntegration](#) (const int K)
- void [optimalOptimisationByDP](#) ()
- void [heuristicOptimisationByEmsr](#) ()
- void [heuristicOptimisationByEmsrA](#) ()
- void [heuristicOptimisationByEmsrB](#) ()
- bool [optimise](#) (stdair::FlightDate &, const stdair::DateTime_T &, const stdair::ForecastingMethod &, const stdair::PartnershipTechnique &)
- void [forecastOnD](#) (const stdair::DateTime_T &)
- stdair::YieldFeatures * [getYieldFeatures](#) (const stdair::OnDDate &, const stdair::CabinCode_T &, stdair::BomRoot &)
- void [forecastOnD](#) (const stdair::YieldFeatures &, stdair::OnDDate &, const stdair::CabinCode_T &, const stdair::DTD_T &, stdair::BomRoot &)
- void [setOnDForecast](#) (const stdair::AirlineClassList &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, stdair::OnDDate &, const stdair::CabinCode_T &, stdair::BomRoot &)
- void [setOnDForecast](#) (const stdair::AirlineCode_T &, const stdair::Date_T &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::CabinCode_T &, const stdair::ClassCode_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::Yield_T &, stdair::BomRoot &)
- void [setOnDForecast](#) (const stdair::AirlineCodeList_T &, const stdair::AirlineCode_T &, const stdair::Date_T &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::CabinCode_T &, const stdair::ClassCodeList_T &, const stdair::MeanValue_T &, const stdair::StdDevValue_T &, const stdair::Yield_T &, stdair::BomRoot &)
- void [resetDemandInformation](#) (const stdair::DateTime_T &)
- void [resetDemandInformation](#) (const stdair::DateTime_T &, const stdair::Inventory &)
- void [projectAggregatedDemandOnLegCabins](#) (const stdair::DateTime_T &)
- void [projectOnDDemandOnLegCabinsUsingYP](#) (const stdair::DateTime_T &)
- void [projectOnDDemandOnLegCabinsUsingDA](#) (const stdair::DateTime_T &)

- void [projectOnDDemandOnLegCabinsUsingDYP](#) (const stdair::DateTime_T &)
- void [projectOnDDemandOnLegCabinsUsingDYP](#) (const stdair::DateTime_T &, const stdair::Inventory &)
- void [optimiseOnD](#) (const stdair::DateTime_T &)
- void [optimiseOnDUsingRMCooperation](#) (const stdair::DateTime_T &)
- void [optimiseOnDUsingAdvancedRMCooperation](#) (const stdair::DateTime_T &)
- void [updateBidPrice](#) (const stdair::DateTime_T &)
- void [updateBidPrice](#) (const stdair::FlightDate &, stdair::BomRoot &)
- std::string [jsonExport](#) (const stdair::AirlineCode_T &, const stdair::FlightNumber_T &, const stdair::Date_T & iDepartureDate) const
- std::string [csvDisplay](#) () const

42.24.1 Detailed Description

Interface for the [RMOL](#) Services.

Definition at line 39 of file [RMOL_Service.hpp](#).

42.24.2 Constructor & Destructor Documentation

42.24.2.1 RMOL::RMOL_Service::RMOL_Service (const stdair::BasLogParams & *iLogParams*, const stdair::BasDBParams & *iDBParams*)

Constructor.

The `initRmolService()` method is called; see the corresponding documentation for more details.

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that a session can be created on the corresponding database.

Parameters

<i>const</i> stdair::BasLogParams&	Parameters for the output log stream.
<i>const</i> stdair::BasDBParams&	Parameters for the database access.

Definition at line 84 of file [RMOL_Service.cpp](#).

42.24.2.2 RMOL::RMOL_Service::RMOL_Service (const stdair::BasLogParams & *iLogParams*)

Constructor.

The `initRmolService()` method is called; see the corresponding documentation for more details.

Moreover, a reference on an output stream is given, so that log outputs can be directed onto that stream.

Parameters

<code>const</code> stdair::BasLogParams& Parameters for the output log stream.
--

Definition at line 63 of file [RMOL_Service.cpp](#).

42.24.2.3 RMOL::RMOL_Service::RMOL_Service (stdair::STDAIR_ServicePtr_T ioSTDAIRServicePtr)

Constructor.

The `initRmolService()` method is called; see the corresponding documentation for more details.

Moreover, as no reference on any output stream is given, it is assumed that the StdAir log service has already been initialised with the proper log output stream by some other methods in the calling chain (for instance, when the [RMOL_Service](#) is itself being initialised by another library service such as [AIRINV_Service](#)).

Parameters

<code>STDAIR_ServicePtr_T</code> - the shared pointer of stdair service.
--

Definition at line 106 of file [RMOL_Service.cpp](#).

42.24.2.4 RMOL::RMOL_Service::~~RMOL_Service ()

Destructor.

Definition at line 123 of file [RMOL_Service.cpp](#).

42.24.3 Member Function Documentation

42.24.3.1 void RMOL::RMOL_Service::parseAndLoad (const stdair::CabinCapacity_T & iCabinCapacity, const stdair::Filename_T & iDemandAndClassDataFile)

Parse the optimisation-related data and load them into memory.

First, the `STDAIR_Service::buildDummyInventory()` method is called, for [RMOL](#) and with the given cabin capacity, in order to build the minimum required flight-date structure in order to perform an optimisation on a leg-cabin.

The CSV input file describes the problem to be optimised, i.e.:

- the demand specifications for all the booking classes (mean and standard deviations for the demand distribution); the yields corresponding to those booking classes.

That CSV file is parsed and instantiated in memory accordingly. The leg-cabin capacity has been set at the initialisation of the ([RMOL](#)) service.

Parameters

<i>const</i>	stdair::CabinCapacity& Capacity of the leg-cabin to be optimised.
<i>const</i>	stdair::Filename_T& (CSV) input file.

Definition at line 200 of file [RMOL_Service.cpp](#).

References [RMOL::InventoryParser::parseInputFileAndBuildBom\(\)](#).

Referenced by [main\(\)](#).

42.24.3.2 void RMOL::RMOL_Service::setUpStudyStatManager ()

Set up the StudyStatManager.

42.24.3.3 void RMOL::RMOL_Service::buildSampleBom ()

Build a sample BOM tree, and attach it to the BomRoot instance.

See also

[stdair::CmdBomManager::buildSampleBom\(\)](#) for more details.

Definition at line 224 of file [RMOL_Service.cpp](#).

Referenced by [main\(\)](#).

42.24.3.4 void RMOL::RMOL_Service::optimalOptimisationByMCIntegration (const int K)

Single resource optimization using the Monte Carlo algorithm.

Definition at line 272 of file [RMOL_Service.cpp](#).

Referenced by [optimise\(\)](#).

42.24.3.5 void RMOL::RMOL_Service::optimalOptimisationByDP ()

Single resource optimization using dynamic programming.

Definition at line 312 of file [RMOL_Service.cpp](#).

Referenced by [optimise\(\)](#).

42.24.3.6 void RMOL::RMOL_Service::heuristicOptimisationByEmsr ()

Single resource optimization using EMSR heuristic.

Definition at line 316 of file [RMOL_Service.cpp](#).

Referenced by [optimise\(\)](#).

42.24.3.7 void RMOL::RMOL_Service::heuristicOptimisationByEmsrA ()

Single resource optimization using EMSR-a heuristic.

Definition at line 357 of file [RMOL_Service.cpp](#).

Referenced by [optimise\(\)](#).

42.24.3.8 void RMOL::RMOL_Service::heuristicOptimisationByEmsrB ()

Single resource optimization using EMSR-b heuristic.

Definition at line 378 of file [RMOL_Service.cpp](#).

Referenced by [optimise\(\)](#).

42.24.3.9 bool RMOL::RMOL_Service::optimise (stdair::FlightDate & *ioFlightDate*, const stdair::DateTime_T & *iRMEventTime*, const stdair::ForecastingMethod & *iForecastingMethod*, const stdair::PartnershipTechnique & *iPartnershipTechnique*)

Optimise (revenue management) an flight-date/network-date

Definition at line 399 of file [RMOL_Service.cpp](#).

References [forecastOnD\(\)](#), [RMOL::Forecaster::forecastUsingAdditivePickUp\(\)](#), [RMOL::Forecaster::forecastUsingMultiplePickUp\(\)](#), [optimiseOnD\(\)](#), [optimiseOnDUsingAdvancedRMCooperation\(\)](#), [optimiseOnDUsingRMCooperation\(\)](#), [projectAggregatedDemandOnLegCabins\(\)](#), [projectOnDDemandOnLegCabinsUsingDYP\(\)](#), [projectOnDDemandOnLegCabinsUsingYP\(\)](#), [resetDemandInformation\(\)](#), and [updateBidPrice\(\)](#).

42.24.3.10 void RMOL::RMOL_Service::forecastOnD (const stdair::DateTime_T & *iRMEventTime*)

[Forecaster](#)

Definition at line 500 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#), and [getYieldFeatures\(\)](#).

Referenced by [optimise\(\)](#).

42.24.3.11 stdair::YieldFeatures * RMOL::RMOL_Service::getYieldFeatures (const stdair::OnDDate & *iOnDDate*, const stdair::CabinCode_T & *iCabinCode*, stdair::BomRoot & *iBomRoot*)

Definition at line 573 of file [RMOL_Service.cpp](#).

Referenced by [forecastOnD\(\)](#).

42.24.3.12 void RMOL::RMOL_Service::forecastOnD (const stdair::YieldFeatures & *iYieldFeatures*, stdair::OnDDate & *iOnDDate*, const stdair::CabinCode_T & *iCabinCode*, const stdair::DTD_T & *iDTD*, stdair::BomRoot & *iBomRoot*)

Definition at line 646 of file [RMOL_Service.cpp](#).

References [setOnDForecast\(\)](#).

42.24.3.13 void RMOL::RMOL_Service::setOnDForecast (const stdair::AirlineClassList & *iAirlineClassList*, const stdair::MeanValue_T & *iMeanValue*, const stdair::StdDevValue_T & *iStdDevValue*, stdair::OnDDate & *iOnDDate*, const stdair::CabinCode_T & *iCabinCode*, stdair::BomRoot & *iBomRoot*)

Definition at line 761 of file [RMOL_Service.cpp](#).

Referenced by [forecastOnD\(\)](#).

```
42.24.3.14 void RMOL::RMOL_Service::setOnDForecast ( const stdair::AirlineCode_T &
            iAirlineCode, const stdair::Date_T & iDepartureDate, const stdair::AirportCode_T &
            iOrigin, const stdair::AirportCode_T & iDestination, const stdair::CabinCode_T &
            iCabinCode, const stdair::ClassCode_T & iClassCode, const stdair::MeanValue_T &
            iMeanValue, const stdair::StdDevValue_T & iStdDevValue, const stdair::Yield_T &
            iYield, stdair::BomRoot & iBomRoot )
```

Definition at line 820 of file [RMOL_Service.cpp](#).

```
42.24.3.15 void RMOL::RMOL_Service::setOnDForecast ( const stdair::AirlineCodeList_T
            & iAirlineCodeList, const stdair::AirlineCode_T & iAirlineCode, const
            stdair::Date_T & iDepartureDate, const stdair::AirportCode_T & iOrigin, const
            stdair::AirportCode_T & iDestination, const stdair::CabinCode_T & iCabinCode,
            const stdair::ClassCodeList_T & iClassCodeList, const stdair::MeanValue_T &
            iMeanValue, const stdair::StdDevValue_T & iStdDevValue, const stdair::Yield_T &
            iYield, stdair::BomRoot & iBomRoot )
```

Definition at line 882 of file [RMOL_Service.cpp](#).

```
42.24.3.16 void RMOL::RMOL_Service::resetDemandInformation ( const stdair::DateTime_T &
            iRMEventTime )
```

Definition at line 997 of file [RMOL_Service.cpp](#).

Referenced by [optimise\(\)](#), [optimiseOnDUsingAdvancedRMCooperation\(\)](#), and [optimiseOnDUsingRMCooperation\(\)](#).

```
42.24.3.17 void RMOL::RMOL_Service::resetDemandInformation ( const stdair::DateTime_T &
            iRMEventTime, const stdair::Inventory & iInventory )
```

Definition at line 1023 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#).

```
42.24.3.18 void RMOL::RMOL_Service::projectAggregatedDemandOnLegCabins ( const
            stdair::DateTime_T & iRMEventTime )
```

Definition at line 1071 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#).

Referenced by [optimise\(\)](#).

```
42.24.3.19 void RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingYP ( const
            stdair::DateTime_T & iRMEventTime )
```

Definition at line 1176 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#).

Referenced by [optimise\(\)](#).

42.24.3.20 void RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingDA (const stdair::DateTime_T & *iRMEventTime*)

Definition at line 1451 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#).

42.24.3.21 void RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingDYP (const stdair::DateTime_T & *iRMEventTime*)

Definition at line 1607 of file [RMOL_Service.cpp](#).

Referenced by [optimise\(\)](#), [optimiseOnDUsingAdvancedRMCooperation\(\)](#), and [optimiseOnDUsingRMCooperation\(\)](#).

42.24.3.22 void RMOL::RMOL_Service::projectOnDDemandOnLegCabinsUsingDYP (const stdair::DateTime_T & *iRMEventTime*, const stdair::Inventory & *ilInventory*)

Definition at line 1633 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#).

42.24.3.23 void RMOL::RMOL_Service::optimiseOnD (const stdair::DateTime_T & *iRMEventTime*)

[Optimiser](#)

Definition at line 1275 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#), and [RMOL::Optimiser::optimiseUsingOnDForecast\(\)](#).

Referenced by [optimise\(\)](#).

42.24.3.24 void RMOL::RMOL_Service::optimiseOnDUsingRMCooperation (const stdair::DateTime_T & *iRMEventTime*)

Definition at line 1749 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#), [RMOL::Optimiser::optimiseUsingOnDForecast\(\)](#), [projectOnDDemandOnLegCabinsUsingDYP\(\)](#), and [resetDemandInformation\(\)](#).

Referenced by [optimise\(\)](#).

42.24.3.25 void RMOL::RMOL_Service::optimiseOnDUsingAdvancedRMCooperation (const stdair::DateTime_T & *iRMEventTime*)

Definition at line 1809 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#), [RMOL::Optimiser::optimiseUsingOnDForecast\(\)](#), [projectOnDDemandOnLegCabinsUsingDYP\(\)](#), [resetDemandInformation\(\)](#), and [update-BidPrice\(\)](#).

Referenced by [optimise\(\)](#).

42.24.3.26 void RMOL::RMOL_Service::updateBidPrice (const stdair::DateTime_T & *iRMEventTime*)

Definition at line 1324 of file [RMOL_Service.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#).

Referenced by [optimise\(\)](#), and [optimiseOnDUsingAdvancedRMCooperation\(\)](#).

42.24.3.27 void RMOL::RMOL_Service::updateBidPrice (const stdair::FlightDate & *iFlightDate*, stdair::BomRoot & *iBomRoot*)

Definition at line 1372 of file [RMOL_Service.cpp](#).

42.24.3.28 std::string RMOL::RMOL_Service::jsonExport (const stdair::AirlineCode_T & , const stdair::FlightNumber_T & , const stdair::Date_T & *iDepartureDate*) const

Recursively dump, in the returned string and in JSON format, the flight-date corresponding to the parameters given as input.

Parameters

<i>const</i> stdair::AirlineCode_T	& Airline code of the flight to dump.
<i>const</i> stdair::FlightNumber_T	& Flight number of the flight to dump.
<i>const</i> stdair::Date_T	& Departure date of a flight to dump.

Returns

std::string Output string in which the BOM tree is JSON-ified.

42.24.3.29 std::string RMOL::RMOL_Service::csvDisplay () const

Recursively display (dump in the returned string) the objects of the BOM tree.

Returns

std::string Output string in which the BOM tree is logged/dumped.

The documentation for this class was generated from the following files:

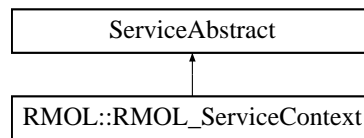
- [rmol/RMOL_Service.hpp](#)
- [rmol/service/RMOL_Service.cpp](#)

42.25 RMOL::RMOL_ServiceContext Class Reference

Inner class holding the context for the [RMOL](#) Service object.

```
#include <rmol/service/RMOL_ServiceContext.hpp>
```

Inheritance diagram for RMOL::RMOL_ServiceContext:



Friends

- class [RMOL_Service](#)
- class [FacRmolServiceContext](#)

42.25.1 Detailed Description

Inner class holding the context for the [RMOL](#) Service object.

Definition at line 29 of file [RMOL_ServiceContext.hpp](#).

42.25.2 Friends And Related Function Documentation

42.25.2.1 friend class [RMOL_Service](#) [friend]

The [RMOL_Service](#) class should be the sole class to get access to ServiceContext content: general users do not want to bother with a context interface.

Definition at line 35 of file [RMOL_ServiceContext.hpp](#).

42.25.2.2 friend class [FacRmolServiceContext](#) [friend]

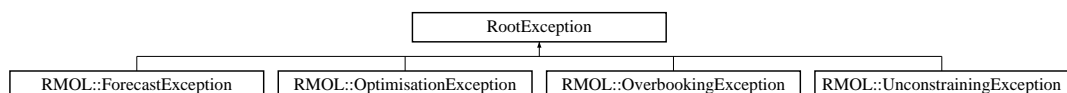
Definition at line 36 of file [RMOL_ServiceContext.hpp](#).

The documentation for this class was generated from the following files:

- [rmol/service/RMOL_ServiceContext.hpp](#)
- [rmol/service/RMOL_ServiceContext.cpp](#)

42.26 RootException Class Reference

Inheritance diagram for RootException:

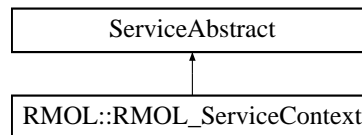


The documentation for this class was generated from the following file:

- [rmol/RMOL_Types.hpp](#)

42.27 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract:

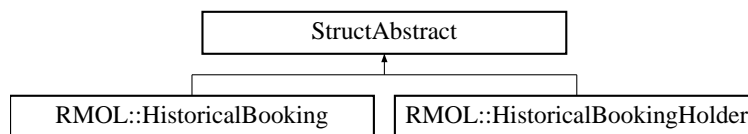


The documentation for this class was generated from the following file:

- [rmol/service/RMOL_ServiceContext.hpp](#)

42.28 StructAbstract Class Reference

Inheritance diagram for StructAbstract:

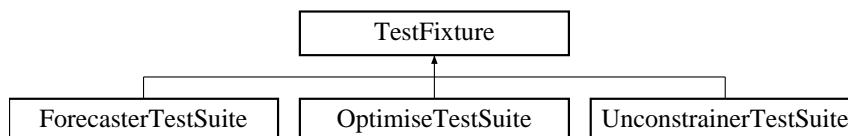


The documentation for this class was generated from the following file:

- [rmol/bom/HistoricalBooking.hpp](#)

42.29 TestFixture Class Reference

Inheritance diagram for TestFixture:



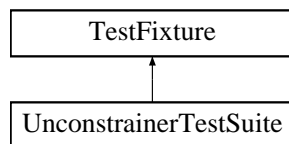
The documentation for this class was generated from the following file:

- [test/rmol/OptimiseTestSuite.hpp](#)

42.30 UnconstrainerTestSuite Class Reference

```
#include <test/rmol/UnconstrainerTestSuite.hpp>
```

Inheritance diagram for UnconstrainerTestSuite:



Public Member Functions

- void [testUnconstrainingByEM](#) ()
- [UnconstrainerTestSuite](#) ()

Protected Attributes

- std::stringstream [_describeKey](#)

42.30.1 Detailed Description

Definition at line 6 of file [UnconstrainerTestSuite.hpp](#).

42.30.2 Constructor & Destructor Documentation

42.30.2.1 UnconstrainerTestSuite::UnconstrainerTestSuite ()

Constructor.

42.30.3 Member Function Documentation

42.30.3.1 void UnconstrainerTestSuite::testUnconstrainingByEM ()

Test data unconstraining by Expectation Maximization.

42.30.4 Member Data Documentation

42.30.4.1 std::stringstream UnconstrainerTestSuite::_describeKey [protected]

Definition at line 19 of file [UnconstrainerTestSuite.hpp](#).

The documentation for this class was generated from the following file:

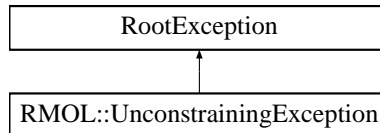
- test/rmol/[UnconstrainerTestSuite.hpp](#)

42.31 RMOL::UnconstrainingException Class Reference

Unconstraining-related exception.

```
#include <rmol/RMOL_Types.hpp>
```

Inheritance diagram for RMOL::UnconstrainingException:



Public Member Functions

- [UnconstrainingException](#) (const std::string &iWhat)

42.31.1 Detailed Description

Unconstraining-related exception.

Definition at line 41 of file [RMOL_Types.hpp](#).

42.31.2 Constructor & Destructor Documentation

42.31.2.1 RMOL::UnconstrainingException::UnconstrainingException (const std::string &iWhat) [inline]

Constructor.

Definition at line 44 of file [RMOL_Types.hpp](#).

The documentation for this class was generated from the following file:

- [rmol/RMOL_Types.hpp](#)

42.32 RMOL::Utilities Class Reference

```
#include <rmol/bom/Utilities.hpp>
```

Static Public Member Functions

- static void [computeDistributionParameters](#) (const [UnconstrainedDemandVector_T](#) &, double &, double &)
- static stdair::DCPList_T [buildRemainingDCPList](#) (const stdair::DTD_T &)
- static stdair::DCPList_T [buildRemainingDCPList2](#) (const stdair::DTD_T &)

- static stdair::NbOfSegments_T [getNbOfDepartedSimilarSegments](#) (const stdair::SegmentCabin &, const stdair::Date_T &)

42.32.1 Detailed Description

Class holding helper methods.

Definition at line 19 of file [Utilities.hpp](#).

42.32.2 Member Function Documentation

42.32.2.1 void RMOL::Utilities::computeDistributionParameters (const UnconstrainedDemandVector_T & *iVector*, double & *ioMean*, double & *ioStdDev*) [static]

Compute the mean and the standard deviation from a set of samples.

Definition at line 24 of file [Utilities.cpp](#).

42.32.2.2 stdair::DCPList_T RMOL::Utilities::buildRemainingDCPList (const stdair::DTD_T & *iDTD*) [static]

Build the list of remaining DCP's for the segment-date.

Definition at line 55 of file [Utilities.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#).

Referenced by [RMOL::Forecaster::forecastUsingAdditivePickUp\(\)](#).

42.32.2.3 stdair::DCPList_T RMOL::Utilities::buildRemainingDCPList2 (const stdair::DTD_T & *iDTD*) [static]

Definition at line 80 of file [Utilities.cpp](#).

References [RMOL::DEFAULT_DCP_LIST](#).

Referenced by [RMOL::Forecaster::forecastUsingAdditivePickUp\(\)](#).

42.32.2.4 stdair::NbOfSegments_T RMOL::Utilities::getNbOfDepartedSimilarSegments (const stdair::SegmentCabin & *iSegmentCabin*, const stdair::Date_T & *iEventDate*) [static]

Retrieve the number of departed similar segments.

Definition at line 105 of file [Utilities.cpp](#).

References [RMOL::GuillotineBlockHelper::getNbOfSegmentAlreadyPassedThisDTD\(\)](#).

The documentation for this class was generated from the following files:

- [rmol/bom/Utilities.hpp](#)
- [rmol/bom/Utilities.cpp](#)

43 File Documentation

- 43.1 doc/local/authors.doc File Reference
- 43.2 doc/local/codingrules.doc File Reference
- 43.3 doc/local/copyright.doc File Reference
- 43.4 doc/local/documentation.doc File Reference
- 43.5 doc/local/features.doc File Reference
- 43.6 doc/local/help_wanted.doc File Reference
- 43.7 doc/local/howto_release.doc File Reference
- 43.8 doc/local/index.doc File Reference
- 43.9 doc/local/installation.doc File Reference
- 43.10 doc/local/linking.doc File Reference
- 43.11 doc/local/test.doc File Reference
- 43.12 doc/local/users_guide.doc File Reference
- 43.13 doc/local/verification.doc File Reference
- 43.14 doc/tutorial/bpsk.doc File Reference
- 43.15 doc/tutorial/convcode.doc File Reference
- 43.16 doc/tutorial/interleaver.doc File Reference
- 43.17 doc/tutorial/itfile.doc File Reference
- 43.18 doc/tutorial/ldpc_bersim_awgn.doc File Reference
- 43.19 doc/tutorial/ldpc_gen_codes.doc File Reference
- 43.20 doc/tutorial/matlab_itpp.doc File Reference
- 43.21 doc/tutorial/mimoconv.doc File Reference

43.22 doc/tutorial/mog.doc File Reference

43.23 doc/tutorial/qpsk_simulation.doc File Reference

43.24 doc/tutorial/rayleigh.doc File Reference

43.25 doc/tutorial/reedsolomon.doc File Reference

43.26 doc/tutorial/spread.doc File Reference

43.27 doc/tutorial/src/bpsk.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- int [main](#) ()

43.27.1 Function Documentation

43.27.1.1 int main ()

Definition at line 9 of file [bpsk.cpp](#).

43.28 bpsk.cpp

```
00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 //These lines are needed for use of cout and endl
00006 using std::cout;
00007 using std::endl;
00008
00009 int main()
00010 {
00011     //Scalars
00012     int N;
00013     double N0;
00014
00015     //Vectors
00016     bvec bits, dec_bits;
00017     vec symbols, rec;
00018
00019     //Classes
00020     BPSK bpsk; //The BPSK modulator/debodulator class
00021     BEREC berc; //The Bit Error Rate Counter class
00022
00023     //Init
00024     N = 500000; //The number of bits to simulate
00025     N0 = 1;     //0 dB SNR
```

```

00026
00027 //Randomize the random number generator
00028 RNG_randomize();
00029
00030 //Generate the bits:
00031 bits = randb(N);
00032
00033 //Do the BPSK modulation
00034 bpsk.modulate_bits(bits, symbols);
00035
00036 //Add the AWGN
00037 rec = symbols + sqrt(N0/2) * randn(N);
00038
00039 //Decode the received bits
00040 bpsk.demodulate_bits(rec, dec_bits);
00041
00042 //Count the number of errors
00043 berc.count(bits, dec_bits);
00044
00045 //Print the results
00046 cout << "There were " << berc.get_errors() << " received bits in error." << endl;
00047 cout << "There were " << berc.get_corrects() << " correctly received bits." << endl;
00048 cout << "The error probability was " << berc.get_errorrates() << endl;
00049 cout << "The theoretical error probability is " << 0.5*erfc(1.0) << endl;
00050
00051 //Exit program:
00052 return 0;
00053
00054 }

```

43.29 doc/tutorial/src/convcode.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- `int main ()`

43.29.1 Function Documentation

43.29.1.1 `int main ()`

Definition at line 9 of file [convcode.cpp](#).

43.30 convcode.cpp

```

00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 //These lines are needed for use of cout and endl
00006 using std::cout;

```

```

00007 using std::endl;
00008
00009 int main()
00010 {
00011     //Scalars
00012     int constraint_length, MaxNrofErrors, Nobits, MaxIterations, p, i;
00013     double Ec, Eb;
00014
00015     //Vectors
00016     ivec generators;
00017     vec EbN0dB, EbN0, N0, ber, trans_symbols, rec_symbols;
00018     bvec uncoded_bits, coded_bits, decoded_bits;
00019
00020     //Classes
00021     BPSK bpsk;
00022     BERC berc;
00023     Convolutional_Code conv_code;
00024     AWGN_Channel channel;
00025
00026     /*
00027     Set up the convolutional encoder/decoder class:
00028     The generators are given in octal form by adding a zero in front of the numbers
00029     .
00029     In this example we will simulate a rate 1/3 code that is listed in J. G. Proakis,
00030     "Digital communications". The encoder has constraint length 7.
00031     */
00032     generators.set_size(3, false);
00033     generators(0) = 0133;
00034     generators(1) = 0145;
00035     generators(2) = 0175;
00036     constraint_length = 7;
00037     conv_code.set_generator_polynomials(generators, constraint_length);
00038
00039     //Init: Calculate some simulation specific parameters:
00040     Ec = 1.0;
00041     EbN0dB = linspace(-2, 6, 5);
00042     EbN0 = inv_dB(EbN0dB);
00043     Eb = Ec / conv_code.get_rate();
00044     N0 = Eb * pow(EbN0, -1);
00045     MaxNrofErrors = 100;
00046     Nobits = 10000;
00047     MaxIterations = 10;
00048     ber.set_size(EbN0dB.length(), false);
00049     ber.clear();
00050
00051     //Randomize the random number generators.
00052     RNG_randomize();
00053
00054     for (p=0; p<EbN0dB.length(); p++) {
00055
00056         cout << "Now simulating point " << p+1 << " out of " << EbN0dB.length() << endl;
00057         berc.clear(); //Clear the bit error rate counter.
00058         channel.set_noise(N0(p)/2.0); //Set the noise value of the AWGN channel.
00059
00060         for (i=0; i<MaxIterations; i++) {
00061
00062             uncoded_bits = randb(Nobits); //The uncoded bits.
00063             coded_bits = conv_code.encode(uncoded_bits); //The convolutional encoder
00064             function.
00065             bpsk.modulate_bits(coded_bits, trans_symbols); //The BPSK modulator.

```



```

00065         rec_symbols = channel( trans_symbols );           //The AWGN channel.
00066         decoded_bits = conv_code.decode(rec_symbols);      //The Viterbi decoder funct
ion.
00067         berc.count(uncoded_bits,decoded_bits);            //Count the errors.
00068         ber(p) = berc.get_errorrate();
00069
00070         //Break the simulation on this point if sufficient number of bit errors wer
e observed:
00071         if (berc.get_errors()>MaxNrofErrors) {
00072             cout << "Breaking on point " << p+1 << " with " << berc.get_errors() << "
errors." << endl; break;
00073         }
00074
00075     }
00076 }
00077
00078 //Print the results:
00079 cout << "BER    = " << ber  << endl;
00080 cout << "EbN0dB = " << EbN0dB << endl;
00081
00082 //Exit program:
00083 return 0;
00084
00085 }

```

43.31 doc/tutorial/src/interleaver.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- int [main](#) ()

43.31.1 Function Documentation

43.31.1.1 int main ()

Definition at line 9 of file [interleaver.cpp](#).

43.32 interleaver.cpp

```

00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 //These lines are needed for use of cout and endl
00006 using std::cout;
00007 using std::endl;
00008
00009 int main()
00010 {
00011     //Declare scalars and vectors:
00012     int rows, cols;
00013     ivec input, output, deinterleaved;

```

```

00014
00015 //Declare the interleaver. The interleaver classes are templated, and therefore
    we must specify
00016 //the type of the data elements. In this example we are using integers:
00017 Block_Interleaver<int> my_interleaver;
00018
00019 //Initialize the interleaver class. Note that this can be done already in the d
    eclaration by writing
00020 //Block_Interleaver<int> my_interleaver(rows,cols);
00021 rows = 4;
00022 cols = 5;
00023 my_interleaver.set_rows(rows);
00024 my_interleaver.set_cols(cols);
00025
00026 //Define the input to the interleaver:
00027 input = "1:20";
00028
00029 //Do the interleaving:
00030 output = my_interleaver.interleave(input);
00031
00032 //Do the de-interleaving:
00033 deinterleaved = my_interleaver.deinterleave(output);
00034
00035 //Print the results:
00036 cout << "input = " << input << endl;
00037 cout << "output = " << output << endl;
00038 cout << "deinterleaved = " << deinterleaved << endl;
00039
00040 //Exit program:
00041 return 0;
00042
00043 }

```

43.33 doc/tutorial/src/ldpc_bersim_awgn.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- int [main](#) (int argc, char **argv)

43.33.1 Function Documentation

43.33.1.1 int main (int argc, char ** argv)

Definition at line 6 of file [ldpc_bersim_awgn.cpp](#).

43.34 ldpc_bersim_awgn.cpp

```

00001 #include <itpp/itcomm.h>
00002
00003 using namespace std;
00004 using namespace itpp;
00005

```

```

00006 extern int main(int argc, char **argv)
00007 {
00008     int Nbits=1000*1000*5000;           // maximum number of bits simulated for any
    SNR point
00009     int Nbers=2000;                     // target number of bit errors per SNR point
00010     double BERmin=1e-6;                 // BER at which to terminate simulation
00011     vec EbN0db = "0.6:0.2:5";
00012
00013     LDPC_Code C(argv[1]);
00014     bool single_snr_mode=false;
00015     if (argc==3) {
00016         double x;
00017         sscanf(argv[2], "%lf", &x);
00018         EbN0db.set_size(1);
00019         EbN0db(0)=x;
00020         single_snr_mode=true;
00021     }
00022
00023     cout << "Running with Eb/N0: " << EbN0db << endl;
00024
00025     // High performance: 2500 iterations, high resolution LLR algebra
00026     C.setup_decoder("bp", "2500 1 0");
00027
00028     // Alt. setting -- High speed: 50 iterations, logmax approximation
00029     // C.setup_decoder("bp", "50 1 0", LLR_calc_unit(12,0,7));
00030
00031     cout << C << endl;
00032
00033     int N = C.get_nvar();                 // number of bits per codeword
00034     BPSK Mod;
00035     bvec bitsin = zeros_b(N);
00036     vec s = Mod.modulate_bits(bitsin);
00037
00038     RNG_randomize();
00039     for (int j=0; j<length(EbN0db); j++) {
00040         // noise variance is N0/2 per dimension
00041         double N0 = pow(10.0, -EbN0db(j)/10.0) / C.get_rate();
00042         AWGN_Channel chan(N0/2);
00043         BERc berc; // counters for coded and uncoded BER
00044         BLERC ferc; // counter for coded FER
00045         ferc.set_blocksize(N);
00046         for (long int i=0; i<Nbits; i+=C.get_nvar()) {
00047             // Received data
00048             vec x = chan(s);
00049
00050             // Demodulate
00051             vec softbits=Mod.demodulate_soft_bits(x, N0);
00052
00053             //Decode the received bits
00054             bvec bitsout=C.decode(softbits);
00055
00056             //Count the number of errors
00057             berc.count(bitsin, bitsout);
00058             ferc.count(bitsin, bitsout);
00059
00060             if (single_snr_mode) {
00061                 cout << "Eb/N0=" << EbN0db(j) << " Simulated "
00062                     << ferc.get_total_blocks() << " frames and "
00063                     << berc.get_total_bits() << " bits. "
00064                     << "Obtained " << berc.get_errors() << " bit errors. "
00065                     << " BER: " << berc.get_errrorate()
00066                     << " FER: " << ferc.get_errrorate() << endl;

```

```

00067         cout.flush();
00068     } else {
00069         if (berc.get_errors()>Nbers) { break;}
00070     }
00071 }
00072
00073 cout << "Eb/N0=" << EbN0db(j) << " Simulated "
00074 << ferc.get_total_blocks() << " frames and "
00075 << berc.get_total_bits() << " bits. "
00076 << "Obtained " << berc.get_errors() << " bit errors. "
00077 << " BER: " << berc.get_errorrate()
00078 << " FER: " << ferc.get_errorrate() << endl;
00079 cout.flush();
00080 if (berc.get_errorrate()<BERmin) { break; }
00081 }
00082 return 0;
00083 }

```

43.35 doc/tutorial/src/ldpc_gen_codes.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- `int main (int argc, char **argv)`

43.35.1 Function Documentation

43.35.1.1 `int main (int argc, char ** argv)`

Definition at line 8 of file `ldpc_gen_codes.cpp`.

43.36 ldpc_gen_codes.cpp

```

00001 // Generate some example LDPC codes
00002
00003 #include <itpp/itcomm.h>
00004
00005 using namespace itpp;
00006 using namespace std;
00007
00008 extern int main(int argc, char **argv)
00009 {
00010     { // This generates a random regular (3,6) code with 500 bits
00011         cout << "===== RANDOM (3,6) CODE =====" << endl;
00012         LDPC_Parity_Matrix H;
00013         H.generate_regular_ldpc(500,3,6,
00014                                "rand", // random unstructured matrix
00015                                "500 10"); // optimize girth
00016         H.display_stats();
00017         LDPC_Code C1(H);
00018         C1.save_to_file("random_3_6_code.it");
00019     }
00020 }

```

```

00021 { // This is the code "204.33.484 (N=204,K=102,M=102,R=0.5)" from
00022 // David MacKay's database over sparse-graph code. It can be
00023 // obtained with "wget
00024 // http://www.inference.phy.cam.ac.uk/mackay/codes/EN/C/204.33.484"
00025 cout << "===== MACKAY CODE =====" << endl;
00026 LDPC_Parity_Matrix H("204.33.484","alist");
00027 H.display_stats();
00028 LDPC_Generator_Matrix G(H);
00029 LDPC_Code C(H,G);
00030 C.save_to_file("mackay_204.33.484.it");
00031
00032 // Now produce a girth-optimized version of this code by removing
00033 // cycles. This slightly improves the performance at high SNR.
00034 H.cycle_removal_MGW(12);
00035 LDPC_Generator_Matrix G1(H);
00036 LDPC_Code C1(H,G1);
00037 C1.save_to_file("mackay_204.33.484_opt.it");
00038 }
00039
00040 // Irregular 1/2-rate codes optimized for the AWGN channel. The
00041 // degree distributions are taken from Richardson & Urbanke,
00042 // Trans. IT 2001.
00043
00044 { // 1000 bits
00045 cout << "===== IRREGULAR CODE 1000 BITS =====" << endl;
00046 LDPC_Parity_Matrix H;
00047 H.generate_irregular_ldpc(1000,
00048                          "0 0.27684 0.28342 0 0 0 0 0 0.43974",
00049                          "0 0 0 0 0 0.01568 0.85244 0.13188",
00050                          "rand", // random unstructured matrix
00051                          "500 8"); // optimize girth
00052 LDPC_Code C(H);
00053 C.save_to_file("RU_1000.it");
00054 }
00055
00056 { // 10000 bits (takes a few minutes to run)
00057 cout << "===== IRREGULAR CODE 10000 BITS =====" << endl;
00058 LDPC_Parity_Matrix H;
00059 H.generate_irregular_ldpc(10000,"0 0.21991 0.23328 0.02058 0 0.08543 0.06540
00060 0.04767 \
00061                          0.01912 0 0 0 0 0 0 0 0 0.08064 0.22798",
00062                          "0 0 0 0 0 0 0 0.64854 0.34747 0.00399",
00063                          "rand", // random unstructured matrix
00064                          "150 8"); // optimize
00065 LDPC_Code C(H);
00066 C.save_to_file("RU_10000.it");
00067 }
00068
00069 { // 100000 bits (takes a while to run)
00070 cout << "===== IRREGULAR CODE 100000 BITS =====" << endl;
00071 LDPC_Parity_Matrix H;
00072 H.generate_irregular_ldpc(100000,"0 0.1712 0.21053 0.00273 0 0 0.00009 0.1526
00073 9 0.09227 \
00074                          0.02802 0 0 0 0 0.01206 0 0 0 0 0 0 0 0 0 0 0 0
00075 0.07212 0 0 0 0 \
00076                          0 0 0 0 0 0 0 0 0 0 0 0 0 0.25830",
00077                          "0 0 0 0 0 0 0 0 0.33620 0.08883 0.57497",
00078                          "rand",
00079                          "40 4"); // less aggressive optimization
00080 LDPC_Code C(H);
00081 C.save_to_file("RU_100000.it");
00082 }

```

```

00080
00081     exit(0);
00082
00083     { // 1000000 bits (THIS CODE REQUIRES ABOUT 450 MB TO STORE AND 2GB
00084       // INTERNAL MEMORY TO GENERATE)
00085       cout << "===== IRREGULAR CODE 1000000 BITS =====" << endl;
00086       LDPC_Parity_Matrix H;
00087       H.generate_irregular_ldpc(1000000, "0 0.1712 0.21053 0.00273 0 0 0.00009 0.152
69 0.09227 \
00088                                     0.02802 0 0 0 0 0.01206 0 0 0 0 0 0 0 0 0 0 0
00089       0.07212 0 0 0 0 \
00089                                     0 0 0 0 0 0 0 0 0 0 0 0 0 0.25830",
00090                                     "0 0 0 0 0 0 0 0 0.33620 0.08883 0.57497",
00091                                     "rand",
00092                                     "0 0"); // no optimization here
00093       LDPC_Code C(H);
00094       C.save_to_file("RU_1000000.it");
00095     }
00096
00097 }

```

43.37 doc/tutorial/src/mimoconv.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- void [ZF_demod](#) (ND_UQAM &channel, ivec &LLR_apr, ivec &LLR_apost, double sigma2, cmat &H, cvec &y)
- int [main](#) (int argc, char **argv)

43.37.1 Function Documentation

43.37.1.1 void [ZF_demod](#) (ND_UQAM & *channel*, ivec & *LLR_apr*, ivec & *LLR_apost*, double *sigma2*, cmat & *H*, cvec & *y*)

Definition at line 18 of file [mimoconv.cpp](#).

Referenced by [main\(\)](#).

43.37.1.2 int [main](#) (int *argc*, char ** *argv*)

Definition at line 32 of file [mimoconv.cpp](#).

References [ZF_demod\(\)](#).

43.38 mimoconv.cpp

```

00001
00002 #include <itpp/itcomm.h>
00003
00004 using std::cout;
00005 using std::endl;

```

```

00006 using namespace itpp;
00007 using namespace std;
00008
00009 /* Zero-forcing detector with ad hoc soft information. This function
00010    applies the ZF (pseudoinverse) linear filter to the received
00011    data. This results in effective noise with covariance matrix
00012     $\text{inv}(H'H) * \sigma^2$ . The diagonal elements of this noise covariance
00013    matrix are taken as noise variances per component in the processed
00014    received data but the noise correlation is ignored.
00015
00016    */
00017
00018 void ZF_demod(ND_UQAM &channel, ivec &LLR_apr, ivec &LLR_apost, double sigma2, c
mat &H, cvec &y)
00019 {
00020     it_assert(H.rows()>=H.cols(), "ZF_demod() - underdetermined systems not tolerate
d");
00021     cvec shat=ls_solve_od(H,y); // the ZF solution
00022     vec Sigma2=real(diag(inv(H.hermitian_transpose()*H)))*sigma2; // noise covaria
nce of shat
00023     cvec h(length(shat));
00024     for (int i=0; i<length(shat); i++) {
00025         shat(i) = shat(i)/sqrt(Sigma2(i));
00026         h(i) = 1.0/sqrt(Sigma2(i));
00027     }
00028     channel.map_demod(LLR_apr, LLR_apost, 1.0, h, shat);
00029 }
00030
00031
00032 extern int main(int argc, char **argv)
00033 {
00034     // -- modulation and channel parameters (taken from command line input) --
00035     int nC; // type of constellation (1=QPSK, 2=16-QAM, 3=64-QA
M)
00036     int nRx; // number of receive antennas
00037     int nTx; // number of transmit antennas
00038     int Tc; // coherence time (number of channel vectors with sa
me H)
00039
00040     if (argc!=5) {
00041         cout << "Usage: cm nTx nRx nC Tc" << endl << "Example: cm 2 2 1 100000 (2x2 Q
PSK MIMO on slow fading channel)" << endl;
00042         exit(1);
00043     } else {
00044         sscanf(argv[1], "%i", &nTx);
00045         sscanf(argv[2], "%i", &nRx);
00046         sscanf(argv[3], "%i", &nC);
00047         sscanf(argv[4], "%i", &Tc);
00048     }
00049
00050     cout << "Initializing.. " << nTx << " TX antennas, " << nRx << " RX antennas, "
<< (1<nC) << "-PAM per dimension, coherence time " << Tc << endl;
00051
00052
00053     // -- simulation control parameters --
00054     const vec EbN0db = "-5:0.5:50"; // SNR range
00055     const int Nmethods =2; // number of demodulators to try
00056     const long int Nbitsmax=50*1000*1000; // maximum number of bits to ever simula
te per SNR point
00057     const int Nu = 1000; // length of data packet (before applyin
g channel coding)
00058

```

```

00059  int Nbers, Nfers;                // target number of bit/frame errors per SNR poi
                                nt
00060  double BERmin, FERmin;           // BER/FER at which to terminate simulation
00061  if (Tc==1) {                    // Fast fading channel, BER is of primary interest
00062      BERmin = 0.001;              // stop simulating a given method if BER<this value
00063      FERmin = 1.0e-10;           // stop simulating a given method if FER<this value
00064      Nbers = 1000;               // move to next SNR point after counting 1000 bit errors

00065      Nfers = 200;                // do not stop on this condition
00066  } else {                         // Slow fading channel, FER is of primary interest here
00067      BERmin = 1.0e-15;           // stop simulating a given method if BER<this value
00068      FERmin = 0.01;              // stop simulating a given method if FER<this value
00069      Nbers = -1;                 // do not stop on this condition
00070      Nfers = 200;               // move to next SNR point after counting 200 frame error
                                s
00071  }
00072
00073  // -- Channel code parameters --
00074  Convolutional_Code code;
00075  ivec generator(3);
00076  generator(0)=0133; // use rate 1/3 code
00077  generator(1)=0165;
00078  generator(2)=0171;
00079  double rate=1.0/3.0;
00080  code.set_generator_polynomials(generator, 7);
00081  bvec dummy;
00082  code.encode_tail(randb(Nu), dummy);
00083  const int Nc = length(dummy);    // find out how long the coded blocks are
00084
00085  // ===== Initialize =====
00086
00087  const int Nctx = (int) (2*nC*nTx*ceil(double(Nc)/double(2*nC*nTx))); // Total
                                number of bits to transmit
00088  const int Nvec = Nctx/(2*nC*nTx); // Number of channel vectors to tran
                                smit
00089  const int Nbitspvec = 2*nC*nTx;  // Number of bits per channel vector

00090
00091  // initialize MIMO channel with uniform QAM per complex dimension and Gray codi
                                ng
00092  ND_UQAM chan;
00093  chan.set_Gray_QAM(nTx, 1<<(2*nC));
00094  cout << chan << endl;
00095
00096  // initialize interleaver
00097  Sequence_Interleaver<bin> sequence_interleaver_b(Nctx);
00098  Sequence_Interleaver<int> sequence_interleaver_i(Nctx);
00099  sequence_interleaver_b.randomize_interleaver_sequence();
00100  sequence_interleaver_i.set_interleaver_sequence(sequence_interleaver_b.get_inte
                                rleaver_sequence());
00101
00102  // RNG_randomize();
00103
00104  Array<cvec> Y(Nvec);              // received data
00105  Array<cmat> H(Nvec/Tc+1);        // channel matrix (new matrix for each coherence in
                                terval)
00106
00107  ivec Contflag = ones_i(Nmethods); // flag to determine whether to run a given
                                demodulator
00108  if (pow(2.0, nC*2.0*nTx)>256) {  // ML decoder too complex..
00109      Contflag(1)=0;
00110  }

```



```

00111  if (nTx>nRx) {
00112      Contflag(0)=0; // ZF not for underdetermined systems
00113  }
00114  cout << "Running methods: " << Contflag << endl;
00115
00116  cout.setf(ios::fixed, ios::floatfield);
00117  cout.setf(ios::showpoint);
00118  cout.precision(5);
00119
00120  // ===== Run simulation =====
00121  for (int nsnr=0; nsnr<length(EbN0db); nsnr++) {
00122      const double Eb=1.0; // transmitted energy per information bit
00123      const double N0 = pow(10.0,-EbN0db(nsnr)/10.0);
00124      const double sigma2=N0; // Variance of each scalar complex noise sample
00125      const double Es=rate*2*nC*Eb; // Energy per complex scalar symbol
00126      // (Each transmitted scalar complex symbol contains rate*2*nC
00127      // information bits.)
00128      const double Ess=sqrt(Es);
00129
00130      Array<BERC> berc(Nmethods); // counter for coded BER
00131      Array<BERC> bercu(Nmethods); // counter for uncoded BER
00132      Array<BLERC> ferc(Nmethods); // counter for coded FER
00133
00134      for (int i=0; i<Nmethods; i++) {
00135          ferc(i).set_blocksize(Nu);
00136      }
00137
00138      long int nbits=0;
00139      while (nbits<Nbitsmax) {
00140          nbits += Nu;
00141
00142          // generate and encode random data
00143          bvec inputbits = randb(Nu);
00144          bvec txbits;
00145          code.encode_tail(inputbits, txbits);
00146          // coded block length is not always a multiple of the number of
00147          // bits per channel vector
00148          txbits=concat(txbits,randb(Nctx-Nc));
00149          txbits = sequence_interleaver_b.interleave(txbits);
00150
00151          // -- generate channel and data ----
00152          for (int k=0; k<Nvec; k++) {
00153              /* A complex valued channel matrix is used here. An
00154              alternative (with equivalent result) would be to use a
00155              real-valued (structured) channel matrix of twice the
00156              dimension.
00157              */
00158              if (k%Tc==0) { // generate a new channel realization every Tc inter
vals
00159                  H(k/Tc) = Ess*randn_c(nRx,nTx);
00160              }
00161
00162              // modulate and transmit bits
00163              bvec bitstamp = txbits(k*2*nTx*nC, (k+1)*2*nTx*nC-1);
00164              cvec x=chan.modulate_bits(bitstamp);
00165              cvec e=sqrt(sigma2)*randn_c(nRx);
00166              Y(k) = H(k/Tc)*x+e;
00167          }
00168
00169          // -- demodulate --
00170          Array<QLLRvec> LLRin(Nmethods);
00171          for (int i=0; i<Nmethods; i++) {

```

```

00172         LLRin(i) = zeros_i(Nctx);
00173     }
00174
00175     QLLRvec llr_apr = zeros_i(nC*2*nTx); // no a priori input to demodulator
00176     QLLRvec llr_apost = zeros_i(nC*2*nTx);
00177     for (int k=0; k<Nvec; k++) {
00178         // zero forcing demodulation
00179         if (Contflag(0)) {
00180             ZF_demod(chan, llr_apr, llr_apost, sigma2, H(k/Tc), Y(k));
00181             LLRin(0).set_subvector(k*Nbitspvec, (k+1)*Nbitspvec-1, llr_apost);
00182         }
00183
00184         // ML demodulation
00185         if (Contflag(1)) {
00186             chan.map_demod(llr_apr, llr_apost, sigma2, H(k/Tc), Y(k));
00187             LLRin(1).set_subvector(k*Nbitspvec, (k+1)*Nbitspvec-1, llr_apost);
00188         }
00189     }
00190
00191     // -- decode and count errors --
00192     for (int i=0; i<Nmethods; i++) {
00193         bvec decoded_bits;
00194         if (Contflag(i)) {
00195             bercu(i).count(txbits(0, Nc-1), LLRin(i)(0, Nc-1) < 0); // uncoded BER
00196             LLRin(i) = sequence_interleaver_i.deinterleave(LLRin(i), 0);
00197             // QLLR values must be converted to real numbers since the convolutiona
00198             l decoder wants this
00199             vec llr = chan.get_llrcalc().to_double(LLRin(i).left(Nc));
00200             // llr = -llr; // UNCOMMENT THIS LINE IF COMPILING WITH 3.10.5 OR EA
00201             RLIER (BEFORE HARMONIZING LLR CONVENTIONS)
00202             code.decode_tail(llr, decoded_bits);
00203             berc(i).count(inputbits(0, Nu-1), decoded_bits(0, Nu-1)); // coded BER
00204             ferc(i).count(inputbits(0, Nu-1), decoded_bits(0, Nu-1)); // coded FER
00205         }
00206     }
00207
00208     /* Check whether it is time to terminate the simulation.
00209     Terminate when all demodulators that are still running have
00210     counted at least Nbers or Nfers bit/frame errors. */
00211     int minber = 1000000;
00212     int minfer = 1000000;
00213     for (int i=0; i<Nmethods; i++) {
00214         if (Contflag(i)) {
00215             minber = min(minber, round_i(bercu(i).get_errors()));
00216             minfer = min(minfer, round_i(ferc(i).get_errors()));
00217         }
00218     }
00219     if (Nbers > 0 && minber > Nbers) { break; }
00220     if (Nfers > 0 && minfer > Nfers) { break; }
00221 }
00222
00223     cout << "-----" << endl;
00224     cout << "Eb/N0: " << EbN0db(nsnr) << " dB. Simulated " << nbits << " bits." <
00225 < endl;
00226     cout << " Uncoded BER: " << bercu(0).get_errorrate() << " (ZF); " << berc
00227 u(1).get_errorrate() << " (ML)" << endl;
00228     cout << " Coded BER: " << berc(0).get_errorrate() << " (ZF); " << berc
00229 (1).get_errorrate() << " (ML)" << endl;
00230     cout << " Coded FER: " << ferc(0).get_errorrate() << " (ZF); " << ferc
00231 (1).get_errorrate() << " (ML)" << endl;
00232     cout.flush();
00233 }

```

```
00228     /* Check wheter it is time to terminate simulation. Stop when all
00229     methods have reached the min BER/FER of interest. */
00230     int contflag=0;
00231     for (int i=0; i<Nmethods; i++) {
00232         if (Contflag(i)) {
00233             if (berc(i).get_errorrate()>BERmin) { contflag=1; } else { Contflag(i)
= 0; }
00234             if (ferc(i).get_errorrate()>FERmin) { contflag=1; } else { Contflag(i)
= 0; }
00235         }
00236     }
00237     if (contflag) { continue; } else {break; }
00238 }
00239
00240 return 0;
00241 }
```

43.39 doc/tutorial/src/mog.cpp File Reference

```
#include <itpp/itstat.h>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <ios>
```

Functions

- int [main](#) ()

43.39.1 Function Documentation

43.39.1.1 int main ()

Definition at line 15 of file [mog.cpp](#).

43.40 mog.cpp

```
00001 #include <itpp/itstat.h>
00002
00003 #include <fstream>
00004 #include <iostream>
00005 #include <iomanip>
00006 #include <ios>
00007
00008 using std::cout;
00009 using std::endl;
00010 using std::fixed;
00011 using std::setprecision;
00012
00013 using namespace itpp;
```

```

00014
00015 int main() {
00016
00017     bool print_progress = false;
00018
00019     //
00020     // first, let's generate some synthetic data
00021
00022     int N = 100000; // number of vectors
00023     int D = 3;      // number of dimensions
00024     int K = 5;      // number of Gaussians
00025
00026     Array<vec> X(N); for(int n=0;n<N;n++) { X(n).set_size(D); X(n) = 0.0; }
00027
00028     // the means
00029
00030     Array<vec> mu(K);
00031     mu(0) = "-6, -4, -2";
00032     mu(1) = "-4, -2, 0";
00033     mu(2) = "-2, 0, 2";
00034     mu(3) = " 0, +2, +4";
00035     mu(4) = "+2, +4, +6";
00036
00037
00038     // the diagonal variances
00039
00040     Array<vec> var(K);
00041     var(0) = "0.1, 0.2, 0.3";
00042     var(1) = "0.2, 0.3, 0.1";
00043     var(2) = "0.3, 0.1, 0.2";
00044     var(3) = "0.1, 0.2, 0.3";
00045     var(4) = "0.2, 0.3, 0.1";
00046
00047     cout << fixed << setprecision(3);
00048     cout << "user configured means and variances:" << endl;
00049     cout << "mu = " << mu << endl;
00050     cout << "var = " << var << endl;
00051
00052     // randomise the order of Gaussians "generating" the vectors
00053     I_Uniform_RNG rnd_uniform(0, K-1);
00054     ivec gaus_id = rnd_uniform(N);
00055
00056     ivec gaus_count(K); gaus_count = 0;
00057     Array<vec> mu_test(K); for(int k=0;k<K;k++) { mu_test(k).set_size(D); mu_test(
k) = 0.0; }
00058     Array<vec> var_test(K); for(int k=0;k<K;k++) { var_test(k).set_size(D); var_te
st(k) = 0.0; }
00059
00060     Normal_RNG rnd_normal;
00061     for(int n=0;n<N;n++) {
00062
00063         int k = gaus_id(n);
00064         gaus_count(k)++;
00065
00066         for(int d=0;d<D;d++) {
00067             rnd_normal.setup( mu(k)(d), var(k)(d) );
00068             double tmp = rnd_normal();
00069             X(n)(d) = tmp;
00070             mu_test(k)(d) += tmp;
00071         }
00072     }
00073

```

```

00074 //
00075 // find the stats for the generated data
00076
00077 for(int k=0;k<K;k++) mu_test(k) /= gaus_count(k);
00078
00079 for(int n=0;n<N;n++) {
00080     int k = gaus_id(n);
00081
00082     for(int d=0;d<D;d++) {
00083         double tmp = X(n)(d) - mu_test(k)(d);
00084         var_test(k)(d) += tmp*tmp;
00085     }
00086 }
00087
00088 for(int k=0;k<K;k++) var_test(k) /= (gaus_count(k)-1.0);
00089
00090 cout << endl << endl;
00091 cout << fixed << setprecision(3);
00092 cout << "stats for X:" << endl;
00093
00094 for(int k=0;k<K;k++) {
00095     cout << "k = " << k << " count = " << gaus_count(k) << " weight = " << gaus
_count(k)/double(N) << endl;
00096     for(int d=0;d<D;d++) cout << " d = " << d << " mu_test = " << mu_test(k)(d)
<< " var_test = " << var_test(k)(d) << endl;
00097     cout << endl;
00098 }
00099
00100
00101 // make a model with initial values (zero mean and unit variance)
00102 // the number of gaussians and dimensions of the model is specified here
00103
00104 MOG_diag mog(K,D);
00105
00106 cout << endl;
00107 cout << fixed << setprecision(3);
00108 cout << "mog.avg_log_lhood(X) = " << mog.avg_log_lhood(X) << endl;
00109
00110 //
00111 // find initial parameters via k-means (which are then used as seeds for EM bas
ed optimisation)
00112
00113 cout << endl << endl;
00114 cout << "running kmeans optimiser" << endl << endl;
00115
00116 MOG_diag_kmeans(mog, X, 10, 0.5, true, print_progress);
00117
00118 cout << fixed << setprecision(3);
00119 cout << "mog.get_means() = " << endl << mog.get_means() << endl;
00120 cout << "mog.get_diag_covs() = " << endl << mog.get_diag_covs() << endl;
00121 cout << "mog.get_weights() = " << endl << mog.get_weights() << endl;
00122
00123 cout << endl;
00124 cout << "mog.avg_log_lhood(X) = " << mog.avg_log_lhood(X) << endl;
00125
00126
00127 //
00128 // EM ML based optimisation
00129
00130 cout << endl << endl;
00131 cout << "running ML optimiser" << endl << endl;
00132

```

```

00133     MOG_diag_ML(mog, X, 10, 0.0, 0.0, print_progress);
00134
00135     cout << fixed << setprecision(3);
00136     cout << "mog.get_means() = " << endl << mog.get_means() << endl;
00137     cout << "mog.get_diag_covs() = " << endl << mog.get_diag_covs() << endl;
00138     cout << "mog.get_weights() = " << endl << mog.get_weights() << endl;
00139
00140     cout << endl;
00141     cout << "mog.avg_log_lhood(X) = " << mog.avg_log_lhood(X) << endl;
00142
00143     return 0;
00144 }

```

43.41 doc/tutorial/src/qpsk_simulation.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- `int main()`

43.41.1 Function Documentation

43.41.1.1 `int main()`

Definition at line 9 of file [qpsk_simulation.cpp](#).

43.42 qpsk_simulation.cpp

```

00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 //These lines are needed for use of cout and endl
00006 using std::cout;
00007 using std::endl;
00008
00009 int main()
00010 {
00011     //Declarations of scalars and vectors:
00012     int i, Number_of_bits;
00013     double Ec, Eb;
00014     vec EbN0dB, EbN0, N0, noise_variance, bit_error_rate; //vec is a vector contain
ing double
00015     bvec transmitted_bits, received_bits; //bvec is a vector contain
ing bits
00016     cvec transmitted_symbols, received_symbols; //cvec is a vector contain
ing double_complex
00017
00018     //Declarations of classes:
00019     QPSK qpsk; //The QPSK modulator class
00020     AWGN_Channel awgn_channel; //The AWGN channel class
00021     it_file ff; //For saving the results to file
00022     BERC berc; //Used to count the bit errors

```

```

00023 Real_Timer tt; //The timer used to measure the execution time
00024
00025 //Reset and start the timer:
00026 tt.tic();
00027
00028 //Init:
00029 Ec = 1.0; //The transmitted energy per QPSK symbol is 1.
00030 Eb = Ec / 2.0; //The transmitted energy per bit is 0.5.
00031 EbN0dB = linspace(0.0,9.0,10); //Simulate for 10 Eb/N0 values from 0 to 9 dB.
00032 EbN0 = inv_dB(EbN0dB); //Calculate Eb/N0 in a linear scale instead of d
B.
00033 N0 = Eb * pow(EbN0,-1.0); //N0 is the variance of the (complex valued) noi
se.
00034 Number_of_bits = 100000; //One hundred thousand bits is transmitted for e
ach Eb/N0 value
00035
00036 //Allocate storage space for the result vector.
00037 //The "false" argument means "Do not copy the old content of the vector to the
new storage area."
00038 bit_error_rate.set_size(EbN0dB.length(),false);
00039
00040 //Randomize the random number generators in it++:
00041 RNG_randomize();
00042
00043 //Iterate over all EbN0dB values:
00044 for (i=0; i<EbN0dB.length(); i++) {
00045
00046     //Show how the simulation progresses:
00047     cout << "Now simulating Eb/N0 value number " << i+1 << " of " << EbN0dB.lengt
h() << endl;
00048
00049     //Generate a vector of random bits to transmit:
00050     transmitted_bits = randb(Number_of_bits);
00051
00052     //Modulate the bits to QPSK symbols:
00053     transmitted_symbols = qpsk.modulate_bits(transmitted_bits);
00054
00055     //Set the noise variance of the AWGN channel:
00056     awgn_channel.set_noise(N0(i));
00057
00058     //Run the transmitted symbols through the channel using the () operator:
00059     received_symbols = awgn_channel(transmitted_symbols);
00060
00061     //Demodulate the received QPSK symbols into received bits:
00062     received_bits = qpsk.demodulate_bits(received_symbols);
00063
00064     //Calculate the bit error rate:
00065     berc.clear(); //Clear the bit error rate counte
r
00066     berc.count(transmitted_bits,received_bits); //Count the bit errors
00067     bit_error_rate(i) = berc.get_errorrte(); //Save the estimated BER in the r
esult vector
00068 }
00069
00070
00071 tt.toc();
00072
00073 //Print the results:
00074 cout << endl;
00075 cout << "EbN0dB = " << EbN0dB << " [dB]" << endl;
00076 cout << "BER = " << bit_error_rate << endl;
00077 cout << "Saving results to ./qpsk_result_file.it" << endl;

```

```
00078     cout << endl;
00079
00080     //Save the results to file:
00081     ff.open("qpsk_result_file.it");
00082     ff << Name("EbN0dB") << EbN0dB;
00083     ff << Name("ber") << bit_error_rate;
00084     ff.close();
00085
00086     //Exit program:
00087     return 0;
00088 }
00089
00090
```

43.43 doc/tutorial/src/rayleigh.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- `int main()`

43.43.1 Function Documentation

43.43.1.1 `int main()`

Definition at line 5 of file [rayleigh.cpp](#).

43.44 rayleigh.cpp

```
00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 int main()
00006 {
00007     // Declare my_channel variable as an instance of the Rayleigh_Channel
00008     // class
00009     TDL_Channel my_channel;
00010
00011     // The normalized Doppler frequency is set to 0.1
00012     double norm_dopp = 0.1;
00013     my_channel.set_norm_doppler(norm_dopp);
00014
00015     // Generate nrof_samples of the fading process and store them in ch_coeffs
00016     // matrix
00017     int nrof_samples = 10000;
00018     cmat ch_coeffs;
00019     my_channel.generate(nrof_samples, ch_coeffs);
00020
00021     // Open an output file "rayleigh_test.it"
00022     it_file ff("rayleigh_test.it");
00023
00024     // Save channel coefficients to the output file

```



```

00025     ff << Name("ch_coeffs") << ch_coeffs;
00026
00027     // Close the output file
00028     ff.close();
00029
00030     // Exit program
00031     return 0;
00032 }

```

43.45 doc/tutorial/src/read_it_file.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- int `main` ()

43.45.1 Function Documentation

43.45.1.1 int main ()

Definition at line 5 of file `read_it_file.cpp`.

43.46 read_it_file.cpp

```

00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 int main()
00006 {
00007     // Declare the it_file class
00008     it_file ff;
00009
00010     // Open the file "it_file_test.it" for reading
00011     ff.open("it_file_test.it");
00012
00013     // Read the variable a from the file. Put result in vector a.
00014     vec a;
00015     ff >> Name("a") >> a;
00016
00017     // Print the result
00018     std::cout << "a = " << a << std::endl;
00019
00020     // Exit the program
00021     return 0;
00022 }

```

43.47 doc/tutorial/src/reedsolomon.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- `int main ()`

43.47.1 Function Documentation

43.47.1.1 `int main ()`

Definition at line 9 of file `reedsolomon.cpp`.

43.48 reedsolomon.cpp

```

00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 //These lines are needed for use of cout and endl
00006 using std::cout;
00007 using std::endl;
00008
00009 int main()
00010 {
00011
00012     //Scalars and vectors:
00013     int m, t, n, k, q, NumBits, NumCodeWords;
00014     double p;
00015     bvec uncoded_bits, coded_bits, received_bits, decoded_bits;
00016
00017     //Set parameters:
00018     NumCodeWords = 1000; //Number of Reed-Solomon code-words to simulate
00019     p = 0.01; //BSC Error probability
00020     m = 3; //Reed-Solomon parameter m
00021     t = 2; //Reed-Solomon parameter t
00022
00023     cout << "Number of Reed-Solomon code-words to simulate: " << NumCodeWords << endl;
00024     cout << "BSC Error probability : " << p << endl;
00025     cout << "RS m: " << m << endl;
00026     cout << "RS t: " << t << endl;
00027
00028     //Classes:
00029     Reed_Solomon reed_solomon(m,t);
00030     BSC bsc(p);
00031     BERC berc;
00032
00033     RNG_randomize();
00034
00035     //Calculate parameters for the Reed-Solomon Code:
00036     n = round_i(pow(2.0,m)-1);
00037     k = round_i(pow(2.0,m)-1-2*t);
00038     q = round_i(pow(2.0,m));
00039
00040     cout << "Simulating an Reed-Solomon code with the following parameters:" << endl;
00041     cout << "n = " << n << endl;
00042     cout << "k = " << k << endl;
00043     cout << "q = " << q << endl;
00044

```

```

00045  NumBits = m * k * NumCodeWords;
00046  uncoded_bits = randb(NumBits);
00047  coded_bits = reed_solomon.encode(uncoded_bits);
00048  received_bits = bsc(coded_bits);
00049  decoded_bits = reed_solomon.decode(received_bits);
00050
00051  berc.count(uncoded_bits,decoded_bits);
00052  cout << "The bit error probability after decoding is " << berc.get_errorrate()
    << endl;
00053
00054  //Exit program:
00055  return 0;
00056
00057 }
```

43.49 doc/tutorial/src/spread.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- int [main](#) ()

43.49.1 Function Documentation

43.49.1.1 int main ()

Definition at line 9 of file [spread.cpp](#).

43.50 spread.cpp

```

00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 //These lines are needed for use of cout and endl
00006 using std::cout;
00007 using std::endl;
00008
00009 int main()
00010 {
00011     //Scalars:
00012     int SF, Ncode, sc, i, j, NumOfBits, MaxIterations, MaxNrOfErrors, MinNrOfErrors
    ;
00013     double Eb;
00014
00015     //Vectors and matrixes:
00016     vec EbN0dB, EbN0, N0, ber;
00017     smat all_codes, spread_codesI, spread_codesQ;
00018     bvec transmited_bits, received_bits;
00019     cvec transmited_symbols, received_symbols, transmited_chips, received_chips;
00020
00021     //Classes:
00022     Multicode_Spread_2d mc_spread;
```

```

00023   AWGN_Channel channel;
00024   QPSK qpsk;
00025   BEREC berc;
00026
00027   //Initialize the spreading:
00028   SF = 4; //The spreading factor is 4
00029   Ncode = 4; //Use all four codes in the multi-code spreader
00030   all_codes = to_smat(hadamard(SF)); //Calculate the spreading codes
00031
00032   //Set the spreading codes:
00033   spread_codesI.set_size(Ncode, SF, false);
00034   spread_codesQ.set_size(Ncode, SF, false);
00035   for (sc = 0; sc<Ncode; sc++) {
00036       spread_codesI.set_row(sc, all_codes.get_row(sc));
00037       spread_codesQ.set_row(sc, all_codes.get_row(sc));
00038   }
00039   mc_spread.set_codes(to_mat(spread_codesI), to_mat(spread_codesQ));
00040
00041   //Specify simulation specific variables:
00042   EbN0dB = linspace(-2,14,17);
00043   EbN0 = pow(10,EbN0dB/10);
00044   Eb = 1.0;
00045   N0 = Eb * pow(EbN0,-1.0);
00046   NumOfBits = 50000;
00047   MaxIterations = 10;
00048   MaxNrOfErrors = 200;
00049   MinNrOfErrors = 5;
00050   ber.set_size(EbN0dB.size(), false);
00051   ber.clear();
00052
00053   //Randomize the random number generators:
00054   RNG_randomize();
00055
00056   for (i=0; i<EbN0dB.length(); i++) {
00057
00058       cout << endl << "Simulating point nr " << i+1 << endl;
00059       berc.clear();
00060       channel.set_noise(N0(i)/2.0);
00061
00062       for (j=0; j<MaxIterations; j++) {
00063
00064           transmitted_bits = randb(NumOfBits);
00065           transmitted_symbols = qpsk.modulate_bits(transmitted_bits);
00066
00067           //This is where we do the multi-code spreading:
00068           transmitted_chips = mc_spread.spread(transmitted_symbols);
00069
00070           received_chips = channel(transmitted_chips);
00071
00072           //The multi-code despreading:
00073           //The second argument tells the despreader that the offset is zero chips.
00074           //This offset is usefull on channels with delay.
00075           received_symbols = mc_spread.despread(received_chips, 0);
00076
00077           received_bits = qpsk.demodulate_bits(received_symbols);
00078
00079           berc.count(transmitted_bits,received_bits);
00080           ber(i) = berc.get_errorrate();
00081
00082           cout << "    Iteration " << j+1 << ": ber = " << berc.get_errorrate() << endl;

```

```

00083         if (berc.get_errors() > MaxNrOfErrors) {
00084             cout << "Breaking on point " << i+1 << " with " << berc.get_errors() << "
errors." << endl; break;
00085         }
00086
00087     }
00088
00089     if (berc.get_errors() < MinNrOfErrors) {
00090         cout << "Exiting Simulation on point " << i+1 << endl; break;
00091     }
00092
00093 }
00094
00095 //Print results:
00096 cout << endl;
00097 cout << "EbN0dB = " << EbN0dB << endl;
00098 cout << "ber = " << ber << endl;
00099
00100 //Exit program:
00101 return 0;
00102
00103 }
00104

```

43.51 doc/tutorial/src/timer.cpp File Reference

```
#include <itpp/itbase.h>
```

Functions

- int [main](#) ()

43.51.1 Function Documentation

43.51.1.1 int main ()

Definition at line 9 of file [timer.cpp](#).

43.52 timer.cpp

```

00001 #include <itpp/itbase.h>
00002
00003 using namespace itpp;
00004
00005 //These lines are needed for use of cout and endl
00006 using std::cout;
00007 using std::endl;
00008
00009 int main()
00010 {
00011     //Declare the scalars used:
00012     long i, sum, N;
00013
00014     //Declare tt as an instance of the timer class:

```

```

00015   Real_Timer tt;
00016
00017   //Initiate the variables:
00018   N = 1000000;
00019   sum = 0;
00020
00021   //Start and reset the timer:
00022   tt.tic();
00023
00024   //Do some processing
00025   for (i=0; i<N; i++) {
00026       sum += i;
00027   }
00028
00029   // Print the elapsed time
00030   tt.toc_print();
00031
00032   //Print the result of the processing:
00033   cout << "The sum of all integers from 0 to " << N-1 << " equals " << sum << endl;
00034
00035   //Exit program:
00036   return 0;
00037
00038 }

```

43.53 doc/tutorial/src/vector_and_matrix.cpp File Reference

```
#include <itpp/itbase.h>
```

Functions

- int [main](#) ()

43.53.1 Function Documentation

43.53.1.1 int main ()

Definition at line 9 of file [vector_and_matrix.cpp](#).

43.54 vector_and_matrix.cpp

```

00001 #include <itpp/itbase.h>
00002
00003 using namespace itpp;
00004
00005 //These lines are needed for use of cout and endl
00006 using std::cout;
00007 using std::endl;
00008
00009 int main()
00010 {
00011     //Declare vectors and matrices:
00012     vec a, b, c;

```

```

00013     mat A, B;
00014
00015     //Use the function linspace to define a vector:
00016     a = linspace(1.0,2.0,10);
00017
00018     //Use a string of values to define a vector:
00019     b = "0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0";
00020
00021     //Add two vectors:
00022     c = a + b;
00023
00024     //Print results:
00025     cout << "a = " << a << endl;
00026     cout << "b = " << b << endl;
00027     cout << "c = " << c << endl;
00028
00029     //Use a string to define a matrix:
00030     A = "1.0 2.0;3.0 4.0";
00031
00032     //Calculate the inverse of matrix A:
00033     B = inv(A);
00034
00035     //Print results:
00036     cout << "A = " << A << endl;
00037     cout << "B = " << B << endl;
00038
00039     //Exit program:
00040     return 0;
00041
00042 }

```

43.55 doc/tutorial/src/write_it_file.cpp File Reference

```
#include <itpp/itcomm.h>
```

Functions

- [int main\(\)](#)

43.55.1 Function Documentation

43.55.1.1 [int main\(\)](#)

Definition at line 5 of file [write_it_file.cpp](#).

43.56 write_it_file.cpp

```

00001 #include <itpp/itcomm.h>
00002
00003 using namespace itpp;
00004
00005 int main()
00006 {
00007     // Declare the it_file class

```

```

00008  it_file ff;
00009
00010  // Open a file with the name "it_file_test.it"
00011  ff.open("it_file_test.it");
00012
00013  // Create some data to put into the file
00014  vec a = linspace(1, 20, 20);
00015
00016  // Put the variable a into the file. The Name("a") tells the file class
00017  // that the next variable shall be named "a".
00018  ff << Name("a") << a;
00019
00020  // Force the file to be written to disc. This is useful when performing
00021  // iterations and ensures that the information is not stored in any cache
00022  // memory. In this simple example it is not necessary to flush the file.
00023  ff.flush();
00024
00025  // Close the file
00026  ff.close();
00027
00028  // Exit program
00029  return 0;
00030 }

```

43.57 doc/tutorial/timer.doc File Reference

43.58 doc/tutorial/tutorial.doc File Reference

43.59 doc/tutorial/vector_and_matrix.doc File Reference

43.60 rmol/basic/BasConst.cpp File Reference

```

#include <rmol/basic/BasConst_General.hpp>
#include <rmol/basic/BasConst_Curves.hpp>
#include <rmol/basic/BasConst_RMOL_Service.hpp>

```

Namespaces

- namespace [RMOL](#)

Variables

- const stdair::AirlineCode_T [RMOL::DEFAULT_RMOL_SERVICE_AIRLINE_CODE](#) = "BA"
- const double [RMOL::DEFAULT_RMOL_SERVICE_CAPACITY](#) = 1.0
- const int [RMOL::DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION](#) = 100000
- const int [RMOL::DEFAULT_PRECISION](#) = 10
- const double [RMOL::DEFAULT_EPSILON](#) = 0.0001
- const double [RMOL::DEFAULT_STOPPING_CRITERION](#) = 0.01

- const double [RMOL::DEFAULT_INITIALIZER_DOUBLE_NEGATIVE](#) = -10.0
- const FRAT5Curve_T [RMOL::DEFAULT_CUMULATIVE_FRAT5_CURVE](#)
- const stdair::DCPList_T [RMOL::DEFAULT_DCP_LIST](#) = DefaultDCPList::init()

43.61 BasConst.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 #include <rmol/basic/BasConst_General.hpp>
00005 #include <rmol/basic/BasConst_Curves.hpp>
00006 #include <rmol/basic/BasConst_RMOL_Service.hpp>
00007
00008 namespace RMOL {
00009
00011     const stdair::AirlineCode_T DEFAULT_RMOL_SERVICE_AIRLINE_CODE = "BA";
00012
00014     const double DEFAULT_RMOL_SERVICE_CAPACITY = 1.0;
00015
00018     const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION = 100000;
00019
00023     const int DEFAULT_PRECISION = 10;
00024
00026     const double DEFAULT_EPSILON = 0.0001;
00027
00029     const double DEFAULT_STOPPING_CRITERION = 0.01;
00030
00032     const double DEFAULT_INITIALIZER_DOUBLE_NEGATIVE = -10.0;
00033
00036     const FRAT5Curve_T DEFAULT_CUMULATIVE_FRAT5_CURVE =
00037         DefaultMap::createCumulativeFRAT5Curve();
00038     FRAT5Curve_T DefaultMap::createCumulativeFRAT5Curve() {
00039         FRAT5Curve_T oCurve;
00040         // oCurve[63] = 1.4; oCurve[56] = 1.45;
00041         // oCurve[49] = 1.5; oCurve[42] = 1.55; oCurve[35] = 1.6;
00042         // oCurve[31] = 1.7; oCurve[27] = 1.8; oCurve[23] = 2.0;
00043         // oCurve[19] = 2.3; oCurve[16] = 2.6; oCurve[13] = 3.0;
00044         // oCurve[10] = 3.3; oCurve[7] = 3.4; oCurve[5] = 3.44;
00045         // oCurve[3] = 3.47; oCurve[1] = 3.5;
00046         oCurve[63] = 1.1; oCurve[56] = 1.11;
00047         oCurve[49] = 1.17; oCurve[42] = 1.27;
00048         oCurve[35] = 1.28; oCurve[31] = 1.28; oCurve[27] = 1.28;
00049         oCurve[23] = 1.37; oCurve[19] = 1.37;
00050         oCurve[16] = 1.6; oCurve[13] = 1.6;
00051         oCurve[10] = 1.8; oCurve[7] = 1.8;
00052         oCurve[5] = 2.23; oCurve[3] = 2.23;
00053         oCurve[1] = 2.5;
00054         // oCurve[63] = 1.05; oCurve[56] = 1.07;
00055         // oCurve[49] = 1.09; oCurve[42] = 1.11; oCurve[35] = 1.14;
00056         // oCurve[31] = 1.16; oCurve[27] = 1.18; oCurve[23] = 1.21;
00057         // oCurve[19] = 1.24; oCurve[16] = 1.27; oCurve[13] = 1.3;
00058         // oCurve[10] = 1.33; oCurve[7] = 1.37; oCurve[5] = 1.4;
00059         // oCurve[3] = 1.45; oCurve[1] = 1.5;
00060         // oCurve[63] = 1.4;
00061         // oCurve[49] = 1.5; oCurve[35] = 1.6;
00062         // oCurve[23] = 2.0; oCurve[16] = 2.6;
00063         // oCurve[10] = 3.3; oCurve[5] = 3.44;
00064         // oCurve[1] = 3.5;
00065         return oCurve;
00066     };

```

```

00067
00069  const stdair::DCPList_T DEFAULT_DCP_LIST = DefaultDCPList::init();
00070  stdair::DCPList_T DefaultDCPList::init() {
00071      stdair::DCPList_T oDCPList;
00072      oDCPList.push_back (63); oDCPList.push_back (49);
00073      oDCPList.push_back (35); oDCPList.push_back (23);
00074      oDCPList.push_back (16); oDCPList.push_back (10);
00075      oDCPList.push_back (5); oDCPList.push_back (1);
00076      oDCPList.push_back (0);
00077      return oDCPList;
00078  }
00079
00080 }

```

43.62 rmol/basic/BasConst_Curves.hpp File Reference

```
#include <rmol/RMOL_Types.hpp>
```

Classes

- struct [RMOL::DefaultMap](#)

Namespaces

- namespace [RMOL](#)

43.63 BasConst_Curves.hpp

```

00001 #ifndef __RMOL_BAS_BASCONST_CURVES_HPP
00002 #define __RMOL_BAS_BASCONST_CURVES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // RMOL
00008 #include <rmol/RMOL_Types.hpp>
00009
00010 namespace RMOL {
00011
00014     extern const FRAT5Curve_T DEFAULT_CUMULATIVE_FRAT5_CURVE;
00015
00017     struct DefaultMap {
00018         static FRAT5Curve_T createCumulativeFRAT5Curve();
00019     };
00020 }
00021 #endif // __RMOL_BAS_BASCONST_CURVES_HPP

```

43.64 rmol/basic/BasConst_General.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Classes

- struct [RMOL::DefaultDCPList](#)

Namespaces

- namespace [RMOL](#)

43.65 BasConst_General.hpp

```

00001 #ifndef __RMOL_BAS_BASCONST_GENERAL_HPP
00002 #define __RMOL_BAS_BASCONST_GENERAL_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_types.hpp>
00009
00010 namespace RMOL {
00011
00014     extern const int DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION;
00015
00018     extern const int DEFAULT_PRECISION;
00019
00021     extern const double DEFAULT_EPSILON;
00022
00024     extern const double DEFAULT_STOPPING_CRITERION;
00025
00027     extern const double DEFAULT_INITIALIZER_DOUBLE_NEGATIVE;
00028
00030     extern const stdair::DCPList_T DEFAULT_DCP_LIST;
00031     struct DefaultDCPList { static stdair::DCPList_T init(); };
00032 }
00033 #endif // __RMOL_BAS_BASCONST_GENERAL_HPP

```

43.66 rmol/basic/BasConst_RMOL_Service.hpp File Reference

```

#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <rmol/RMOL_Types.hpp>

```

Namespaces

- namespace [RMOL](#)

43.67 BasConst_RMOL_Service.hpp

```

00001 #ifndef __RMOL_BAS_BASCONST_RMOL_SERVICE_HPP
00002 #define __RMOL_BAS_BASCONST_RMOL_SERVICE_HPP

```

```

00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <vector>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 // RMOL
00012 #include <rmol/RMOL_Types.hpp>
00013
00014 namespace RMOL {
00015
00017     extern const stdair::AirlineCode_T DEFAULT_RMOL_SERVICE_AIRLINE_CODE;
00018
00020     extern const double DEFAULT_RMOL_SERVICE_CAPACITY;
00021
00022 }
00023 #endif // __RMOL_BAS_BASCONST_RMOL_SERVICE_HPP

```

43.68 rmol/batches/rmol.cpp File Reference

```

#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/program_options.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/RMOL_Service.hpp>
#include <rmol/config/rmol-paths.hpp>

```

Functions

- const std::string [K_RMOL_DEFAULT_LOG_FILENAME](#) ("rmol.log")
- const std::string [K_RMOL_DEFAULT_INPUT_FILENAME](#) (STDAIR_SAMPLE_DIR"/rm01.csv")
- template<class T >
std::ostream & [operator<<](#) (std::ostream &os, const std::vector< T > &v)
- int [readConfiguration](#) (int argc, char *argv[], int &ioRandomDraws, double &ioCapacity, short &ioMethod, bool &iolsBuiltin, std::string &ioInputFilename, std::string &ioLogFilename)
- void [optimise](#) (RMOL::RMOL_Service &rmolService, const short &iMethod, const int &iRandomDraws)
- int [main](#) (int argc, char *argv[])

Variables

- const bool [K_RMOL_DEFAULT_BUILT_IN_INPUT](#) = false
- const int [K_RMOL_DEFAULT_RANDOM_DRAWS](#) = 100000
- const double [K_RMOL_DEFAULT_CAPACITY](#) = 500.0
- const short [K_RMOL_DEFAULT_METHOD](#) = 0
- const int [K_RMOL_EARLY_RETURN_STATUS](#) = 99

43.68.1 Function Documentation

43.68.1.1 `const std::string K_RMOL_DEFAULT_LOG_FILENAME ("rmol.log")`

Default name and location for the log file.

Referenced by [readConfiguration\(\)](#).

43.68.1.2 `const std::string K_RMOL_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR"rm01.csv")`

Default name and location for the (CSV) input file.

Referenced by [readConfiguration\(\)](#).

43.68.1.3 `template<class T > std::ostream& operator<< (std::ostream & os, const std::vector< T > & v)`

Definition at line 47 of file [rmol.cpp](#).

43.68.1.4 `int readConfiguration (int argc, char * argv[], int & ioRandomDraws, double & ioCapacity, short & ioMethod, bool & iolsBuiltin, std::string & ioInputFilename, std::string & ioLogFilename)`

Read and parse the command line options.

Definition at line 57 of file [rmol.cpp](#).

References [K_RMOL_DEFAULT_BUILT_IN_INPUT](#), [K_RMOL_DEFAULT_CAPACITY](#), [K_RMOL_DEFAULT_INPUT_FILENAME\(\)](#), [K_RMOL_DEFAULT_LOG_FILENAME\(\)](#), [K_RMOL_DEFAULT_METHOD](#), [K_RMOL_DEFAULT_RANDOM_DRAWS](#), [K_RMOL_EARLY_RETURN_STATUS](#), [PACKAGE_NAME](#), [PACKAGE_VERSION](#), and [PREFIXDIR](#).

Referenced by [main\(\)](#).

43.68.1.5 `void optimise (RMOL::RMOL_Service & rmolService, const short & iMethod, const int & iRandomDraws)`

Definition at line 167 of file [rmol.cpp](#).

References [RMOL::RMOL_Service::heuristicOptimisationByEmsr\(\)](#), [RMOL::RMOL_Service::heuristicOptimisationByEmsrB\(\)](#), [RMOL::RMOL_Service::optimalOptimisationByDP\(\)](#), and [RMOL::RMOL_Service::optimalOptimisationByMCIntegration\(\)](#).

Referenced by [main\(\)](#).

43.68.1.6 `int main (int argc, char * argv[])`

Definition at line 204 of file `rmol.cpp`.

References `RMOL::RMOL_Service::buildSampleBom()`, `K_RMOL_EARLY_RETURN_STATUS`, `optimise()`, `RMOL::RMOL_Service::parseAndLoad()`, and `readConfiguration()`.

43.68.2 Variable Documentation

43.68.2.1 `const bool K_RMOL_DEFAULT_BUILT_IN_INPUT = false`

Default for the input type. It can be either built-in or provided by an input file. That latter must then be given with the `-i/--input` option.

Definition at line 23 of file `rmol.cpp`.

Referenced by `readConfiguration()`.

43.68.2.2 `const int K_RMOL_DEFAULT_RANDOM_DRAWS = 100000`

Default number of random draws to be generated (best if over 100).

Definition at line 29 of file `rmol.cpp`.

Referenced by `readConfiguration()`.

43.68.2.3 `const double K_RMOL_DEFAULT_CAPACITY = 500.0`

Default value for the capacity of the resource (e.g., a flight cabin).

Definition at line 32 of file `rmol.cpp`.

Referenced by `readConfiguration()`.

43.68.2.4 `const short K_RMOL_DEFAULT_METHOD = 0`

Default name and location for the Revenue Management method to be used.

- 0 = Monte-Carlo
- 1 = Dynamic Programming
- 2 = EMSR
- 3 = EMSR-a
- 4 = EMSR-b

Definition at line 43 of file `rmol.cpp`.

Referenced by `readConfiguration()`.

43.68.2.5 `const int K_RMOL_EARLY_RETURN_STATUS = 99`

Early return status (so that it can be differentiated from an error).

Definition at line 54 of file [rmol.cpp](#).

Referenced by [main\(\)](#), and [readConfiguration\(\)](#).

43.69 rmol.cpp

```

00001 // STL
00002 #include <cassert>
00003 #include <iostream>
00004 #include <sstream>
00005 #include <fstream>
00006 #include <string>
00007 // Boost (Extended STL)
00008 #include <boost/date_time/posix_time/posix_time.hpp>
00009 #include <boost/date_time/gregorian/gregorian.hpp>
00010 #include <boost/program_options.hpp>
00011 // StdAir
00012 #include <stdair/service/Logger.hpp>
00013 // RMOL
00014 #include <rmol/RMOL_Service.hpp>
00015 #include <rmol/config/rmol-paths.hpp>
00016
00017 // ////////// Constants //////////
00019 const std::string K_RMOL_DEFAULT_LOG_FILENAME ("rmol.log");
00020
00023 const bool K_RMOL_DEFAULT_BUILT_IN_INPUT = false;
00024
00026 const std::string K_RMOL_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR "/rmol.csv");
00027
00029 const int K_RMOL_DEFAULT_RANDOM_DRAWS = 100000;
00030
00032 const double K_RMOL_DEFAULT_CAPACITY = 500.0;
00033
00043 const short K_RMOL_DEFAULT_METHOD = 0;
00044
00045 // ////////// Parsing of Options & Configuration //////////
00046 // A helper function to simplify the main part.
00047 template<class T> std::ostream& operator<< (std::ostream& os,
00048                                     const std::vector<T>& v) {
00049     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00050     return os;
00051 }
00052
00054 const int K_RMOL_EARLY_RETURN_STATUS = 99;
00055
00057 int readConfiguration(int argc, char* argv[],
00058                     int& ioRandomDraws, double& ioCapacity,
00059                     short& ioMethod, bool& ioIsBuiltin,
00060                     std::string& ioInputFilename, std::string& ioLogFilename){
00061
00062     // Default for the built-in input
00063     ioIsBuiltin = K_RMOL_DEFAULT_BUILT_IN_INPUT;
00064
00065     // Declare a group of options that will be allowed only on command line
00066     boost::program_options::options_description generic ("Generic options");
00067     generic.add_options()
00068         ("prefix", "print installation prefix")
00069         ("version,v", "print version string")
00070         ("help,h", "produce help message");
00071
00072     // Declare a group of options that will be allowed both on command

```

```

00073 // line and in config file
00074 boost::program_options::options_description config ("Configuration");
00075 config.add_options()
00076     ("draws,d",
00077      boost::program_options::value<int>(&ioRandomDraws)->default_value(
00078      K_RMOL_DEFAULT_RANDOM_DRAWS),
00079      "Number of to-be-generated random draws")
00080     ("capacity,c",
00081      boost::program_options::value<double>(&ioCapacity)->default_value(
00082      K_RMOL_DEFAULT_CAPACITY),
00083      "Resource capacity (e.g., for a flight leg)")
00084     ("method,m",
00085      boost::program_options::value<short>(&ioMethod)->default_value(
00086      K_RMOL_DEFAULT_METHOD),
00087      "Revenue Management method to be used (0 = Monte-Carlo, 1 = Dynamic Programm
00088      ing, 2 = EMSR, 3 = EMSR-a, 4 = EMSR-b)")
00089     ("builtin,b",
00090      "The cabin set up can be either built-in or parsed from an input file. That
00091      latter must then be given with the -i/--input option")
00092     ("input,i",
00093      boost::program_options::value< std::string >(&ioInputFilename)->default_valu
00094      e(K_RMOL_DEFAULT_INPUT_FILENAME),
00095      "(CSV) input file for the demand distribution parameters and resource (leg-c
00096      abin) capacities")
00097     ("log,l",
00098      boost::program_options::value< std::string >(&ioLogFilename)->default_value(
00099      K_RMOL_DEFAULT_LOG_FILENAME),
00100      "Filename for the logs")
00101     ;
00102
00103 // Hidden options, will be allowed both on command line and
00104 // in config file, but will not be shown to the user.
00105 boost::program_options::options_description hidden ("Hidden options");
00106 hidden.add_options()
00107     ("copyright",
00108      boost::program_options::value< std::vector<std::string> >(),
00109      "Show the copyright (license)");
00110
00111 boost::program_options::options_description cmdline_options;
00112 cmdline_options.add(generic).add(config).add(hidden);
00113
00114 boost::program_options::options_description config_file_options;
00115 config_file_options.add(config).add(hidden);
00116
00117 boost::program_options::options_description visible ("Allowed options");
00118 visible.add(generic).add(config);
00119
00120 boost::program_options::positional_options_description p;
00121 p.add ("copyright", -1);
00122
00123 boost::program_options::variables_map vm;
00124 boost::program_options::
00125     store (boost::program_options::command_line_parser (argc, argv).
00126            options (cmdline_options).positional(p).run(), vm);
00127
00128 std::ifstream ifs ("rmol.cfg");
00129 boost::program_options::store (parse_config_file (ifs, config_file_options),
00130                                vm);
00131 boost::program_options::notify (vm);
00132
00133 if (vm.count ("help")) {
00134     std::cout << visible << std::endl;

```



```

00127     return K_RMOL_EARLY_RETURN_STATUS;
00128 }
00129
00130 if (vm.count ("version")) {
00131     std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
00132     return K_RMOL_EARLY_RETURN_STATUS;
00133 }
00134
00135 if (vm.count ("prefix")) {
00136     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00137     return K_RMOL_EARLY_RETURN_STATUS;
00138 }
00139
00140 if (vm.count ("builtin")) {
00141     ioIsBuiltin = true;
00142 }
00143 const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00144 std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00145
00146 if (ioIsBuiltin == false) {
00147     if (vm.count ("input")) {
00148         ioInputFilename = vm["input"].as< std::string >();
00149         std::cout << "Input filename is: " << ioInputFilename << std::endl;
00150     }
00151 }
00152
00153 if (vm.count ("log")) {
00154     ioLogFilename = vm["log"].as< std::string >();
00155     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00156 }
00157
00158 std::cout << "The number of random draws is: " << ioRandomDraws << std::endl;
00159 std::cout << "The resource capacity is: " << ioCapacity << std::endl;
00160 std::cout << "The optimisation method is: " << ioMethod << std::endl;
00161 std::cout << std::endl;
00162
00163 return 0;
00164 }
00165
00166 // //////////////////////////////////////
00167 void optimise (RMOL::RMOL_Service& rmolService,
00168               const short& iMethod, const int& iRandomDraws) {
00169
00170     switch (iMethod) {
00171     case 0: {
00172         // Calculate the optimal protections by the Monte Carlo
00173         // Integration approach
00174         rmolService.optimalOptimisationByMCIntegration (iRandomDraws);
00175         break;
00176     }
00177     case 1: {
00178         // Calculate the optimal protections by DP.
00179         rmolService.optimalOptimisationByDP ();
00180         break;
00181     }
00182     case 2: {
00183         // Calculate the Bid-Price Vector by EMSR
00184         rmolService.heuristicOptimisationByEmsr ();
00185         break;
00186     }
00187     case 3: {
00188         // Calculate the protections by EMSR-a

```

```

00189     rmolService.heuristicOptimisationByEmsrA ();
00190     break;
00191 }
00192 case 4: {
00193     // Calculate the protections by EMSR-b
00194     rmolService.heuristicOptimisationByEmsrB ();
00195     break;
00196 }
00197 default: {
00198     rmolService.optimalOptimisationByMCIntegration (iRandomDraws);
00199 }
00200 }
00201 }
00202
00203 // ////////// M A I N //////////
00204 int main (int argc, char* argv[]) {
00205
00206     // Number of random draws to be generated (best if greater than 100)
00207     int lRandomDraws = 0;
00208
00209     // Cabin Capacity (it must be greater then 100 here)
00210     double lCapacity = 0.0;
00211
00212     // Methods of optimisation (0 = Monte-Carlo, 1 = Dynamic Programming,
00213     // 2 = EMSR, 3 = EMSR-a, 4 = EMSR-b)
00214     short lMethod = 0;
00215
00216     // Built-in
00217     bool isBuiltin;
00218
00219     // Input file name
00220     std::string lInputFilename;
00221
00222     // Output log File
00223     std::string lLogFilename;
00224
00225     // Call the command-line option parser
00226     const int lOptionParserStatus =
00227         readConfiguration (argc, argv, lRandomDraws, lCapacity, lMethod,
00228                             isBuiltin, lInputFilename, lLogFilename);
00229
00230     if (lOptionParserStatus == K_RMOL_EARLY_RETURN_STATUS) {
00231         return 0;
00232     }
00233
00234     // Set the log parameters
00235     std::ofstream logOutputFile;
00236     // Open and clean the log outputfile
00237     logOutputFile.open (lLogFilename.c_str());
00238     logOutputFile.clear();
00239
00240     // Initialise the log stream
00241     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00242
00243     // Initialise the RMOL service
00244     RMOL::RMOL_Service rmolService (lLogParams);
00245
00246     if (isBuiltin == true) {
00247         // DEBUG
00248         STDAIR_LOG_DEBUG ("No input file has been given."
00249                         "A sample BOM tree will therefore be built.");
00250

```

```

00251     // Build a sample BOM tree
00252     rmolService.buildSampleBom();
00253
00254 } else {
00255     // DEBUG
00256     STDAIR_LOG_DEBUG ("RMOL will parse " << lInputFilename
00257                     << " and build the corresponding BOM tree.");
00258
00259     //
00260     rmolService.parseAndLoad (lCapacity, lInputFilename);
00261 }
00262
00263 // Launch the optimisation
00264 optimise (rmolService, lMethod, lRandomDraws);
00265
00266 //
00267 logOutputFile.close();
00268
00269 return 0;
00270 }

```

43.70 rmol/bom/BucketHolderTypes.hpp File Reference

```

#include <list>
#include <map>
#include <stdair/stdair_basic_types.hpp>

```

Namespaces

- namespace [RMOL](#)

Typedefs

- typedef std::list< BucketHolder * > [RMOL::BucketHolderList_T](#)

43.71 BucketHolderTypes.hpp

```

00001 #ifndef __RMOL_BUCKETHOLDERTYPES_HPP
00002 #define __RMOL_BUCKETHOLDERTYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <list>
00009 #include <map>
00010 // STDAIR
00011 #include <stdair/stdair_basic_types.hpp>
00012
00013 namespace RMOL {
00014
00016     class BucketHolder;
00017

```

```

00019     typedef std::list<BucketHolder*> BucketHolderList_T;
00020
00023     typedef std::map<const stdair::MapKey_T, BucketHolder*>;
00024 }
00025 #endif // __RMOL_BUCKETHOLDERTYPES_HPP

```

43.72 rmol/bom/DistributionParameterList.hpp File Reference

```

#include <list>

#include <rmol/field/FldDistributionParameters.hpp>

```

Namespaces

- namespace [RMOL](#)

Typedefs

- typedef std::list< FldDistributionParameters > [RMOL::DistributionParameterList_T](#)

43.73 DistributionParameterList.hpp

```

00001 #ifndef __RMOL_DISTRIBUTIONPARAMETERLIST_HPP
00002 #define __RMOL_DISTRIBUTIONPARAMETERLIST_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <list>
00009 // RMOL
00010 #include <rmol/field/FldDistributionParameters.hpp>
00011
00012 namespace RMOL {
00013
00016     typedef std::list<FldDistributionParameters> DistributionParameterList_T;
00017
00018 }
00019 #endif // __RMOL_DISTRIBUTIONPARAMETERLIST_HPP

```

43.74 rmol/bom/DPOptimiser.cpp File Reference

```

#include <cassert>

#include <sstream>

#include <vector>

#include <cmath>

#include <boost/math/distributions/normal.hpp>

```

```
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/bom/DPOptimiser.hpp>
```

Namespaces

- namespace [RMOL](#)

43.75 DPOptimiser.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <vector>
00008 #include <cmath>
00009 // Boost Math
00010 #include <boost/math/distributions/normal.hpp>
00011 // StdAir
00012 #include <stdair/bom/LegCabin.hpp>
00013 #include <stdair/bom/VirtualClassStruct.hpp>
00014 #include <stdair/service/Logger.hpp>
00015 // RMOL
00016 #include <rmol/basic/BasConst_General.hpp>
00017 #include <rmol/bom/DPOptimiser.hpp>
00018
00019 namespace RMOL {
00020
00021 // //////////////////////////////////////
00022 void DPOptimiser::optimalOptimisationByDP (stdair::LegCabin& ioLegCabin) {
00023     // // Number of classes/buckets: n
00024     // const short nbOfClasses = ioBucketHolder.getSize();
00025
00026     // // Number of values of x to compute for each Vj(x).
00027     // const int maxValue = static_cast<int> (iCabinCapacity * DEFAULT_PRECISION)
00028     ;
00029     // // Vector of the Expected Maximal Revenue (Vj).
00030     // std::vector< std::vector<double> > MERVectorHolder;
00031
00032     // // Vector of V_0(x).
00033     // std::vector<double> initialMERVector (maxValue+1, 0.0);
00034     // MERVectorHolder.push_back (initialMERVector);
00035
00036     // // Current cumulative protection level (y_j * DEFAULT_PRECISION).
00037     // // Initialise with y_0 = 0.
00038     // int currentProtection = 0;
00039
00040     // int currentBucketIndex = 1;
00041     // ioBucketHolder.begin();
00042 }
```

```

00043 // while (currentProtection < maxValue && currentBucketIndex < nbOfClasses) {
00044 // //while (currentBucketIndex == 1) {
00045 //   bool protectionChanged = false;
00046 //   double nextProtection = 0.0;
00047 //   std::vector<double> currentMERVector;
00048 //   // double testGradient = 10000;
00049 //
00050 //   Bucket& currentBucket = ioBucketHolder.getCurrentBucket();
00051 //   const double meanDemand = currentBucket.getMean();
00052 //   const double SDDemand = currentBucket.getStandardDeviation();
00053 //   const double currentYield = currentBucket.getAverageYield();
00054 //   const double errorFactor = 1.0;
00055 //
00056 //   Bucket& nextBucket = ioBucketHolder.getNextBucket();
00057 //   const double nextYield = nextBucket.getAverageYield();
00058 //
00059 //   // For x <= currentProtection (y_(j-1)), V_j(x) = V_(j-1)(x).
00060 //   for (int x = 0; x <= currentProtection; ++x) {
00061 //     const double MERValue = MERVectorHolder.at(currentBucketIndex-1).at(x)
00062 //     ;
00063 //     currentMERVector.push_back (MERValue);
00064 //   }
00065 //   //
00066 //   boost::math::normal lNormalDistribution (meanDemand, SDDemand);
00067 //
00068 //   // Vector of gaussian pdf values.
00069 //   std::vector<double> pdfVector;
00070 //   for (int s = 0; s <= maxValue - currentProtection; ++s) {
00071 //     const double pdfValue =
00072 //       boost::math::pdf (lNormalDistribution, s/DEFAULT_PRECISION);
00073 //     pdfVector.push_back (pdfValue);
00074 //   }
00075 //
00076 //   // Vector of gaussian cdf values.
00077 //   std::vector<double> cdfVector;
00078 //   for (int s = 0; s <= maxValue - currentProtection; ++s) {
00079 //     const double cdfValue =
00080 //       boost::math::cdf (boost::math::complement (lNormalDistribution,
00081 //       s/DEFAULT_PRECISION));
00082 //     cdfVector.push_back (cdfValue);
00083 //   }
00084 //
00085 //   // Compute V_j(x) for x > currentProtection (y_(j-1)).
00086 //   for (int x = currentProtection + 1; x <= maxValue; ++x) {
00087 //     const double lowerBound = static_cast<double> (x - currentProtection);
00088 //
00089 //     // Compute the first integral in the V_j(x) formulation (see
00090 //     // the memo of Jerome Contant).
00091 //     const double power1 =
00092 //       - 0.5 * meanDemand * meanDemand / (SDDemand * SDDemand);
00093 //     const double e1 = std::exp (power1);
00094 //     const double power2 =
00095 //       - 0.5 * (lowerBound / DEFAULT_PRECISION - meanDemand)
00096 //       * (lowerBound / DEFAULT_PRECISION - meanDemand)
00097 //       / (SDDemand * SDDemand);
00098 //     const double e2 = std::exp (power2);
00099 //
00100 //     const double cdfValue0 =
00101 //       boost::math::cdf (boost::math::complement (lNormalDistribution,

```

```

00102         //                                     0.0));
00103         //     const double cdfValue1 =
00104         //         boost::math::cdf(boost::math::complement(lNormalDistribution,
00105         //                                     lowerBound/DEFAULT_PRECISIO
N));
00106         //     const double integralResult1 = currentYield
00107         //         * ((e1 - e2) * SDDemand / sqrt (2 * 3.14159265)
00108         //         + meanDemand * (cdfValue0 - cdfValue1));
00109
00110         //     double integralResult2 = 0.0;
00111
00112         //     for (int s = 0; s < lowerBound; ++s) {
00113         //         const double partialResult =
00114         //             2 * MERVectorHolder.at(currentBucketIndex-1).at(x-s)
00115         //             * pdfVector.at(s);
00116
00117         //         integralResult2 += partialResult;
00118         //     }
00119         //     integralResult2 -= MERVectorHolder.at(currentBucketIndex-1).at(x) *
00120         //         pdfVector.at(0);
00121
00122         //     const int intLowerBound = static_cast<int>(lowerBound);
00123         //     integralResult2 +=
00124         //         MERVectorHolder.at(currentBucketIndex-1).at(x - intLowerBound) *
00125         //         pdfVector.at(intLowerBound);
00126
00127         //     integralResult2 /= 2 * DEFAULT_PRECISION;
00128         //     /*
00129         //     for (int s = 0; s < lowerBound; ++s) {
00130         //         const double partialResult =
00131         //             (MERVectorHolder.at(currentBucketIndex-1).at(x-s) +
00132         //             MERVectorHolder.at(currentBucketIndex-1).at(x-s-1)) *
00133         //             (cdfVector.at(s+1) - cdfVector.at(s)) / 2;
00134         //         integralResult2 += partialResult;
00135         //     }
00136         //     */
00137         //     const double firstElement = integralResult1 + integralResult2;
00138
00139         //     // Compute the second integral in the V_j(x) formulation (see
00140         //     // the memo of Jerome Contant).
00141         //     const double constCoefOfSecondElement =
00142         //         currentYield * lowerBound / DEFAULT_PRECISION
00143         //         + MERVectorHolder.at(currentBucketIndex-1).at(currentProtection);
00144
00145         //     const double secondElement = constCoefOfSecondElement
00146         //         * boost::math::cdf(boost::math::complement(lNormalDistribution,
00147         //                                     lowerBound/DEFAULT_PRECIS
ION));
00148
00149         //     const double MERValue = (firstElement + secondElement) / errorFactor;
00150
00151
00152         //     assert (currentMERVector.size() > 0);
00153         //     const double lastMERValue = currentMERVector.back();
00154
00155         //     const double currentGradient =
00156         //         (MERValue - lastMERValue) * DEFAULT_PRECISION;
00157
00158         //     //assert (currentGradient >= 0);
00159         //     if (currentGradient < -0) {
00160         //         std::ostringstream ostr;
00161         //         ostr << currentGradient << std::endl

```

```

00162      //          << "x = " << x << std::endl
00163      //          << "class: " << currentBucketIndex << std::endl;
00164      //          STDAIR_LOG_DEBUG (ostr.str());
00165      //      }
00166
00167      //      /*
00168      //      assert (currentGradient <= testGradient);
00169      //      testGradient = currentGradient;
00170      //      */
00171      //      if (protectionChanged == false && currentGradient <= nextYield) {
00172      //          nextProtection = x - 1;
00173      //          protectionChanged = true;
00174      //      }
00175
00176      //      if (protectionChanged == true && currentGradient > nextYield) {
00177      //          protectionChanged = false;
00178      //      }
00179
00180      //      if (protectionChanged == false && x == maxValue) {
00181      //          nextProtection = maxValue;
00182      //      }
00183
00184      //      currentMERVector.push_back (MERValue);
00185      //  }
00186
00187      //  // DEBUG
00188      //  STDAIR_LOG_DEBUG ("Vmaxindex = " << currentMERVector.back());
00189
00190      //  MERVectorHolder.push_back (currentMERVector);
00191
00192      //  const double realProtection = nextProtection / DEFAULT_PRECISION;
00193      //  const double bookingLimit = iCabinCapacity - realProtection;
00194
00195      //  currentBucket.setCumulatedProtection (realProtection);
00196      //  nextBucket.setCumulatedBookingLimit (bookingLimit);
00197
00198      //  currentProtection = static_cast<int> (std::floor (nextProtection));
00199
00200      //  ioBucketHolder.iterate();
00201      //  ++currentBucketIndex;
00202      // }
00203  }
00204
00205 }
```

43.76 rmol/bom/DPOptimiser.hpp File Reference

```
#include <rmol/RMOL_Types.hpp>
```

Classes

- class [RMOL::DPOptimiser](#)

Namespaces

- namespace [stdair](#)

- namespace [RMOL](#)

43.77 DPOptimiser.hpp

```

00001 #ifndef __RMOL_BOM_DPOPTIMISER_HPP
00002 #define __RMOL_BOM_DPOPTIMISER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // RMOL
00008 #include <rmol/RMOL_Types.hpp>
00009
00011 namespace stdair {
00012     class LegCabin;
00013 }
00014
00015 namespace RMOL {
00017     class DPOptimiser {
00018     public:
00019
00025         static void optimalOptimisationByDP (stdair::LegCabin&);
00026
00030         static double cdfGaussianQ (const double, const double);
00031     };
00032 }
00033 #endif // __RMOL_BOM_DPOPTIMISER_HPP

```

43.78 rmol/bom/EMDetruncator.cpp File Reference

```

#include <iostream>
#include <cmath>
#include <vector>
#include <cassert>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/EMDetruncator.hpp>

```

Namespaces

- namespace [RMOL](#)

43.79 EMDetruncator.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////

```

```

00004 // STL
00005 #include <iostream>
00006 #include <cmath>
00007 #include <vector>
00008 #include <cassert>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/service/Logger.hpp>
00012 // RMOL
00013 #include <rmol/bom/HistoricalBookingHolder.hpp>
00014 #include <rmol/bom/EMDetruncator.hpp>
00015
00016 namespace RMOL {
00017
00018 ///////////////////////////////////////////////////////////////////
00019 void EMDetruncator::unconstrainUsingEMMethod
00020 (HistoricalBookingHolder& ioHistoricalBookingHolder) {
00021
00022     // Number of flights.
00023     const short lNbOfFlights =
00024         ioHistoricalBookingHolder.getNbOfFlights();
00025
00026     // Number of uncensored booking data.
00027     const short lNbOfUncensoredData =
00028         ioHistoricalBookingHolder.getNbOfUncensoredData();
00029
00030     if (lNbOfUncensoredData > 1) {
00031         // Number of uncensored bookings.
00032         const stdair::NbOfBookings_T lNbOfUncensoredBookings =
00033             ioHistoricalBookingHolder.getNbOfUncensoredBookings();
00034
00035         const double lMeanOfUncensoredBookings =
00036             static_cast<double>(lNbOfUncensoredBookings/lNbOfUncensoredData);
00037
00038         const double lStdDevOfUncensoredBookings =
00039             ioHistoricalBookingHolder.getUncensoredStandardDeviation
00040                 (lMeanOfUncensoredBookings, lNbOfUncensoredData);
00041
00042         std::vector<bool> toBeUnconstrained =
00043             ioHistoricalBookingHolder.getListOfToBeUnconstrainedFlags();
00044
00045         double lDemandMean = lMeanOfUncensoredBookings;
00046         double lStdDev = lStdDevOfUncensoredBookings;
00047
00048         // DEBUG
00049         STDAIR_LOG_DEBUG ("mean: " << lDemandMean << ", std: " << lStdDev);
00050
00051         if (lStdDev != 0) {
00052             bool stopUnconstraining = false;
00053             while (stopUnconstraining == false) {
00054                 stopUnconstraining = true;
00055
00056                 for (short i = 0; i < lNbOfFlights; ++i) {
00057                     if (toBeUnconstrained.at(i) == true) {
00058                         // Get the unconstrained demand of the (i+1)-th flight.
00059                         const stdair::NbOfBookings_T demand =
00060                             ioHistoricalBookingHolder.getUnconstrainedDemand (i);
00061                         //STDAIR_LOG_DEBUG ("demand: " << demand);
00062
00063                         // Execute the Expectation step.
00064                         const stdair::NbOfBookings_T expectedDemand =
00065                             ioHistoricalBookingHolder.

```

```

00066         calculateExpectedDemand (lDemandMean, lStdDev, i, demand);
00067         //STDAIR_LOG_DEBUG ("expected: " << expectedDemand);
00068         assert (expectedDemand >= 0 || expectedDemand < 0);
00069
00070         double absDiff =
00071             static_cast<double>(expectedDemand - demand);
00072
00073         if (absDiff < 0) {
00074             absDiff = - absDiff;
00075         }
00076         if (absDiff < 0.001) {
00077             toBeUnconstrained.at (i) = false;
00078         }
00079         else {
00080             stopUnconstraining = false;
00081         }
00082
00083         ioHistoricalBookingHolder.setUnconstrainedDemand (expectedDemand,
00084                                                         i);
00085     }
00086 }
00087
00088 if (stopUnconstraining == false) {
00089     lDemandMean = ioHistoricalBookingHolder.getDemandMean();
00090     lStdDev =
00091         ioHistoricalBookingHolder.getStandardDeviation (lDemandMean);
00092 }
00093 }
00094 }
00095 }
00096
00097 }
00098 }

```

43.80 rmol/bom/EMDetruncator.hpp File Reference

Classes

- class [RMOL::EMDetruncator](#)

Namespaces

- namespace [RMOL](#)

43.81 EMDetruncator.hpp

```

00001 #ifndef __RMOL_BOM_EMDETRUNCATOR_HPP
00002 #define __RMOL_BOM_EMDETRUNCATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 namespace RMOL {
00008     // Forward declarations.
00009     struct HistoricalBookingHolder;
00010

```

```

00012     class EMDetruncator {
00013     public:
00016         static void unconstrainUsingEMMethod (HistoricalBookingHolder&);
00017     };
00018 }
00019 #endif // __RMOL_BOM_EMDETRUNCATOR_HPP

```

43.82 rmol/bom/Emsr.cpp File Reference

```

#include <assert.h>
#include <iostream>
#include <cmath>
#include <list>
#include <algorithm>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <rmol/bom/Emsr.hpp>
#include <rmol/bom/EmsrUtils.hpp>

```

Namespaces

- namespace [RMOL](#)

43.83 Emsr.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // C
00005 #include <assert.h>
00006 // STL
00007 #include <iostream>
00008 #include <cmath>
00009 #include <list>
00010 #include <algorithm>
00011 // StdAir
00012 #include <stdair/stdair_rm_types.hpp>
00013 #include <stdair/bom/LegCabin.hpp>
00014 #include <stdair/bom/VirtualClassStruct.hpp>
00015 // RMOL
00016 #include <rmol/bom/Emsr.hpp>
00017 #include <rmol/bom/EmsrUtils.hpp>
00018
00019 namespace RMOL {
00020     // //////////////////////////////////////
00021     void Emsr::heuristicOptimisationByEmsrA (stdair::LegCabin& ioLegCabin) {
00022         stdair::VirtualClassList_T& lVirtualClassList =
00023             ioLegCabin.getVirtualClassList ();

```

```

00024     const stdair::CabinCapacity_T& lCabinCapacity =
00025         ioLegCabin.getOfferedCapacity();
00026
00032     stdair::VirtualClassList_T::iterator itVC = lVirtualClassList.begin();
00033     assert (itVC != lVirtualClassList.end());
00034
00035     stdair::VirtualClassStruct& lFirstVC = *itVC;
00036     lFirstVC.setCumulatedBookingLimit (lCabinCapacity);
00037     ++itVC;
00038     for (; itVC != lVirtualClassList.end(); ++itVC) {
00039         stdair::VirtualClassStruct& lNextVC = *itVC;
00040
00041         // Initialise the protection for class/bucket j.
00042         stdair::ProtectionLevel_T lProtectionLevel = 0.0;
00043
00044         for(stdair::VirtualClassList_T::iterator itHigherVC =
00045             lVirtualClassList.begin(); itHigherVC != itVC; ++itHigherVC) {
00046             stdair::VirtualClassStruct& lHigherVC = *itHigherVC;
00047             const double lPartialProtectionLevel =
00048                 EmsrUtils::computeProtectionLevel (lHigherVC, lNextVC);
00049             lProtectionLevel += lPartialProtectionLevel;
00050         }
00051         stdair::VirtualClassList_T::iterator itCurrentVC = itVC; --itCurrentVC;
00052         stdair::VirtualClassStruct& lCurrentVC = *itCurrentVC;
00053         lCurrentVC.setCumulatedProtection (lProtectionLevel);
00054
00055         // Compute the booking limit for the class/bucket j+1 (can be negative).
00056         const double lBookingLimit = lCabinCapacity - lProtectionLevel;
00057
00058         // Set the booking limit for class/bucket j+1.
00059         lNextVC.setCumulatedBookingLimit (lBookingLimit);
00060     }
00061 }
00062
00063 // //////////////////////////////////////
00064 void Emsr::heuristicOptimisationByEmsrB (stdair::LegCabin& ioLegCabin) {
00065     stdair::VirtualClassList_T& lVirtualClassList =
00066         ioLegCabin.getVirtualClassList ();
00067     const stdair::CabinCapacity_T& lCabinCapacity =
00068         ioLegCabin.getOfferedCapacity();
00069
00075     stdair::VirtualClassList_T::iterator itVC = lVirtualClassList.begin();
00076     assert (itVC != lVirtualClassList.end());
00077
00078     stdair::VirtualClassStruct& lFirstVC = *itVC;
00079     lFirstVC.setCumulatedBookingLimit (lCabinCapacity);
00080     ++itVC;
00081     stdair::VirtualClassStruct lAggregatedVC = lFirstVC;
00082     for (; itVC != lVirtualClassList.end(); ++itVC) {
00083         stdair::VirtualClassStruct& lNextVC = *itVC;
00084
00085         // Compute the protection level for the aggregated class/bucket
00086         // using the Little-Wood formular.
00087         const stdair::ProtectionLevel_T lProtectionLevel =
00088             EmsrUtils::computeProtectionLevel (lAggregatedVC, lNextVC);
00089
00090         // Set the protection level for class/bucket j.
00091         stdair::VirtualClassList_T::iterator itCurrentVC = itVC; --itCurrentVC;
00092         stdair::VirtualClassStruct& lCurrentVC = *itCurrentVC;
00093         lCurrentVC.setCumulatedProtection (lProtectionLevel);
00094
00095         // Compute the booking limit for class/bucket j+1 (can be negative).

```

```

00096         const double lBookingLimit = lCabinCapacity - lProtectionLevel;
00097
00098         // Set the booking limit for class/bucket j+1.
00099         lNextVC.setCumulatedBookingLimit (lBookingLimit);
00100
00101         // Compute the aggregated class/bucket of classes/buckets 1,...,j.
00102         EmsrUtils::computeAggregatedVirtualClass (lAggregatedVC, lNextVC);
00103     }
00104 }
00105
00106
00107 // //////////////////////////////////////
00108 void Emsr::heuristicOptimisationByEmsr (stdair::LegCabin& ioLegCabin) {
00109     stdair::VirtualClassList_T& lVirtualClassList =
00110         ioLegCabin.getVirtualClassList ();
00111     const stdair::CabinCapacity_T& lCapacity = ioLegCabin.getOfferedCapacity();
00112     ioLegCabin.emptyBidPriceVector();
00113     stdair::BidPriceVector_T& lBidPriceVector =
00114         ioLegCabin.getBidPriceVector();
00115
00116     // Cabin capacity in integer.
00117     const int lCabinCapacity = static_cast<const int> (lCapacity);
00118
00119     // List of all EMSR values.
00120     stdair::EmsrValueList_T lEmsrValueList;
00121
00122     for (stdair::VirtualClassList_T::iterator itVC = lVirtualClassList.begin();
00123          itVC != lVirtualClassList.end(); ++itVC) {
00124         stdair::VirtualClassStruct& lCurrentVC = *itVC;
00125         for (int k = 1; k <= lCabinCapacity; ++k) {
00126             const double emsrValue = EmsrUtils::computeEmsrValue (k, lCurrentVC);
00127             lEmsrValueList.push_back(emsrValue);
00128         }
00129     }
00130
00131     // Sort the EMSR values from high to low.
00132     std::sort(lEmsrValueList.rbegin(), lEmsrValueList.rend());
00133
00134     // Sanity check
00135     const int lEmsrValueListSize = lEmsrValueList.size();
00136     assert (lEmsrValueListSize >= lCabinCapacity);
00137
00138     // Copy the EMSR sorted values to the BPV.
00139     stdair::EmsrValueList_T::const_iterator itCurrentValue =
00140         lEmsrValueList.begin();
00141     for (int j = 0; j < lCabinCapacity; ++j, ++itCurrentValue) {
00142         const double lBidPrice = *itCurrentValue;
00143         lBidPriceVector.push_back (lBidPrice);
00144     }
00145     lEmsrValueList.clear();
00146
00147     // Build the protection levels and booking limits.
00148     if (lVirtualClassList.size() > 1) {
00149         int lCapacityIndex = 0;
00150         for (stdair::VirtualClassList_T::iterator itVC = lVirtualClassList.begin();
00151              itVC != lVirtualClassList.end(); ++itVC) {
00152             stdair::VirtualClassStruct& lCurrentVC = *itVC;
00153             if (itVC != lVirtualClassList.end()) {
00154                 ++itVC;
00155             }
00156             stdair::VirtualClassStruct& lNextVC = *itVC;
00157             const stdair::Yield_T lNextYield = lNextVC.getYield();

```

```

00163         while ((lCapacityIndex < lCabinCapacity)
00164                && (lBidPriceVector.at(lCapacityIndex) > lNextYield)) {
00165             ++lCapacityIndex;
00166         }
00167         lCurrentVC.setCumulatedProtection (lCapacityIndex);
00168         lNextVC.setCumulatedBookingLimit (lCapacity - lCapacityIndex);
00169     }
00170 }
00171 }
00172
00173 }
```

43.84 rmol/bom/Emsr.hpp File Reference

```
#include <rmol/RMOL_Types.hpp>
```

Classes

- class [RMOL::Emsr](#)

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.85 Emsr.hpp

```

00001 #ifndef __RMOL_EMSR_HPP
00002 #define __RMOL_EMSR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // RMOL
00008 #include <rmol/RMOL_Types.hpp>
00009
00010 namespace stdair {
00011     class LegCabin;
00012 }
00013
00014 namespace RMOL {
00015     class Emsr {
00016     public:
00017         static void heuristicOptimisationByEmsr (stdair::LegCabin&);
00018         static void heuristicOptimisationByEmsrA (stdair::LegCabin&);
00019         static void heuristicOptimisationByEmsrB (stdair::LegCabin&);
00020     };
00021 }
00022 #endif // __RMOL_EMSR_HPP
```

43.86 rmol/bom/EmsrUtils.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <boost/math/distributions/normal.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <rmol/bom/EmsrUtils.hpp>
#include <rmol/basic/BasConst_General.hpp>
```

Namespaces

- namespace [RMOL](#)

43.87 EmsrUtils.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <cmath>
00007 // Boost Math
00008 #include <boost/math/distributions/normal.hpp>
00009 // StdAir
00010 #include <stdair/stdair_maths_types.hpp>
00011 #include <stdair/bom/VirtualClassStruct.hpp>
00012 // RMOL
00013 #include <rmol/bom/EmsrUtils.hpp>
00014 #include <rmol/basic/BasConst_General.hpp>
00015
00016 namespace RMOL {
00017 // //////////////////////////////////////
00018 void EmsrUtils::computeAggregatedVirtualClass
00019 (stdair::VirtualClassStruct& ioAggregatedVirtualClass,
00020  stdair::VirtualClassStruct& ioCurrentVirtualClass) {
00021     // Retrieve the demand mean, demand standard deviation and average
00022     // yield of the classes/buckets.
00023     const stdair::MeanValue_T lAggregatedMean =
00024         ioAggregatedVirtualClass.getMean();
00025     const stdair::MeanValue_T lCurrentMean = ioCurrentVirtualClass.getMean();
00026     const stdair::StdDevValue_T lAggregatedSD =
00027         ioAggregatedVirtualClass.getStdDev();
00028     const stdair::StdDevValue_T lCurrentSD = ioCurrentVirtualClass.getStdDev();
00029     const stdair::Yield_T lAggregatedYield =
00030         ioAggregatedVirtualClass.getYield();
00031     const stdair::Yield_T lCurrentYield = ioCurrentVirtualClass.getYield();
00032
00033     // Compute the new demand mean, new demand standard deviation and
00034     // new average yield for the new aggregated class/bucket.
00035     const stdair::MeanValue_T lNewMean = lAggregatedMean + lCurrentMean;
00036     const stdair::StdDevValue_T lNewSD =
00037         sqrt (lAggregatedSD*lAggregatedSD + lCurrentSD*lCurrentSD);
```



```

00038     stdair::Yield_T lNewYield = lCurrentYield;
00039     if (lNewMean > 0) {
00040         lNewYield = (lAggregatedYield*lAggregatedMean +
00041             lCurrentYield*lCurrentMean) / lNewMean;
00042     }
00043     // Set the new yield range for the new aggregated class/bucket.
00044     ioAggregatedVirtualClass.setYield(lNewYield);
00045
00046     // Set the new demand for the new aggregated class/bucket.
00047     ioAggregatedVirtualClass.setMean (lNewMean);
00048     ioAggregatedVirtualClass.setStdDev (lNewSD);
00049 }
00050
00051 // //////////////////////////////////////
00052 const stdair::ProtectionLevel_T EmsrUtils::
00053 computeProtectionLevel (stdair::VirtualClassStruct& ioAggregatedVirtualClass,
00054     stdair::VirtualClassStruct& ioNextVirtualClass) {
00055     // Retrieve the mean & standard deviation of the aggregated
00056     // class/bucket and the average yield of all the two
00057     // classes/buckets.
00058     const stdair::MeanValue_T lMean = ioAggregatedVirtualClass.getMean();
00059     const stdair::StdDevValue_T lSD = ioAggregatedVirtualClass.getStdDev();
00060     const stdair::Yield_T lAggregatedYield = ioAggregatedVirtualClass.getYield();
00061     const stdair::Yield_T lNextYield = ioNextVirtualClass.getYield();
00062     assert (lAggregatedYield != 0);
00063
00064     // Compute the yield ratio between the higher bucket and the current one
00065     const double lYieldRatio = lNextYield / lAggregatedYield;
00066
00067     boost::math::normal lNormalDistribution (lMean, lSD);
00071     const stdair::ProtectionLevel_T lProtection =
00072         boost::math::quantile (boost::math::complement (lNormalDistribution,
00073             lYieldRatio));
00074
00075     return lProtection;
00076 }
00077
00078 // //////////////////////////////////////
00079 const double EmsrUtils::
00080 computeEmsrValue (double iCapacity,
00081     stdair::VirtualClassStruct& ioVirtualClass){
00082     // Retrieve the average yield, mean and standard deviation of the
00083     // demand of the class/bucket.
00084     const stdair::MeanValue_T lMean = ioVirtualClass.getMean();
00085     const stdair::StdDevValue_T lSD = ioVirtualClass.getStdDev();
00086     const stdair::Yield_T lYield = ioVirtualClass.getYield();
00087
00088     // Compute the EMSR value = lYield * Pr (demand >= iCapacity).
00089     boost::math::normal lNormalDistribution (lMean, lSD);
00090     const double emsrValue =
00091         lYield * boost::math::cdf (boost::math::complement (lNormalDistribution,
00092             iCapacity));
00093
00094     return emsrValue;
00095 }
00096 }

```

43.88 rmol/bom/EmsrUtils.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
```

Classes

- class [RMOL::EmsrUtils](#)

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.89 EmsrUtils.hpp

```

00001 #ifndef __RMOL_EMSRUTILS_HPP
00002 #define __RMOL_EMSRUTILS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_inventory_types.hpp>
00009
00010 // Forward declarations.
00011 namespace stdair {
00012     struct VirtualClassStruct;
00013 }
00014
00015 namespace RMOL {
00016
00017     class EmsrUtils {
00018     public:
00019         static void computeAggregatedVirtualClass (stdair::VirtualClassStruct&,
00020                                                     stdair::VirtualClassStruct&);
00021
00022         static const stdair::ProtectionLevel_T computeProtectionLevel (stdair::VirtualClassStruct&, stdair::VirtualClassStruct&);
00023
00024         static const double computeEmsrValue (double, stdair::VirtualClassStruct&);
00025     };
00026 }
00027
00028 #endif // __RMOL_EMSRUTILS_HPP

```

43.90 rmol/bom/GuillotineBlockHelper.cpp File Reference

```

#include <cassert>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/GuillotineBlock.hpp>
#include <stdair/service/Logger.hpp>

```

```
#include <rmol/bom/GuillotineBlockHelper.hpp>
```

Namespaces

- namespace [RMOL](#)

43.91 GuillotineBlockHelper.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/basic/BasConst_Inventory.hpp>
00008 #include <stdair/bom/BomManager.hpp>
00009 #include <stdair/bom/SegmentDate.hpp>
00010 #include <stdair/bom/SegmentCabin.hpp>
00011 #include <stdair/bom/BookingClass.hpp>
00012 #include <stdair/bom/GuillotineBlock.hpp>
00013 #include <stdair/service/Logger.hpp>
00014 // RMOL
00015 #include <rmol/bom/GuillotineBlockHelper.hpp>
00016
00017 namespace RMOL {
00018     // //////////////////////////////////////
00019     stdair::NbOfSegments_T GuillotineBlockHelper::
00020     getNbOfSegmentAlreadyPassedThisDTD (const stdair::GuillotineBlock& iGB,
00021                                         const stdair::DTD_T& iDTD,
00022                                         const stdair::Date_T& iCurrentDate) {
00023         stdair::NbOfSegments_T oNbOfSegments = 0;
00024
00025         // Browse the list of segments and check if it has passed the given DTD.
00026         const stdair::SegmentCabinIndexMap_T& lSCMap=iGB.getSegmentCabinIndexMap();
00027         for (stdair::SegmentCabinIndexMap_T::const_iterator itSC = lSCMap.begin();
00028             itSC != lSCMap.end(); ++itSC) {
00029             const stdair::SegmentCabin* lSC_ptr = itSC->first;
00030             assert (lSC_ptr != NULL);
00031
00032             if (hasPassedThisDTD (*lSC_ptr, iDTD, iCurrentDate) == true) {
00033                 ++oNbOfSegments;
00034             }
00035         }
00036
00037         return oNbOfSegments;
00038     }
00039
00040     // //////////////////////////////////////
00041     bool GuillotineBlockHelper::
00042     hasPassedThisDTD (const stdair::SegmentCabin& iSegmentCabin,
00043                     const stdair::DTD_T& iDTD,
00044                     const stdair::Date_T& iCurrentDate) {
00045         // Retrieve the boarding date.
00046         const stdair::SegmentDate& lSegmentDate =
00047             stdair::BomManager::getParent<stdair::SegmentDate> (iSegmentCabin);
00048         const stdair::Date_T& lBoardingDate = lSegmentDate.getBoardingDate();
00049
00050         // Compare the date offset between the boarding date and the current date
00051         // to the DTD.
```

```

00052     stdair::DateOffset_T lDateOffset = lBoardingDate - iCurrentDate;
00053     stdair::DTD_T lDateOffsetInDays = lDateOffset.days();
00054     if (iDTD < lDateOffsetInDays) {
00055         return false;
00056     } else {
00057         return true;
00058     }
00059 }
00060 }

```

43.92 rmol/bom/GuillotineBlockHelper.hpp File Reference

```

#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_date_time_types.hpp>

```

Classes

- class [RMOL::GuillotineBlockHelper](#)

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.93 GuillotineBlockHelper.hpp

```

00001 #ifndef __RMOL_BOM_GUILLOTINEBLOCKHELPER_HPP
00002 #define __RMOL_BOM_GUILLOTINEBLOCKHELPER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/stdair_date_time_types.hpp>
00012
00013 // Forward declarations
00014 namespace stdair {
00015     class GuillotineBlock;
00016     class SegmentCabin;
00017 }
00018
00019 namespace RMOL {
00020
00023     class GuillotineBlockHelper {
00024     public:
00025         // ////////////////////////////////// Business Methods //////////////////////////////////
00030         static stdair::NbOfSegments_T getNbOfSegmentAlreadyPassedThisDTD (const stdai
            r::GuillotineBlock&, const stdair::DTD_T&, const stdair::Date_T&);

```

```

00031
00035     static bool hasPassedThisDTD (const stdair::SegmentCabin&,
00036                                   const stdair::DTD_T&, const stdair::Date_T&);
00037 };
00038
00039 }
00040 #endif // __RMOL_BOM_GUILLOTINEBLOCKHELPER_HPP

```

43.94 rmol/bom/HistoricalBooking.cpp File Reference

```

#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
#include <rmol/bom/HistoricalBooking.hpp>

```

Namespaces

- namespace [RMOL](#)

43.95 HistoricalBooking.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 #include <iomanip>
00008 #include <iostream>
00009 // RMOL
00010 #include <rmol/bom/HistoricalBooking.hpp>
00011
00012 namespace RMOL {
00013
00014 // //////////////////////////////////////
00015 HistoricalBooking::HistoricalBooking () :
00016     _numberOfBookings (0.0), _unconstrainedDemand (0.0), _flag (false) {
00017 }
00018
00019 // //////////////////////////////////////
00020 HistoricalBooking::
00021 HistoricalBooking (const stdair::NbOfBookings_T iNbOfBookings,
00022                  const stdair::Flag_T iFlag)
00023     : _numberOfBookings (iNbOfBookings),
00024       _unconstrainedDemand (iNbOfBookings), _flag (iFlag) {
00025 }
00026
00027 // //////////////////////////////////////
00028 HistoricalBooking::HistoricalBooking
00029 (const HistoricalBooking& iHistoricalBooking) :
00030     _numberOfBookings (iHistoricalBooking.getNbOfBookings()),
00031     _unconstrainedDemand (iHistoricalBooking.getUnconstrainedDemand()),

```

```

00032     _flag (iHistoricalBooking.getFlag()) {
00033     }
00034
00035     // //////////////////////////////////////
00036     HistoricalBooking::~HistoricalBooking() {
00037     }
00038
00039     // //////////////////////////////////////
00040     void HistoricalBooking::setParameters
00041     (const stdair::NbOfBookings_T iNbOfBookings, const stdair::Flag_T iFlag) {
00042         _numberOfBookings = iNbOfBookings;
00043         _unconstrainedDemand = iNbOfBookings;
00044         _flag = iFlag;
00045     }
00046
00047     // //////////////////////////////////////
00048     const std::string HistoricalBooking::describe() const {
00049         std::ostringstream ostr;
00050         ostr << "Struct of hitorical booking, unconstrained demand and flag of "
00051             << "censorship for a FlightDate/Class.";
00052
00053         return ostr.str();
00054     }
00055
00056     // //////////////////////////////////////
00057     void HistoricalBooking::toStream (std::ostream& ioOut) const {
00058         const stdair::NbOfBookings_T bj = getNbOfBookings();
00059         const stdair::NbOfBookings_T uj = getUnconstrainedDemand();
00060         const stdair::Flag_T fj = getFlag();
00061         ioOut << std::fixed << std::setprecision (2)
00062             << bj << "; " << uj << "; " << fj << std::endl;
00063     }
00064
00065     // //////////////////////////////////////
00066     void HistoricalBooking::display () const {
00067         toStream (std::cout);
00068     }
00069 }

```

43.96 rmol/bom/HistoricalBooking.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

```

Classes

- struct [RMOL::HistoricalBooking](#)

Structure keeping track, for a given class, of the number of historical bookings and of the censorship flag.

Namespaces

- namespace [RMOL](#)

43.97 HistoricalBooking.hpp

```

00001 #ifndef __RMOL_BOM_HISTORICALBOOKING_HPP
00002 #define __RMOL_BOM_HISTORICALBOOKING_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/basic/StructAbstract.hpp>
00010
00011 namespace RMOL {
00012
00017     struct HistoricalBooking : public stdair::StructAbstract {
00018
00019     public:
00020         // ////////////////////////////////// Getters //////////////////////////////////
00022         const stdair::NbOfBookings_T& getNbOfBookings() const {
00023             return _numberOfBookings;
00024         }
00026         const stdair::NbOfBookings_T& getUnconstrainedDemand() const {
00027             return _unconstrainedDemand;
00028         }
00031         const stdair::Flag_T& getFlag() const {
00032             return _flag;
00033         }
00034
00035     public:
00036         // ////////////////////////////////// Setters //////////////////////////////////
00038         void setUnconstrainedDemand (const stdair::NbOfBookings_T& iDemand) {
00039             _unconstrainedDemand = iDemand;
00040         }
00041
00043         void setParameters (const stdair::NbOfBookings_T, const stdair::Flag_T);
00044
00045     public:
00046         // ////////////////////////////////// Display Methods //////////////////////////////////
00052         void toStream (std::ostream& ioOut) const;
00053
00057         const std::string describe() const;
00058
00062         void display () const;
00063
00064     public:
00065         // ////////////////////////////////// Constructors and destructor. //////////////////////////////////
00069         HistoricalBooking (const stdair::NbOfBookings_T, const stdair::Flag_T);
00073         HistoricalBooking ();
00077         HistoricalBooking (const HistoricalBooking&);
00078
00082         virtual ~HistoricalBooking ();
00083
00084     private:
00085         // ////////////////////////////////// Attributes //////////////////////////////////
00089         stdair::NbOfBookings_T _numberOfBookings;
00090
00094         stdair::NbOfBookings_T _unconstrainedDemand;
00095
00099         stdair::Flag_T _flag;
00100     };
00101 }
00102 #endif // __RMOL_BOM_HISTORICALBOOKING_HPP

```

43.98 rmol/bom/HistoricalBookingHolder.cpp File Reference

```

#include <sstream>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <cassert>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>

```

Namespaces

- namespace [RMOL](#)

43.99 HistoricalBookingHolder.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <iostream>
00007 #include <iomanip>
00008 #include <cmath>
00009 #include <cassert>
00010 // StdAir
00011 #include <stdair/service/Logger.hpp>
00012 // RMOL
00013 #include <rmol/bom/HistoricalBooking.hpp>
00014 #include <rmol/bom/HistoricalBookingHolder.hpp>
00015
00016 namespace RMOL {
00017
00018 // //////////////////////////////////////
00019 HistoricalBookingHolder::HistoricalBookingHolder () {
00020 }
00021
00022 // //////////////////////////////////////
00023 HistoricalBookingHolder::~HistoricalBookingHolder () {
00024     _historicalBookingVector.clear();
00025 }
00026
00027 // //////////////////////////////////////
00028 const short HistoricalBookingHolder::getNbOfFlights () const {
00029     return _historicalBookingVector.size();
00030 }
00031
00032 // //////////////////////////////////////
00033 const short HistoricalBookingHolder::getNbOfUncensoredData () const {
00034     short lResult = 0;
00035     const short lSize = _historicalBookingVector.size();

```



```

00036
00037     for (short ite = 0; ite < lSize; ++ite) {
00038         const stdair::Flag_T lFlag = _historicalBookingVector.at(ite).getFlag ();
00039         if (lFlag == false) {
00040             ++ lResult;
00041         }
00042     }
00043
00044     return lResult;
00045 }
00046
00047 // //////////////////////////////////////
00048 const stdair::NbOfBookings_T HistoricalBookingHolder::
00049 getNbOfUncensoredBookings () const {
00050     stdair::NbOfBookings_T lResult = 0;
00051     const short lSize = _historicalBookingVector.size();
00052
00053     for (short ite = 0; ite < lSize; ++ite) {
00054         const HistoricalBooking& lHistorialBooking =
00055             _historicalBookingVector.at (ite);
00056         const stdair::Flag_T lFlag = lHistorialBooking.getFlag ();
00057         if (lFlag == false) {
00058             const stdair::NbOfBookings_T& lBooking =
00059                 lHistorialBooking.getNbOfBookings ();
00060             lResult += lBooking;
00061         }
00062     }
00063
00064     return lResult;
00065 }
00066
00067 // //////////////////////////////////////
00068 const double HistoricalBookingHolder::
00069 getUncensoredStandardDeviation (const double& iMeanOfUncensoredBookings,
00070                                 const short iNbOfUncensoredData) const {
00071
00072     double lResult = 0;
00073     const short lSize = _historicalBookingVector.size();
00074
00075     for (short ite = 0; ite < lSize; ++ite) {
00076         const stdair::Flag_T lFlag = _historicalBookingVector.at(ite).getFlag ();
00077         if (lFlag == false) {
00078             const HistoricalBooking& lHistorialBooking =
00079                 _historicalBookingVector.at (ite);
00080
00081             const stdair::NbOfBookings_T& lBooking =
00082                 lHistorialBooking.getNbOfBookings ();
00083
00084             lResult += (lBooking - iMeanOfUncensoredBookings)
00085                 * (lBooking - iMeanOfUncensoredBookings);
00086         }
00087     }
00088     lResult /= (iNbOfUncensoredData - 1);
00089     lResult = sqrt (lResult);
00090
00091     return lResult;
00092 }
00093
00094 // //////////////////////////////////////
00095 const double HistoricalBookingHolder::getDemandMean () const {
00096     double lResult = 0;
00097     const short lSize = _historicalBookingVector.size();

```

```

00098
00099     for (short ite = 0; ite < lSize; ++ite) {
00100         const HistoricalBooking& lHistorialBooking =
00101             _historicalBookingVector.at(ite);
00102
00103         const stdair::NbOfBookings_T& lDemand =
00104             lHistorialBooking.getUnconstrainedDemand ();
00105
00106         lResult += static_cast<double>(lDemand);
00107     }
00108
00109     lResult /= lSize;
00110
00111     return lResult;
00112 }
00113
00114 // //////////////////////////////////////
00115 const double HistoricalBookingHolder::getStandardDeviation
00116 (const double iDemandMean) const {
00117     double lResult = 0;
00118     const short lSize = _historicalBookingVector.size();
00119
00120     for (short ite = 0; ite < lSize; ++ite) {
00121         const HistoricalBooking& lHistorialBooking =
00122             _historicalBookingVector.at(ite);
00123
00124         const stdair::NbOfBookings_T& lDemand =
00125             lHistorialBooking.getUnconstrainedDemand ();
00126
00127         const double lDoubleDemand = static_cast<double> (lDemand);
00128         lResult += (lDoubleDemand - iDemandMean) * (lDoubleDemand - iDemandMean);
00129     }
00130
00131     lResult /= (lSize - 1);
00132
00133     lResult = sqrt (lResult);
00134
00135     return lResult;
00136 }
00137
00138 // //////////////////////////////////////
00139 const std::vector<bool> HistoricalBookingHolder::
00140 getListOfToBeUnconstrainedFlags () const {
00141     std::vector<bool> lResult;
00142     const short lSize = _historicalBookingVector.size();
00143
00144     for (short ite = 0; ite < lSize; ++ite) {
00145         const HistoricalBooking& lHistorialBooking =
00146             _historicalBookingVector.at(ite);
00147         const stdair::Flag_T lFlag = lHistorialBooking.getFlag ();
00148         if (lFlag == true) {
00149             lResult.push_back(true);
00150         }
00151         else {
00152             lResult.push_back(false);
00153         }
00154     }
00155
00156     return lResult;
00157 }
00158
00159 // //////////////////////////////////////

```

```

00160     const stdair::NbOfBookings_T& HistoricalBookingHolder::
00161     getHistoricalBooking (const short i) const {
00162         const HistoricalBooking& lHistorialBooking =
00163             _historicalBookingVector.at(i);
00164         return lHistorialBooking.getNbOfBookings();
00165     }
00166
00167     // //////////////////////////////////////
00168     const stdair::NbOfBookings_T& HistoricalBookingHolder::
00169     getUnconstrainedDemand (const short i) const {
00170         const HistoricalBooking& lHistorialBooking =
00171             _historicalBookingVector.at(i);
00172         return lHistorialBooking.getUnconstrainedDemand();
00173     }
00174
00175     // //////////////////////////////////////
00176     const stdair::Flag_T& HistoricalBookingHolder::
00177     getCensorshipFlag (const short i) const {
00178         const HistoricalBooking& lHistorialBooking =
00179             _historicalBookingVector.at(i);
00180         return lHistorialBooking.getFlag();
00181     }
00182
00183     // //////////////////////////////////////
00184     void HistoricalBookingHolder::setUnconstrainedDemand
00185     (const stdair::NbOfBookings_T& iExpectedDemand, const short i) {
00186         _historicalBookingVector.at(i).setUnconstrainedDemand(iExpectedDemand);
00187     }
00188
00189     // //////////////////////////////////////
00190     const stdair::NbOfBookings_T HistoricalBookingHolder::calculateExpectedDemand
00191     (const double iMean, const double iSD,
00192      const short i, const stdair::NbOfBookings_T iDemand) const {
00193
00194         const HistoricalBooking lHistorialBooking =
00195             _historicalBookingVector.at(i);
00196         const double lBooking =
00197             static_cast<double> (lHistorialBooking.getNbOfBookings());
00198         double e, d1, d2;
00199
00200         e = - (lBooking - iMean) * (lBooking - iMean) * 0.625 / (iSD * iSD);
00201         //STDAIR_LOG_DEBUG ("e: " << e);
00202         e = exp (e);
00203         //STDAIR_LOG_DEBUG ("e: " << e);
00204
00205         double s = sqrt (1 - e);
00206         //STDAIR_LOG_DEBUG ("s: " << s);
00207
00208         if (lBooking >= iMean) {
00209             if (e < 0.01) {
00210                 return iDemand;
00211             }
00212             d1 = 0.5 * (1 - s);
00213         }
00214         else {
00215             d1 = 0.5 * (1 + s);
00216         }
00217         //STDAIR_LOG_DEBUG ("d1: " << d1);
00218
00219         e = - (lBooking - iMean) * (lBooking - iMean) * 0.5 / (iSD * iSD);
00220         e = exp (e);
00221         d2 = e * iSD / sqrt (2 * 3.14159265);

```

```

00222     //STDAIR_LOG_DEBUG ("d2: " << d2);
00223
00224     if (d1 == 0) {
00225         return iDemand;
00226     }
00227
00228     const stdair::NbOfBookings_T lDemand =
00229         static_cast<stdair::NbOfBookings_T> (iMean + d2/d1);
00230
00231     return lDemand;
00232 }
00233
00234 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00235 void HistoricalBookingHolder::addHistoricalBooking
00236 (const HistoricalBooking& iHistoricalBooking) {
00237     _historicalBookingVector.push_back(iHistoricalBooking);
00238 }
00239
00240 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00241 void HistoricalBookingHolder::toStream (std::ostream& ioOut) const {
00242     const short lSize = _historicalBookingVector.size();
00243
00244     ioOut << "Historical Booking; Unconstrained Demand; Flag" << std::endl;
00245
00246     for (short ite = 0; ite < lSize; ++ite) {
00247         const HistoricalBooking& lHistorialBooking =
00248             _historicalBookingVector.at(ite);
00249
00250         const stdair::NbOfBookings_T& lBooking =
00251             lHistorialBooking.getNbOfBookings();
00252
00253         const stdair::NbOfBookings_T& lDemand =
00254             lHistorialBooking.getUnconstrainedDemand();
00255
00256         const stdair::Flag_T lFlag = lHistorialBooking.getFlag();
00257
00258         ioOut << lBooking << "      "
00259             << lDemand << "      "
00260             << lFlag << std::endl;
00261     }
00262 }
00263
00264 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00265 const std::string HistoricalBookingHolder::describe() const {
00266     std::ostringstream ostr;
00267     ostr << "Holder of HistoricalBooking structs.";
00268
00269     return ostr.str();
00270 }
00271
00272 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00273 void HistoricalBookingHolder::display() const {
00274     toStream (std::cout);
00275 }
00276 }

```

43.100 rmol/bom/HistoricalBookingHolder.hpp File Reference

```
#include <iostream>
```

```
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct [RMOL::HistoricalBookingHolder](#)

Namespaces

- namespace [RMOL](#)

Typedefs

- typedef std::vector< HistoricalBooking > [RMOL::HistoricalBookingVector_T](#)

43.101 HistoricalBookingHolder.hpp

```
00001 #ifndef __RMOL_BOM_HISTORICALBOOKINGHOLDER_HPP
00002 #define __RMOL_BOM_HISTORICALBOOKINGHOLDER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <iostream>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013
00014 namespace RMOL {
00015     struct HistoricalBooking;
00016
00017     typedef std::vector<HistoricalBooking> HistoricalBookingVector_T;
00018
00019     struct HistoricalBookingHolder : public stdair::StructAbstract {
00020     public:
00021         // ===== Getters =====
00022         const short getNbOfFlights () const;
00023
00024         const short getNbOfUncensoredData () const;
00025
00026         const stdair::NbOfBookings_T getNbOfUncensoredBookings () const;
00027
00028         const double getUncensoredStandardDeviation
00029             (const double& iMeanOfUncensoredBookings,
00030              const short iNbOfUncensoredData) const;
00031
00032         const double getDemandMean () const;
00033
00034         const double getStandardDeviation (const double) const;
```

```

00046
00048     const std::vector<bool> getListOfToBeUnconstrainedFlags() const;
00049
00051     const stdair::NbOfBookings_T& getHistoricalBooking (const short i) const;
00052
00054     const stdair::NbOfBookings_T& getUnconstrainedDemand (const short i) const;
00055
00057     const stdair::Flag_T& getCensorshipFlag (const short i) const;
00058
00060     const stdair::NbOfBookings_T& getUnconstrainedDemandOnFirstElement() const {
00061         return getUnconstrainedDemand (0);
00062     }
00063
00065     const stdair::NbOfBookings_T calculateExpectedDemand (const double,
00066                                                            const double,
00067                                                            const short,
00068                                                            const stdair::NbOfBookings_T) const;
00069
00071     void setUnconstrainedDemand (const stdair::NbOfBookings_T& iExpectedDemand,
00072                                 const short i);
00073
00075     void addHistoricalBooking (const HistoricalBooking& iHistoricalBooking);
00076
00080     void toStream (std::ostream& ioOut) const;
00081
00082     // /////////// Display Methods ///////////
00084     const std::string describe() const;
00085
00087     void display () const;
00088
00090     virtual ~HistoricalBookingHolder();
00091
00092     public:
00095         HistoricalBookingHolder ();
00096
00097     private:
00099         HistoricalBookingVector_T _historicalBookingVector;
00100
00101     protected:
00102     };
00103 }
00104 #endif // __RMOL_BOM_HISTORICALBOOKINGHOLDER_HPP
00105

```

43.102 rmol/bom/MCOptimiser.cpp File Reference

```

#include <cassert>
#include <string>
#include <sstream>
#include <algorithm>
#include <cmath>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegCabin.hpp>

```

```
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <rmol/bom/MCOptimiser.hpp>
```

Namespaces

- namespace [RMOL](#)

43.103 MCOptimiser.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 #include <sstream>
00008 #include <algorithm>
00009 #include <cmath>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/LegCabin.hpp>
00014 #include <stdair/bom/SegmentCabin.hpp>
00015 #include <stdair/bom/BookingClass.hpp>
00016 #include <stdair/bom/VirtualClassStruct.hpp>
00017 #include <stdair/service/Logger.hpp>
00018
00019 #include <stdair/basic/RandomGeneration.hpp>
00020 #include <stdair/basic/BasConst_General.hpp>
00021 // RMOL
00022 #include <rmol/bom/MCOptimiser.hpp>
00023
00024 namespace RMOL {
00025
00026 // // //////////////////////////////////////
00027 void MCOptimiser::
00028 optimalOptimisationByMCIntegration (stdair::LegCabin& ioLegCabin) {
00029     // Retrieve the segment-cabin
00030     const stdair::SegmentCabinList_T lSegmentCabinList =
00031         stdair::BomManager::getList<stdair::SegmentCabin> (ioLegCabin);
00032     stdair::SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00033     assert (itSC != lSegmentCabinList.end());
00034     const stdair::SegmentCabin* lSegmentCabin_ptr = *itSC;
00035     assert (lSegmentCabin_ptr != NULL);
00036
00037     // Retrieve the class list.
00038     const stdair::BookingClassList_T lBookingClassList =
00039         stdair::BomManager::getList<stdair::BookingClass> (*lSegmentCabin_ptr);
```

```

00040
00041 // Retrieve the remaining cabin capacity.
00042 const stdair::Availability_T& lCap = ioLegCabin.getAvailabilityPool();
00043 const int lCapacity = static_cast<const int> (lCap);
00044 const stdair::UnsignedIndex_T lCapacityIndex =
00045     static_cast<const stdair::UnsignedIndex_T> ((lCapacity+abs(lCapacity))/2);
00046
00047 // Retrieve the virtual class list.
00048 stdair::VirtualClassList_T& lVCLList = ioLegCabin.getVirtualClassList();
00049
00050 // Parse the virtual class list and compute the protection levels.
00051 stdair::VirtualClassList_T::iterator itCurrentVC = lVCLList.begin();
00052 assert (itCurrentVC != lVCLList.end());
00053 stdair::VirtualClassList_T::iterator itNextVC = itCurrentVC; ++itNextVC;
00054
00055 // Initialise the partial sum holder with the demand sample of the first
00056 // virtual class.
00057 stdair::VirtualClassStruct& lFirstVC = *itCurrentVC;
00058 stdair::GeneratedDemandVector_T lPartialSumHolder =
00059     lFirstVC.getGeneratedDemandVector();
00060
00061 // Initialise the booking limit for the first class, which is equal to
00062 // the remaining capacity.
00063 lFirstVC.setCumulatedBookingLimit (lCap);
00064
00065 // Initialise bid price vector with the first element (index 0) equal to
00066 // the highest yield.
00067 ioLegCabin.emptyBidPriceVector();
00068 stdair::BidPriceVector_T& lBPV = ioLegCabin.getBidPriceVector();
00069 //const stdair::Yield_T& y1 = lFirstVC.getYield ();
00070 //lBPV.push_back (y1);
00071 stdair::UnsignedIndex_T idx = 1;
00072
00073 for (; itNextVC != lVCLList.end(); ++itCurrentVC, ++itNextVC) {
00074     // Get the yields of the two classes.
00075     stdair::VirtualClassStruct& lCurrentVC = *itCurrentVC;
00076     stdair::VirtualClassStruct& lNextVC = *itNextVC;
00077     const stdair::Yield_T& yj = lCurrentVC.getYield ();
00078     const stdair::Yield_T& yj1 = lNextVC.getYield ();
00079
00080     // Consistency check: the yield/price of a higher class/bucket
00081     // (with the j index lower) must be higher.
00082     assert (yj > yj1);
00083
00084     // Sort the partial sum holder.
00085     std::sort (lPartialSumHolder.begin(), lPartialSumHolder.end());
00086     const stdair::UnsignedIndex_T K = lPartialSumHolder.size ();
00087
00088     // Compute the optimal index lj = floor {[y(j)-y(j+1)]/y(j) . K}
00089     const double ljdouble = std::floor (K * (yj - yj1) / yj);
00090     stdair::UnsignedIndex_T lj =
00091         static_cast<stdair::UnsignedIndex_T> (ljdouble);
00092
00093     // Consistency check.
00094     assert (lj >= 1 && lj < K);
00095
00096     // The optimal protection: p(j) = 1/2 [S(j,lj) + S(j, lj+1)]
00097     const double sjl = lPartialSumHolder.at (lj - 1);
00098     const double sjlp1 = lPartialSumHolder.at (lj + 1 - 1);
00099     const double pj = (sjl + sjlp1) / 2;
00100
00101     // Set the cumulated protection level for the current class.

```



```

00102     lCurrentVC.setCumulatedProtection (pj);
00103     // Set the cumulated booking limit for the next class.
00104     lNextVC.setCumulatedBookingLimit (lCap - pj);
00105
00110     const stdair::UnsignedIndex_T pjint = static_cast<const int> (pj);
00111     stdair::GeneratedDemandVector_T::iterator itLowerBound =
00112         lPartialSumHolder.begin();
00113     for (; idx <= pjint && idx <= lCapacityIndex; ++idx) {
00114         itLowerBound =
00115             std::lower_bound (itLowerBound, lPartialSumHolder.end(), idx);
00116         const stdair::UnsignedIndex_T pos =
00117             itLowerBound - lPartialSumHolder.begin();
00118
00119         const stdair::BidPrice_T lBP = yj * (K - pos) / K;
00120         lBPV.push_back (lBP);
00121     }
00122
00123     // Update the partial sum holder.
00124     const stdair::GeneratedDemandVector_T& lNextPSH =
00125         lNextVC.getGeneratedDemandVector();
00126     assert (K <= lNextPSH.size());
00127     for (stdair::UnsignedIndex_T i = 0; i < K - lj; ++i) {
00128         lPartialSumHolder.at(i) = lPartialSumHolder.at(i + lj) + lNextPSH.at(i);
00129     }
00130     lPartialSumHolder.resize (K - lj);
00131 }
00132
00137     stdair::VirtualClassStruct& lLastVC = *itCurrentVC;
00138     const stdair::Yield_T& yn = lLastVC.getYield();
00139     stdair::GeneratedDemandVector_T::iterator itLowerBound =
00140         lPartialSumHolder.begin();
00141     for (; idx <= lCapacityIndex; ++idx) {
00142         itLowerBound =
00143             std::lower_bound (itLowerBound, lPartialSumHolder.end(), idx);
00144         const stdair::UnsignedIndex_T pos =
00145             itLowerBound - lPartialSumHolder.begin();
00146         const stdair::UnsignedIndex_T K = lPartialSumHolder.size();
00147         const stdair::BidPrice_T lBP = yn * (K - pos) / K;
00148         lBPV.push_back (lBP);
00149     }
00150 }
00151
00152 // //////////////////////////////////////
00153     stdair::GeneratedDemandVector_T MCOptimiser::
00154     generateDemandVector (const stdair::MeanValue_T& iMean,
00155                         const stdair::StdDevValue_T& iStdDev,
00156                         const unsigned int& K) {
00157         stdair::GeneratedDemandVector_T oDemandVector;
00158         if (iStdDev > 0) {
00159             stdair::RandomGeneration lGenerator (stdair::DEFAULT_RANDOM_SEED);
00160             for (unsigned int i = 0; i < K; ++i) {
00161                 stdair::RealNumber_T lDemandSample =
00162                     lGenerator.generateNormal (iMean, iStdDev);
00163                 oDemandVector.push_back (lDemandSample);
00164             }
00165         } else {
00166             for (unsigned int i = 0; i < K; ++i) {
00167                 oDemandVector.push_back (iMean);
00168             }
00169         }
00170         return oDemandVector;
00171     }

```

```

00172
00173 // //////////////////////////////////////
00174 void MCOptimiser::
00175 optimisationByMCIntegration (stdair::LegCabin& ioLegCabin) {
00176     // Number of MC samples
00177     unsigned int K = 100000;
00178
00179     const stdair::YieldLevelDemandMap_T& lYieldDemandMap =
00180         ioLegCabin.getYieldLevelDemandMap();
00181     assert (!lYieldDemandMap.empty());
00182
00183     std::ostringstream oStr;
00184     oStr << "Yield list ";
00185     for (stdair::YieldLevelDemandMap_T::const_iterator itYD =
00186         lYieldDemandMap.begin();
00187         itYD != lYieldDemandMap.end(); ++itYD) {
00188         const stdair::Yield_T& y = itYD->first;
00189         oStr << y << " ";
00190     }
00191
00192     STDAIR_LOG_DEBUG (oStr.str());
00193     ioLegCabin.emptyBidPriceVector();
00194     stdair::BidPriceVector_T& lBidPriceVector =
00195         ioLegCabin.getBidPriceVector();
00196     const stdair::Availability_T& lAvailabilityPool =
00197         ioLegCabin.getAvailabilityPool();
00198     // Initialise the minimal bid price to 1.0 (just to avoid problems
00199     // of division by zero).
00200     const stdair::BidPrice_T& lMinBP = 1.0;
00201
00202     stdair::YieldLevelDemandMap_T::const_reverse_iterator itCurrentYD =
00203         lYieldDemandMap.rbegin();
00204     stdair::YieldLevelDemandMap_T::const_reverse_iterator itNextYD = itCurrentYD;
00205
00206     ++itNextYD;
00207
00208     // Initialise the partial sum holder
00209     stdair::MeanStdDevPair_T lMeanStdDevPair = itCurrentYD->second;
00210     stdair::GeneratedDemandVector_T lPartialSumHolder =
00211         generateDemandVector(lMeanStdDevPair.first, lMeanStdDevPair.second, K);
00212
00213     stdair::UnsignedIndex_T idx = 1;
00214     for (; itNextYD!=lYieldDemandMap.rend(); ++itCurrentYD, ++itNextYD) {
00215         const stdair::Yield_T& yj = itCurrentYD->first;
00216         const stdair::Yield_T& yj1 = itNextYD->first;
00217         // Consistency check: the yield/price of a higher class/bucket
00218         // (with the j index lower) must be higher.
00219         assert (yj > yj1);
00220         // Sort the partial sum holder.
00221         std::sort (lPartialSumHolder.begin(), lPartialSumHolder.end());
00222         // STDAIR_LOG_DEBUG ("Partial sums : max = " << lPartialSumHolder.back()
00223         // << " min = " << lPartialSumHolder.front());
00224         K = lPartialSumHolder.size ();
00225         // Compute the optimal index lj = floor {[y(j)-y(j+1)]/y(j) . K}
00226         const double ljdoube = std::floor (K * (yj - yj1) / yj);
00227         stdair::UnsignedIndex_T lj =
00228             static_cast<stdair::UnsignedIndex_T> (ljdoube);
00229         // Consistency check.
00230         assert (lj >= 1 && lj < K);
00231         // The optimal protection: p(j) = 1/2 [S(j,lj) + S(j, lj+1)]
00232         const double sj1 = lPartialSumHolder.at (lj - 1);
00233         const double sjlp1 = lPartialSumHolder.at (lj + 1 - 1);

```

```

00233     const double pj = (sjl + sjlp1) / 2;
00238     const stdair::UnsignedIndex_T pjint = static_cast<const int> (pj);
00239     stdair::GeneratedDemandVector_T::iterator itLowerBound =
00240         lPartialSumHolder.begin();
00241     for (; idx <= pjint && idx <= lAvailabilityPool; ++idx) {
00242         itLowerBound =
00243             std::lower_bound (itLowerBound, lPartialSumHolder.end(), idx);
00244         const stdair::UnsignedIndex_T pos =
00245             itLowerBound - lPartialSumHolder.begin();
00246
00247         const stdair::BidPrice_T lBP = yj * (K - pos) / K;
00248         lBidPriceVector.push_back (lBP);
00249     }
00250     // Update the partial sum holder.
00251     lMeanStdDevPair = itNextYD->second;
00252     const stdair::GeneratedDemandVector_T& lNextDV =
00253         generateDemandVector (lMeanStdDevPair.first,
00254             lMeanStdDevPair.second, K - lj);
00255     for (stdair::UnsignedIndex_T i = 0; i < K - lj; ++i) {
00256         lPartialSumHolder.at(i) = lPartialSumHolder.at(i + lj) + lNextDV.at(i);
00257     }
00258     lPartialSumHolder.resize (K - lj);
00259 }
00268 // STDAIR_LOG_DEBUG ("Partial sums : max = " << lPartialSumHolder.back()
00269 //                   << " min = " << lPartialSumHolder.front());
00270
00271 std::sort (lPartialSumHolder.begin(), lPartialSumHolder.end());
00272 const stdair::Yield_T& yn = itCurrentYD->first;
00273 stdair::GeneratedDemandVector_T::iterator itLowerBound =
00274     lPartialSumHolder.begin();
00275 K = lPartialSumHolder.size();
00276
00277 bool lMinBPReached = false;
00278 for (; idx <= lAvailabilityPool; ++idx) {
00279     itLowerBound =
00280         std::lower_bound (itLowerBound, lPartialSumHolder.end(), idx);
00281
00282     if (!lMinBPReached) {
00283         const stdair::UnsignedIndex_T pos =
00284             itLowerBound - lPartialSumHolder.begin();
00285         stdair::BidPrice_T lBP = yn * (K - pos) / K;
00286
00287         if (lBP < lMinBP) {
00288             lBP = lMinBP; lMinBPReached = true;
00289         }
00290
00291         lBidPriceVector.push_back (lBP);
00292     } else {
00293         lBidPriceVector.push_back (lMinBP);
00294     }
00295 }
00296
00297 // Updating the bid price values
00300 ioLegCabin.updatePreviousBidPrice();
00301 ioLegCabin.setCurrentBidPrice (lBidPriceVector.back());
00302
00303 // Compute and display the bid price variation after optimisation
00304 const stdair::BidPrice_T lPreviousBP = ioLegCabin.getPreviousBidPrice();
00305 stdair::BidPrice_T lNewBP = ioLegCabin.getCurrentBidPrice();
00306 // Check

```

```

00307     assert (lPreviousBP != 0);
00308     stdair::BidPrice_T lBidPriceDelta = lNewBP - lPreviousBP;
00309
00310     double lBidPriceVariation = 100*lBidPriceDelta/lPreviousBP;
00311
00312     STDAIR_LOG_DEBUG ("Bid price: previous value " << lPreviousBP
00313                      << ", new value " << lNewBP
00314                      << ", variation " << lBidPriceVariation << " %"
00315                      << ", BPV size " << lBidPriceVector.size());
00316 }
00317
00318 }

```

43.104 rmol/bom/MCOptimiser.hpp File Reference

```

#include <rmol/RMOL_Types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>

```

Classes

- class [RMOL::MCOptimiser](#)

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.105 MCOptimiser.hpp

```

00001 #ifndef __RMOL_BOM_MCUTILS_HPP
00002 #define __RMOL_BOM_MCUTILS_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // RMOL
00008 #include <rmol/RMOL_Types.hpp>
00009 #include <stdair/stdair_maths_types.hpp>
00010 #include <stdair/stdair_rm_types.hpp>
00011
00012 // Forward declarations.
00013 namespace stdair {
00014     class LegCabin;
00015 }
00016
00017 namespace RMOL {
00019     class MCOptimiser {
00020     public:
00021
00022         static void optimalOptimisationByMCIntegration (stdair::LegCabin&);
00031

```

```

00035     static stdair::GeneratedDemandVector_T
00036     generateDemandVector (const stdair::MeanValue_T&,
00037                           const stdair::StdDevValue_T&, const unsigned int&);
00038
00039     static void optimisationByMCIntegration (stdair::LegCabin&);
00040
00041 };
00042 }
00043 #endif // __RMOL_BOM_MCUUTILS_HPP

```

43.106 rmol/bom/old/DemandGeneratorList.cpp File Reference

```
#include <rmol/bom/DemandGeneratorList.hpp>
```

Namespaces

- namespace [RMOL](#)

43.107 DemandGeneratorList.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // RMOL
00005 #include <rmol/bom/DemandGeneratorList.hpp>
00006
00007 namespace RMOL {
00008
00009 // //////////////////////////////////////
00010 DemandGeneratorList::DemandGeneratorList () {
00011     const DistributionParameterList_T aDistributionParameterList;
00012     init (aDistributionParameterList);
00013 }
00014
00015 // //////////////////////////////////////
00016 DemandGeneratorList::
00017 DemandGeneratorList (const DemandGeneratorList& iDemandGeneratorList) {
00018     // TODO: copy the distribution parameters of the input generator list
00019     const DistributionParameterList_T aDistributionParameterList;
00020     init (aDistributionParameterList);
00021 }
00022
00023 // //////////////////////////////////////
00024 DemandGeneratorList::
00025 DemandGeneratorList (const DistributionParameterList_T& iDistributionParameterL
00026 ist) {
00027     init (iDistributionParameterList);
00028 }
00029
00030 // //////////////////////////////////////
00031 DemandGeneratorList::~DemandGeneratorList () {
00032 }
00033
00034 // //////////////////////////////////////
00035 void DemandGeneratorList::
00036 init (const DistributionParameterList_T& iDistributionParameterList) {

```

```

00036
00037     DistributionParameterList_T::const_iterator itParams =
00038         iDistributionParameterList.begin();
00039     for ( ; itParams != iDistributionParameterList.end(); itParams++) {
00040         const FldDistributionParameters& aParams = *itParams;
00041
00042         const Gaussian gaussianGenerator (aParams);
00043
00044         _demandGeneratorList.push_back (gaussianGenerator);
00045     }
00046 }
00047
00048 // //////////////////////////////////////
00049 void DemandGeneratorList::
00050 generateVariateList (VariateList_T& ioVariateList) const {
00051
00052     // Iterate on the (number of) classes/buckets, n
00053     DemandGeneratorList_T::const_iterator itGenerator =
00054         _demandGeneratorList.begin();
00055     for ( ; itGenerator != _demandGeneratorList.end(); itGenerator++) {
00056         const Gaussian& gaussianGenerator = *itGenerator;
00057
00058         // Generate a random variate following the Gaussian distribution
00059         const double generatedVariate = gaussianGenerator.generateVariate ();
00060         ioVariateList.push_back (generatedVariate);
00061     }
00062 }
00063
00064 }

```

43.108 rmol/bom/old/DemandGeneratorList.hpp File Reference

```

#include <list>

#include <rmol/bom/VariateList.hpp>

#include <rmol/bom/DistributionParameterList.hpp>

#include <rmol/bom/Gaussian.hpp>

```

Classes

- class [RMOL::DemandGeneratorList](#)

Namespaces

- namespace [RMOL](#)

43.109 DemandGeneratorList.hpp

```

00001 #ifndef __RMOL_DEMANDGENERATORLIST_HPP
00002 #define __RMOL_DEMANDGENERATORLIST_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section

```

```

00006 // //////////////////////////////////////
00007 // STL
00008 #include <list>
00009 // RMOL
00010 #include <rmol/bom/VariateList.hpp>
00011 #include <rmol/bom/DistributionParameterList.hpp>
00012 #include <rmol/bom/Gaussian.hpp>
00013
00014 namespace RMOL {
00015
00016     class DemandGeneratorList {
00017     protected:
00020         typedef std::list<Gaussian> DemandGeneratorList_T;
00021
00022     public:
00024         DemandGeneratorList ();
00025         DemandGeneratorList (const DemandGeneratorList&);
00027         DemandGeneratorList (const DistributionParameterList_T&);
00028
00030         virtual ~DemandGeneratorList ();
00031
00033         void generateVariateList (VariateList_T&) const;
00034
00035     private:
00036         DemandGeneratorList_T _demandGeneratorList;
00037
00039         void init (const DistributionParameterList_T&);
00040
00041     };
00042 }
00043 #endif // __RMOL_DEMANDGENERATORLIST_HPP

```

43.110 rmol/bom/Utilities.cpp File Reference

```

#include <cassert>
#include <string>
#include <numeric>
#include <algorithm>
#include <cmath>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/basic/BasConst_General.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/GuillotineBlockHelper.hpp>

```

Namespaces

- namespace [RMOL](#)

43.111 Utilities.cpp

```

00001
00002 // //////////////////////////////////////
00003 // Import section
00004 // //////////////////////////////////////
00005 // STL
00006 #include <cassert>
00007 #include <string>
00008 #include <numeric>
00009 #include <algorithm>
00010 #include <cmath>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/BomManager.hpp>
00014 #include <stdair/bom/SegmentCabin.hpp>
00015 #include <stdair/service/Logger.hpp>
00016 // RMOL
00017 #include <rmol/basic/BasConst_General.hpp>
00018 #include <rmol/bom/Utilities.hpp>
00019 #include <rmol/bom/GuillotineBlockHelper.hpp>
00020
00021 namespace RMOL {
00022 // //////////////////////////////////////
00023 void Utilities::
00024 computeDistributionParameters (const UnconstrainedDemandVector_T& iVector,
00025                               double& ioMean, double& ioStdDev) {
00026     ioMean = 0.0; ioStdDev = 0.0;
00027     unsigned int lNbOfSamples = iVector.size();
00028     assert (lNbOfSamples > 1);
00029
00030     // Compute the mean
00031     for (UnconstrainedDemandVector_T::const_iterator itSample = iVector.begin();
00032          itSample != iVector.end(); ++itSample) {
00033         //STDAIR_LOG_NOTIFICATION (*itSample);
00034         ioMean += *itSample;
00035     }
00036     ioMean /= lNbOfSamples;
00037
00038     // Compute the standard deviation
00039     for (UnconstrainedDemandVector_T::const_iterator itSample = iVector.begin();
00040          itSample != iVector.end(); ++itSample) {
00041         const double& lSample = *itSample;
00042         ioStdDev += (lSample - ioMean) * (lSample - ioMean);
00043     }
00044     ioStdDev /= (lNbOfSamples - 1);
00045     ioStdDev = sqrt (ioStdDev);
00046
00047     // Sanity check
00048     if (ioStdDev == 0) {
00049         ioStdDev = 0.1;
00050     }
00051 }
00052
00053 // //////////////////////////////////////
00054 stdair::DCPList_T Utilities::

```



```

00055 buildRemainingDCPList (const stdair::DTD_T& iDTD) {
00056     stdair::DCPList_T oDCPList;
00057
00058     const stdair::DCPList_T lWholeDCPList = stdair::DEFAULT_DCP_LIST;
00059     stdair::DCPList_T::const_iterator itDCP = lWholeDCPList.begin();
00060     while (itDCP != lWholeDCPList.end()) {
00061         const stdair::DCP_T& lDCP = *itDCP;
00062         if (iDTD >= lDCP) {
00063             break;
00064         }
00065         ++itDCP;
00066     }
00067     assert (itDCP != lWholeDCPList.end());
00068
00069     oDCPList.push_back (iDTD);
00070     ++itDCP;
00071     for (; itDCP != lWholeDCPList.end(); ++itDCP) {
00072         oDCPList.push_back (*itDCP);
00073     }
00074
00075     return oDCPList;
00076 }
00077
00078 // //////////////////////////////////////
00079 stdair::DCPList_T Utilities::
00080 buildRemainingDCPList2 (const stdair::DTD_T& iDTD) {
00081     stdair::DCPList_T oDCPList;
00082
00083     const stdair::DCPList_T lWholeDCPList = RMOL::DEFAULT_DCP_LIST;
00084     stdair::DCPList_T::const_iterator itDCP = lWholeDCPList.begin();
00085     while (itDCP != lWholeDCPList.end()) {
00086         const stdair::DCP_T& lDCP = *itDCP;
00087         if (iDTD >= lDCP) {
00088             break;
00089         }
00090         ++itDCP;
00091     }
00092     assert (itDCP != lWholeDCPList.end());
00093
00094     oDCPList.push_back (iDTD);
00095     ++itDCP;
00096     for (; itDCP != lWholeDCPList.end(); ++itDCP) {
00097         oDCPList.push_back (*itDCP);
00098     }
00099
00100     return oDCPList;
00101 }
00102
00103 // //////////////////////////////////////
00104 stdair::NbOfSegments_T Utilities::
00105 getNbOfDepartedSimilarSegments (const stdair::SegmentCabin& iSegmentCabin,
00106                                 const stdair::Date_T& iEventDate) {
00107     stdair::DTD_T lDTD = 0;
00108     // Retrieve the guillotine block.
00109     const stdair::GuillotineBlock& lGuillotineBlock =
00110         iSegmentCabin.getGuillotineBlock();
00111     return GuillotineBlockHelper::
00112         getNbOfSegmentAlreadyPassedThisDTD (lGuillotineBlock, lDTD, iEventDate);
00113 }
00114
00115 }

```

43.112 rmol/bom/Utilities.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Classes

- class [RMOL::Utilities](#)

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.113 Utilities.hpp

```
00001 #ifndef __RMOL_BOM_UTILITIES_HPP
00002 #define __RMOL_BOM_UTILITIES_HPP
00003 // //////////////////////////////////////
00004 // Import section
00005 // //////////////////////////////////////
00006 // StdAir
00007 #include <stdair/stdair_inventory_types.hpp>
00008 // RMOL
00009 #include <rmol/RMOL_Types.hpp>
00010
00011 // Forward declarations
00012 namespace stdair {
00013     class SegmentCabin;
00014 }
00015
00016 namespace RMOL {
00017
00018     class Utilities {
00019     public:
00020         static void computeDistributionParameters (const UnconstrainedDemandVector_T&
00021 , double&, double&);
00022
00023         static stdair::DCPList_T buildRemainingDCPList (const stdair::DTD_T&);
00024         static stdair::DCPList_T buildRemainingDCPList2 (const stdair::DTD_T&);
00025
00026         static stdair::NbOfSegments_T getNbOfDepartedSimilarSegments (const stdair::S
00027 egmentCabin&, const stdair::Date_T&);
00028
00029     };
00030
00031 };
00032
00033 #endif // __RMOL_BOM_UTILITIES_HPP
```

43.114 rmol/command/Detruncator.cpp File Reference

```
#include <cassert>
```

```

#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/GuillotineBlock.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/GuillotineBlockHelper.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/bom/EMDetruncator.hpp>
#include <rmol/command/Detruncator.hpp>

```

Namespaces

- namespace [RMOL](#)

43.115 Detruncator.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/basic/BasConst_Inventory.hpp>
00008 #include <stdair/bom/GuillotineBlock.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/FlightDate.hpp>
00011 #include <stdair/bom/SegmentDate.hpp>
00012 #include <stdair/bom/SegmentCabin.hpp>
00013 #include <stdair/bom/BookingClass.hpp>
00014 #include <stdair/service/Logger.hpp>
00015 // RMOL
00016 #include <rmol/bom/GuillotineBlockHelper.hpp>
00017 #include <rmol/bom/HistoricalBookingHolder.hpp>
00018 #include <rmol/bom/HistoricalBooking.hpp>
00019 #include <rmol/bom/EMDetruncator.hpp>
00020 #include <rmol/command/Detruncator.hpp>
00021
00022 namespace RMOL {
00023 // //////////////////////////////////////
00024 void Detruncator::
00025     unconstrainUsingAdditivePickUp (const stdair::SegmentCabin& iSegmentCabin,
00026                                     BookingClassUnconstrainedDemandVectorMap_T& ioBkgClassUncDemMap,
00027                                     UnconstrainedDemandVector_T& ioQEquivalentDemandVector,
00028                                     const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,

```

```

00029         const stdair::Date_T& iCurrentDate) {
00030
00031     // Retrieve the guillotine block.
00032     const stdair::GuillotineBlock& lGuillotineBlock =
00033         iSegmentCabin.getGuillotineBlock();
00034
00035     // Build the historical booking holders for the product-oriented bookings
00036     // of the casses and the Q-equivalent (price-oriented) bookings of the cabin
00037     const stdair::NbOfSegments_T lNbOfUsableSegments = GuillotineBlockHelper::
00038         getNbOfSegmentAlreadyPassedThisDTD (lGuillotineBlock, iDCPEnd,
00039         iCurrentDate);
00040
00041     // Parse the booking class list and unconstrain historical bookings.
00042     for (BookingClassUnconstrainedDemandVectorMap_T::iterator itBCUDV =
00043         ioBkgClassUncDemMap.begin(); itBCUDV != ioBkgClassUncDemMap.end();
00044         ++itBCUDV) {
00045         stdair::BookingClass* lBC_ptr = itBCUDV->first;
00046         assert (lBC_ptr != NULL);
00047         const stdair::MapKey_T& lBCKey = lBC_ptr->describeKey();
00048         const stdair::BlockIndex_T& lBlockIdx =
00049             lGuillotineBlock.getBlockIndex (lBCKey);
00050         UnconstrainedDemandVector_T& lUncDemVector = itBCUDV->second;
00051
00052         STDAIR_LOG_DEBUG ("Unconstrain product-oriented bookings for " << lBCKey);
00053         // STDAIR_LOG_NOTIFICATION (lBCKey << ";" << iDCPBegin
00054         // << ";" << iDCPEnd);
00055         unconstrainUsingAdditivePickUp (lGuillotineBlock, lUncDemVector,
00056             iDCPBegin, iDCPEnd,
00057             lNbOfUsableSegments, lBlockIdx);
00058     }
00059
00060     // Unconstrain the Q-equivalent bookings.
00061     // Retrieve the block index of the segment-cabin.
00062     std::ostringstream lSCMapKey;
00063     lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
00064         << iSegmentCabin.describeKey();
00065     const stdair::BlockIndex_T& lCabinIdx =
00066         lGuillotineBlock.getBlockIndex (lSCMapKey.str());
00067
00068     STDAIR_LOG_DEBUG ("Unconstrain price-oriented bookings");
00069     //STDAIR_LOG_NOTIFICATION (iDCPBegin << ";" << iDCPEnd);
00070     unconstrainUsingAdditivePickUp (lGuillotineBlock, ioQEquivalentDemandVector,
00071         iDCPBegin, iDCPEnd, lNbOfUsableSegments,
00072         lCabinIdx, iSegmentCabin, iCurrentDate);
00073 }
00074
00075 // //////////////////////////////////////
00076 void Detruncator::unconstrainUsingAdditivePickUp
00077 (const stdair::GuillotineBlock& iGuillotineBlock,
00078     UnconstrainedDemandVector_T& ioUncDemVector,
00079     const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,
00080     const stdair::NbOfSegments_T& iNbOfUsableSegments,
00081     const stdair::BlockIndex_T& iBlockIdx) {
00082     // TODO:
00083     stdair::NbOfSegments_T lSegBegin = 0;
00084     if (iNbOfUsableSegments > 52) lSegBegin = iNbOfUsableSegments - 52;
00085     // Retrieve the gross daily booking and availability snapshots.
00086     stdair::ConstSegmentCabinDTDRangeSnapshotView_T lBookingView =
00087         iGuillotineBlock.getConstSegmentCabinDTDRangeProductAndPriceOrientedBooking
00088         SnapshotView (lSegBegin, iNbOfUsableSegments - 1, iDCPEnd, iDCPBegin);
00089     stdair::ConstSegmentCabinDTDRangeSnapshotView_T lAvlView =
00090         iGuillotineBlock.getConstSegmentCabinDTDRangeAvailabilitySnapshotView (lSeg

```

```

Begin, iNbOfUsableSegments - 1, iDCPEnd, iDCPBegin);
00090
00091 // Browse the list of segments and build the historical booking holder.
00092 const stdair::ValueTypeIndexMap_T& lVTIdxMap =
00093     iGuillotineBlock.getValueTypeIndexMap();
00094 const unsigned int lNbOfValueTypes = lVTIdxMap.size();
00095 HistoricalBookingHolder lHBHolder;
00096 for (short i = 0; i < iNbOfUsableSegments - lSegBegin; ++i) {
00097     stdair::Flag_T lCensorshipFlag = false;
00098     stdair::NbOfBookings_T lNbOfHistoricalBkgs = 0.0;
00099     const short lNbOfDTDs = iDCPBegin - iDCPEnd + 1;
00100
00101     // Parse the DTDs during the period
00102     for (short j = 0; j < lNbOfDTDs; ++j) {
00103         // Check if the data has been censored during this day.
00104         // STDAIR_LOG_DEBUG ("i: " << i << ", NbOfValues: " << lNbOfValueTypes
00105         //                     << ", BlockIdx: " << iBlockIdx << ", j: " << j);
00106         if (lCensorshipFlag == false) {
00107             if (lAvlView[i * lNbOfValueTypes + iBlockIdx][j] < 1.0) {
00108                 lCensorshipFlag = true;
00109             }
00110         }
00111
00112         // Get the bookings of the day.
00113         //STDAIR_LOG_DEBUG ("Bookings of the day: " << lBookingView[i * lNbOfValueT
00114         ypes + iBlockIdx][j]);
00115         lNbOfHistoricalBkgs += lBookingView[i * lNbOfValueTypes + iBlockIdx][j];
00116     }
00117     HistoricalBooking lHistoricalBkg (lNbOfHistoricalBkgs, lCensorshipFlag);
00118     lHBHolder.addHistoricalBooking (lHistoricalBkg);
00119
00120     // DEBUG
00121     STDAIR_LOG_DEBUG ("Historical bkgs: " << lNbOfHistoricalBkgs
00122                     << ", censored: " << lCensorshipFlag);
00123     // STDAIR_LOG_NOTIFICATION (lNbOfHistoricalBkgs
00124     //                          << "; " << lCensorshipFlag);
00125 }
00126
00127 // DEBUG
00128 STDAIR_LOG_DEBUG ("Unconstrain by EM");
00129
00130 // Unconstrain the booking figures
00131 EMDetruncator::unconstrainUsingEMMethod (lHBHolder);
00132
00133 // Add the unconstrained demand of the period to the unconstrained demand
00134 // vector.
00135 short idx = 0;
00136 for (UnconstrainedDemandVector_T::iterator itUD = ioUncDemVector.begin();
00137     itUD != ioUncDemVector.end(); ++itUD, ++idx) {
00138     *itUD += lHBHolder.getUnconstrainedDemand (idx);
00139     //STDAIR_LOG_NOTIFICATION (lHBHolder.getUnconstrainedDemand (idx));
00140 }
00141 }
00142
00143 // //////////////////////////////////////
00144 void Detruncator::unconstrainUsingAdditivePickUp
00145 (const stdair::GuillotineBlock& iGuillotineBlock,
00146  UnconstrainedDemandVector_T& ioUncDemVector,
00147  const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,
00148  const stdair::NbOfSegments_T& iNbOfUsableSegments,
00149  const stdair::BlockIndex_T& iBlockIdx,

```

```

00150     const stdair::SegmentCabin& iSegmentCabin,
00151     const stdair::Date_T& iCurrentDate) {
00152         // TODO
00153         stdair::NbOfSegments_T lSegBegin = 0;
00154         if (iNbOfUsableSegments > 52) lSegBegin = iNbOfUsableSegments - 52;
00155         // Retrieve the gross daily booking and availability snapshots.
00156         stdair::ConstSegmentCabinDTDRangeSnapshotView_T lBookingView =
00157             iGuillotineBlock.getConstSegmentCabinDTDRangeProductAndPriceOrientedBooking
00158             SnapshotView (lSegBegin, iNbOfUsableSegments - 1, iDCPEnd, iDCPBegin);
00159         stdair::ConstSegmentCabinDTDRangeSnapshotView_T lAvlView =
00160             iGuillotineBlock.getConstSegmentCabinDTDRangeAvailabilitySnapshotView (lSeg
00161             Begin, iNbOfUsableSegments - 1, iDCPEnd, iDCPBegin);
00162
00163         // Browse the list of segments and build the historical booking holder.
00164         const stdair::ValueTypeIndexMap_T& lVTIdxMap =
00165             iGuillotineBlock.getValueTypeIndexMap();
00166         const unsigned int lNbOfValueTypes = lVTIdxMap.size();
00167         HistoricalBookingHolder lHBHolder;
00168         for (short i = 0; i < iNbOfUsableSegments - lSegBegin; ++i) {
00169             stdair::Flag_T lCensorshipFlag = false;
00170             stdair::NbOfBookings_T lNbOfHistoricalBkgs = 0.0;
00171             const short lNbOfDTDs = iDCPBegin - iDCPEnd + 1;
00172
00173             // Parse the DTDs during the period
00174             for (short j = 0; j < lNbOfDTDs; ++j) {
00175                 // Check if the data has been censored during this day.
00176                 // STDAIR_LOG_DEBUG ("i: " << i << ", NbOfValues: " << lNbOfValueTypes
00177                 // << ", BlockIdx: " << iBlockIdx << ", j: " << j);
00178                 if (lCensorshipFlag == false) {
00179                     if (lAvlView[i * lNbOfValueTypes + iBlockIdx][j] < 1.0) {
00180                         lCensorshipFlag = true;
00181                     }
00182                 }
00183
00184                 // Get the bookings of the day.
00185                 //STDAIR_LOG_DEBUG ("Bookings of the day: " << lBookingView[i * lNbOfValueT
00186                 ypes + iBlockIdx][j]);
00187                 lNbOfHistoricalBkgs += lBookingView[i * lNbOfValueTypes + iBlockIdx][j];
00188             }
00189
00190             HistoricalBooking lHistoricalBkg (lNbOfHistoricalBkgs, lCensorshipFlag);
00191             lHBHolder.addHistoricalBooking (lHistoricalBkg);
00192
00193             // DEBUG
00194             STDAIR_LOG_DEBUG ("Historical bkgs: " << lNbOfHistoricalBkgs
00195             << ", censored: " << lCensorshipFlag);
00196             // STDAIR_LOG_NOTIFICATION (lNbOfHistoricalBkgs
00197             // << "; " << lCensorshipFlag);
00198         }
00199
00200         // DEBUG
00201         STDAIR_LOG_DEBUG ("Unconstrain by EM");
00202
00203         // Unconstrain the booking figures
00204         EMDetruncator::unconstrainUsingEMMethod (lHBHolder);
00205
00206         // Add the unconstrained demand of the period to the unconstrained demand
00207         // vector.
00208         // LOG
00209         const stdair::SegmentDate& lSegmentDate = stdair::BomManager::
00210             getParent<stdair::SegmentDate, stdair::SegmentCabin> (iSegmentCabin);
00211         const stdair::FlightDate& lFlightDate = stdair::BomManager::

```

```

00209     getParent<stdair::FlightDate, stdair::SegmentDate> (lSegmentDate);
00210     const stdair::Date_T& lDepDate = lFlightDate.getDepartureDate();
00211     const boost::gregorian::date_duration lDD = lDepDate - iCurrentDate;
00212     const long lDTD = lDD.days();
00213     stdair::Date_T lRefDate (2012, boost::gregorian::Jan, 01);
00214
00215     short idx = 0;
00216     for (UnconstrainedDemandVector_T::iterator itUD = ioUncDemVector.begin();
00217          itUD != ioUncDemVector.end(); ++itUD, ++idx) {
00218         *itUD += lHBHolder.getUnconstrainedDemand (idx);
00219         if (lDepDate > lRefDate) {
00220             const stdair::DateOffset_T lDateOffset (7 *(52 - idx) + 420);
00221             const stdair::Date_T lHDate = lDepDate - lDateOffset;
00222             STDAIR_LOG_NOTIFICATION (boost::gregorian::to_iso_string(lDepDate)
00223                                     << ";" << lDTD << ";" << iDCPBegin << ";"
00224                                     << iDCPEnd << ";"
00225                                     << boost::gregorian::to_iso_string (lHDate)
00226                                     << ";" << lHBHolder.getUnconstrainedDemand (idx));
00227
00228             STDAIR_LOG_NOTIFICATION (boost::gregorian::to_iso_string(lDepDate)
00229                                     << ";" << lDTD << ";" << iDCPBegin << ";"
00230                                     << iDCPEnd << ";"
00231                                     << boost::gregorian::to_iso_string (lHDate)
00232                                     << ";" << lHBHolder.getHistoricalBooking (idx));
00233         }
00234     }
00235 }
00236
00237 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00238 void Detruncator::retrieveUnconstrainedDemandForFirstDCP
00239 (const stdair::SegmentCabin& iSegmentCabin,
00240  BookingClassUnconstrainedDemandVectorMap_T& ioBkgClassUncDemVectorMap,
00241  UnconstrainedDemandVector_T& ioQEquivalentDemandVector,
00242  const stdair::DCP_T& iFirstDCP, const stdair::NbOfSegments_T& iNbOfSegments,
00243  const stdair::NbOfSegments_T& iNbOfUsedSegments) {
00244
00245     // Retrieve the guillotine block.
00246     const stdair::GuillotineBlock& lGuillotineBlock =
00247         iSegmentCabin.getGuillotineBlock();
00248
00249     // Parse the booking class list and unconstrain historical bookings.
00250     for (BookingClassUnconstrainedDemandVectorMap_T::iterator itBCUDV =
00251          ioBkgClassUncDemVectorMap.begin();
00252          itBCUDV != ioBkgClassUncDemVectorMap.end(); ++itBCUDV) {
00253         stdair::BookingClass* lBC_ptr = itBCUDV->first;
00254         assert (lBC_ptr != NULL);
00255         const stdair::MapKey_T& lBCKey = lBC_ptr->describeKey();
00256         const stdair::BlockIndex_T& lBlockIdx =
00257             lGuillotineBlock.getBlockIndex (lBCKey);
00258         UnconstrainedDemandVector_T& lUncDemVector = itBCUDV->second;
00259
00260         STDAIR_LOG_DEBUG("Retrieve the unconstrained product-oriented demand for "
00261                         << lBCKey);
00262         retrieveUnconstrainedDemandForFirstDCP (lGuillotineBlock, lUncDemVector,
00263                                                 iFirstDCP, lBlockIdx,
00264                                                 iNbOfSegments, iNbOfUsedSegments);
00265     }
00266
00267     // Unconstrain the Q-equivalent bookings.
00268     // Retrieve the block index of the segment-cabin.
00269     std::ostream lSCMapKey;
00270     lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE

```

```

00271         << iSegmentCabin.describeKey();
00272     const stdair::BlockIndex_T& lCabinValueIdx =
00273         lGuillotineBlock.getBlockIndex (lSCMapKey.str());
00274
00275     STDAIR_LOG_DEBUG ("Retrieve the unconstrained price-oriented demand");
00276     retrieveUnconstrainedDemandForFirstDCP (lGuillotineBlock,
00277         ioQEquivalentDemandVector,iFirstDCP,
00278         lCabinValueIdx, iNbOfSegments,
00279         iNbOfUsedSegments);
00280 }
00281
00282 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00283 void Detruncator::retrieveUnconstrainedDemandForFirstDCP
00284 (const stdair::GuillotineBlock& iGuillotineBlock,
00285  UnconstrainedDemandVector_T& ioUnconstrainedDemandVector,
00286  const stdair::DCP_T& iFirstDCP, const stdair::BlockIndex_T& iValueIdx,
00287  const stdair::NbOfSegments_T& iNbOfSegments,
00288  const stdair::NbOfSegments_T& iNbOfUsedSegments) {
00289
00290     //TODO
00291     stdair::NbOfSegments_T lSegBegin = iNbOfSegments - iNbOfUsedSegments;
00292
00293     // Retrieve the snapshots of the corresponding booking value from the
00294     // first DTD (usually 365) till the given iFirstDCP.
00295     stdair::ConstSegmentCabinDTDRangeSnapshotView_T lRangeBookingView =
00296         iGuillotineBlock.getConstSegmentCabinDTDRangeProductAndPriceOrientedBooking
00297         SnapshotView (lSegBegin, iNbOfSegments -1, iFirstDCP, stdair::DEFAULT_MAX_DTD);
00298
00299     // Sum the bookings from the first day till the given iFirstDCP in order to
00300     // get the supposing unconstrained demand for this period.
00301     const stdair::ValueTypeIndexMap_T& lVTIdxMap =
00302         iGuillotineBlock.getValueTypeIndexMap();
00303     const unsigned int lNbOfValueTypes = lVTIdxMap.size();
00304     for (int itSegment = 0; itSegment < iNbOfSegments-lSegBegin; ++itSegment) {
00305         for (int i = iFirstDCP; i <= stdair::DEFAULT_MAX_DTD; ++i) {
00306             stdair::NbOfRequests_T& lUncDemand =
00307                 ioUnconstrainedDemandVector.at (itSegment);
00308             lUncDemand +=
00309                 lRangeBookingView[iValueIdx + itSegment*lNbOfValueTypes][i-iFirstDCP];
00310             // STDAIR_LOG_NOTIFICATION (ioUnconstrainedDemandVector.at (itSegment)
00311             // << " " << itSegment);
00312         }
00313     }
00314
00315 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00316 void Detruncator::unconstrainUsingMultiplicativePickUp
00317 (const stdair::SegmentCabin& iSegmentCabin,
00318  BookingClassUnconstrainedDemandVectorMap_T& ioBkgClassUncDemMap,
00319  UnconstrainedDemandVector_T& ioQEquivalentDemandVector,
00320  const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,
00321  const stdair::Date_T& iCurrentDate,
00322  const stdair::NbOfSegments_T& iNbOfDepartedSegments) {
00323
00324     // Retrieve the guillotine block.
00325     const stdair::GuillotineBlock& lGuillotineBlock =
00326         iSegmentCabin.getGuillotineBlock();
00327
00328     // Build the historical booking holders for the product-oriented bookings
00329     // of the casses and the Q-equivalent (price-oriented) bookings of the cabin
00330     const stdair::NbOfSegments_T lNbOfUsableSegments = GuillotineBlockHelper::
00331         getNbOfSegmentAlreadyPassedThisDTD (lGuillotineBlock, iDCPEnd,

```



```

00332                                     iCurrentDate);
00333
00334 // Parse the booking class list and unconstrain historical bookings.
00335 for (BookingClassUnconstrainedDemandVectorMap_T::iterator itBCUDV =
00336      ioBkgClassUncDemMap.begin(); itBCUDV != ioBkgClassUncDemMap.end();
00337      ++itBCUDV) {
00338     stdair::BookingClass* lBC_ptr = itBCUDV->first;
00339     assert (lBC_ptr != NULL);
00340     const stdair::MapKey_T& lBCKey = lBC_ptr->describeKey();
00341     const stdair::BlockIndex_T& lBlockIdx =
00342         lGuillotineBlock.getBlockIndex (lBCKey);
00343     UnconstrainedDemandVector_T& lUncDemVector = itBCUDV->second;
00344
00345     STDAIR_LOG_DEBUG ("Unconstrain product-oriented bookings for " << lBCKey);
00346     unconstrainUsingMultiplicativePickUp (lGuillotineBlock, lUncDemVector,
00347         iDCPBegin, iDCPEnd,
00348         lNbOfUsableSegments, lBlockIdx,
00349         iNbOfDepartedSegments);
00350 }
00351
00352 // Unconstrain the Q-equivalent bookings.
00353 // Retrieve the block index of the segment-cabin.
00354 std::ostream lSCMapKey;
00355 lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
00356     << iSegmentCabin.describeKey();
00357 const stdair::BlockIndex_T& lCabinIdx =
00358     lGuillotineBlock.getBlockIndex (lSCMapKey.str());
00359
00360 STDAIR_LOG_DEBUG ("Unconstrain price-oriented bookings");
00361 unconstrainUsingMultiplicativePickUp (lGuillotineBlock,
00362     ioQEquivalentDemandVector,
00363     iDCPBegin, iDCPEnd,
00364     lNbOfUsableSegments, lCabinIdx,
00365     iNbOfDepartedSegments,
00366     iSegmentCabin, iCurrentDate);
00367 }
00368
00369 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00370 void Detruncator::unconstrainUsingMultiplicativePickUp
00371 (const stdair::GuillotineBlock& iGuillotineBlock,
00372  UnconstrainedDemandVector_T& ioUncDemVector,
00373  const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,
00374  const stdair::NbOfSegments_T& iNbOfUsableSegments,
00375  const stdair::BlockIndex_T& iBlockIdx,
00376  const stdair::NbOfSegments_T& iNbOfDepartedSegments) {
00377     // TODO:
00378     stdair::NbOfSegments_T lSegBegin = 0;
00379     if (iNbOfDepartedSegments > 52) {
00380         lSegBegin = iNbOfDepartedSegments - 52;
00381     }
00382
00383     // Retrieve the gross daily booking and availability snapshots.
00384     stdair::ConstSegmentCabinDTRangeSnapshotView_T lBookingView =
00385         iGuillotineBlock.getConstSegmentCabinDTRangeProductAndPriceOrientedBooking
00386         SnapshotView (lSegBegin, iNbOfUsableSegments - 1, iDCPEnd, iDCPBegin);
00387     stdair::ConstSegmentCabinDTRangeSnapshotView_T lAvlView =
00388         iGuillotineBlock.getConstSegmentCabinDTRangeAvailabilitySnapshotView (lSeg
00389         Begin, iNbOfUsableSegments - 1, iDCPEnd, iDCPBegin);
00388
00389     // Browse the list of segments and build the historical booking holder.
00390     const stdair::ValueTypeIndexMap_T& lVTIdxMap =
00391         iGuillotineBlock.getValueTypeIndexMap();

```

```

00392     const unsigned int lNbOfValueTypes = lVTIdxMap.size();
00393     HistoricalBookingHolder lHBHolder;
00394     std::vector<short> lDataIndexList;
00395     for (short i = 0; i < iNbOfUsableSegments-lSegBegin; ++i) {
00396         stdair::Flag_T lCensorshipFlag = false;
00397         stdair::NbOfBookings_T lNbOfHistoricalBkgs = 0.0;
00398         const short lNbOfDTDs = iDCPBegin - iDCPEnd + 1;
00399
00400         // Parse the DTDs during the period
00401         for (short j = 0; j < lNbOfDTDs; ++j) {
00402             // Check if the data has been censored during this day.
00403             // STDAIR_LOG_DEBUG ("i: " << i << ", NbOfValues: " << lNbOfValueTypes
00404             //                     << ", BlockIdx: " << iBlockIdx << ", j: " << j);
00405             if (lCensorshipFlag == false) {
00406                 if (lAvlView[i*lNbOfValueTypes + iBlockIdx][j] < 1.0) {
00407                     lCensorshipFlag = true;
00408                 }
00409             }
00410
00411             // Get the bookings of the day.
00412             //STDAIR_LOG_DEBUG ("Bookings of the day: " << lBookingView[i*lNbOfValueT
ypes + iBlockIdx][j]);
00413             lNbOfHistoricalBkgs += lBookingView[i*lNbOfValueTypes + iBlockIdx][j];
00414         }
00415
00416         // If there is no booking till now for this class and for this segment,
00417         // there will be no unconstraining process.
00418         stdair::NbOfRequests_T& lUncDemand = ioUncDemVector.at (i);
00419         if (lUncDemand < 1.0) {
00420             lUncDemand += lNbOfHistoricalBkgs;
00421         } else {
00422             double lBkgDemandFactor = lNbOfHistoricalBkgs / lUncDemand;
00423             HistoricalBooking lHistoricalBkg (lBkgDemandFactor, lCensorshipFlag);
00424             lHBHolder.addHistoricalBooking (lHistoricalBkg);
00425             lDataIndexList.push_back (i);
00426         }
00427
00428         // DEBUG
00429         STDAIR_LOG_DEBUG ("Historical bkgs: " << lNbOfHistoricalBkgs
00430                           << ", censored: " << lCensorshipFlag);
00431     }
00432
00433     // DEBUG
00434     STDAIR_LOG_DEBUG ("Unconstrain by multiplicative pick-up using EM");
00435
00436     // Unconstrain the booking figures
00437     unconstrainUsingMultiplicativePickUp (lHBHolder);
00438
00439     // Update the unconstrained demand vector.
00440     short i = 0;
00441     for (std::vector<short>::iterator itIdx = lDataIndexList.begin();
00442          itIdx != lDataIndexList.end(); ++itIdx, ++i) {
00443         short lIdx = *itIdx;
00444         stdair::NbOfRequests_T& lPastDemand = ioUncDemVector.at (lIdx);
00445         const stdair::NbOfRequests_T& lUncDemandFactorOfThisPeriod =
00446             lHBHolder.getUnconstrainedDemand (i);
00447         lPastDemand *= (1+lUncDemandFactorOfThisPeriod);
00448     }
00449 }
00450
00451 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00452 void Detruncator::unconstrainUsingMultiplicativePickUp

```

```

00453 (const stdair::GuillotineBlock& iGuillotineBlock,
00454     UnconstrainedDemandVector_T& ioUncDemVector,
00455     const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,
00456     const stdair::NbOfSegments_T& iNbOfUsableSegments,
00457     const stdair::BlockIndex_T& iBlockIdx,
00458     const stdair::NbOfSegments_T& iNbOfDepartedSegments,
00459     const stdair::SegmentCabin& iSegmentCabin,
00460     const stdair::Date_T& iCurrentDate) {
00461     // TODO:
00462     stdair::NbOfSegments_T lSegBegin = 0;
00463     if (iNbOfDepartedSegments > 52) {
00464         lSegBegin = iNbOfDepartedSegments - 52;
00465     }
00466
00467     // Retrieve the gross daily booking and availability snapshots.
00468     stdair::ConstSegmentCabinDTDRangeSnapshotView_T lBookingView =
00469         iGuillotineBlock.getConstSegmentCabinDTDRangeProductAndPriceOrientedBooking
00470         SnapshotView (lSegBegin, iNbOfUsableSegments -1, iDCPEnd, iDCPBegin);
00471     stdair::ConstSegmentCabinDTDRangeSnapshotView_T lAvlView =
00472         iGuillotineBlock.getConstSegmentCabinDTDRangeAvailabilitySnapshotView (lSeg
00473         Begin, iNbOfUsableSegments -1, iDCPEnd, iDCPBegin);
00474
00475     // Browse the list of segments and build the historical booking holder.
00476     const stdair::ValueTypeIndexMap_T lVTIdxMap =
00477         iGuillotineBlock.getValueTypeIndexMap();
00478     const unsigned int lNbOfValueTypes = lVTIdxMap.size();
00479     HistoricalBookingHolder lHBHolder;
00480     std::vector<short> lDataIndexList;
00481     for (short i = 0; i < iNbOfUsableSegments-lSegBegin; ++i) {
00482         stdair::Flag_T lCensorshipFlag = false;
00483         stdair::NbOfBookings_T lNbOfHistoricalBkgs = 0.0;
00484         const short lNbOfDTDs = iDCPBegin - iDCPEnd + 1;
00485
00486         // Parse the DTDs during the period
00487         for (short j = 0; j < lNbOfDTDs; ++j) {
00488             // Check if the data has been censored during this day.
00489             // STDAIR_LOG_DEBUG ("i: " << i << ", NbOfValues: " << lNbOfValueTypes
00490             // << ", BlockIdx: " << iBlockIdx << ", j: " << j);
00491             if (lCensorshipFlag == false) {
00492                 if (lAvlView[i*lNbOfValueTypes + iBlockIdx][j] < 1.0) {
00493                     lCensorshipFlag = true;
00494                 }
00495             }
00496
00497             // Get the bookings of the day.
00498             //STDAIR_LOG_DEBUG ("Bookings of the day: " << lBookingView[i*lNbOfValueT
00499             ypes + iBlockIdx][j]);
00500             lNbOfHistoricalBkgs += lBookingView[i*lNbOfValueTypes + iBlockIdx][j];
00501         }
00502
00503         // If there is no booking till now for this class and for this segment,
00504         // there will be no unconstraining process.
00505         stdair::NbOfRequests_T lUncDemand = ioUncDemVector.at (i);
00506         if (lUncDemand < 1.0) {
00507             lUncDemand += lNbOfHistoricalBkgs;
00508         } else {
00509             double lBkgDemandFactor = lNbOfHistoricalBkgs / lUncDemand;
00510             HistoricalBooking lHistoricalBkg (lBkgDemandFactor, lCensorshipFlag);
00511             lHBHolder.addHistoricalBooking (lHistoricalBkg);
00512             lDataIndexList.push_back (i);
00513         }
00514     }
00515 }

```

```

00512         // DEBUG
00513         STDAIR_LOG_DEBUG ("Historical bkgs: " << lNbOfHistoricalBkgs
00514                         << ", censored: " << lCensorshipFlag);
00515     }
00516
00517     // DEBUG
00518     STDAIR_LOG_DEBUG ("Unconstrain by multiplicative pick-up");
00519
00520     // Unconstrain the booking figures
00521     unconstrainUsingMultiplicativePickUp (lHBHolder);
00522
00523     // Update the unconstrained demand vector.
00524     // LOG
00525     const stdair::SegmentDate& lSegmentDate = stdair::BomManager::
00526         getParent<stdair::SegmentDate, stdair::SegmentCabin> (iSegmentCabin);
00527     const stdair::FlightDate& lFlightDate = stdair::BomManager::
00528         getParent<stdair::FlightDate, stdair::SegmentDate> (lSegmentDate);
00529     const stdair::Date_T& lDepDate = lFlightDate.getDepartureDate();
00530     const boost::gregorian::date_duration lDD = lDepDate - iCurrentDate;
00531     const long lDTD = lDD.days();
00532     stdair::Date_T lRefDate (2012, boost::gregorian::Jan, 01);
00533
00534     short i = 0;
00535     for (std::vector<short>::iterator itIdx = lDataIndexList.begin();
00536          itIdx != lDataIndexList.end(); ++itIdx, ++i) {
00537         short lIdx = *itIdx;
00538         stdair::NbOfRequests_T& lPastDemand = ioUncDemVector.at (lIdx);
00539         const stdair::NbOfRequests_T& lUncDemandFactorOfThisPeriod =
00540             lHBHolder.getUnconstrainedDemand (i);
00541         const double lUncDemThisPeriod =
00542             lPastDemand * lUncDemandFactorOfThisPeriod;
00543         lPastDemand *= (1+lUncDemandFactorOfThisPeriod);
00544         if (lDepDate > lRefDate) {
00545             const stdair::DateOffset_T lDateOffset (7 *(53 - lIdx) + 420);
00546             const stdair::Date_T lHDate = lDepDate - lDateOffset;
00547             STDAIR_LOG_NOTIFICATION (boost::gregorian::to_iso_string(lDepDate)
00548                                     << ";" << lDTD << ";" << iDCPBegin << ";"
00549                                     << iDCPEnd << ";"
00550                                     << boost::gregorian::to_iso_string (lHDate)
00551                                     << ";" << lUncDemThisPeriod);
00552         }
00553     }
00554 }
00555
00556 // //////////////////////////////////////
00557 void Detruncator::
00558 unconstrainUsingMultiplicativePickUp (HistoricalBookingHolder& ioHBHolder) {
00559     // We use two loops in this algorithm. The first one is for calculating the
00560     // average of unconstrained data. The second one is for calculating the
00561     // average of unconstrained data and the constrained data which are higher
00562     // than the first average.
00563     short lNbOfUsedData = ioHBHolder.getNbOfUncensoredData();
00564     if (lNbOfUsedData > 0) {
00565         double lSumOfValues = 0.0;
00566         const short lNbOfData = ioHBHolder.getNbOfFlights();
00567
00568         // First loop
00569         for (short i = 0; i < lNbOfData; ++i) {
00570             if (ioHBHolder.getCensorshipFlag (i) == false) {
00571                 lSumOfValues += ioHBHolder.getHistoricalBooking (i);
00572             }
00573         }

```

```

00574         double lFirstAverage = lSumOfValues / lNbOfUsedData;
00575
00576         // Second loop
00577         for (short i = 0; i < lNbOfData; ++i) {
00578             if (ioHBHolder.getCensorshipFlag (i) == true) {
00579                 const stdair::NbOfBookings_T& lBkgs =
00580                     ioHBHolder.getHistoricalBooking (i);
00581                 if (lBkgs >= lFirstAverage) {
00582                     lSumOfValues += lBkgs;
00583                     ++lNbOfUsedData;
00584                 }
00585             }
00586         }
00587         double lSecondAverage = lSumOfValues / lNbOfUsedData;
00588
00589         // Last loop for updating the demand.
00590         for (short i = 0; i < lNbOfData; ++i) {
00591             if (ioHBHolder.getCensorshipFlag (i) == true) {
00592                 const stdair::NbOfBookings_T& lBkgs =
00593                     ioHBHolder.getHistoricalBooking (i);
00594                 if (lBkgs < lSecondAverage) {
00595                     ioHBHolder.setUnconstrainedDemand (lSecondAverage, i);
00596                 }
00597             }
00598         }
00599     }
00600 }
00601 }
00602

```

43.116 rmol/command/Detruncator.hpp File Reference

```

#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>

```

Classes

- class [RMOL::Detruncator](#)

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.117 Detruncator.hpp

```

00001 #ifndef __RMOL_COMMAND_DETRUNCATOR_HPP
00002 #define __RMOL_COMMAND_DETRUNCATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir

```

Generated on Wed Nov 2 2011 23:53:24 for RMOL by Doxygen

```
00096         const stdair::SegmentCabin&, const stdair::Date_T&);
00097
00101     static void retrieveUnconstrainedDemandForFirstDCP
00102     (const stdair::GuillotineBlock&, UnconstrainedDemandVector_T&,
00103      const stdair::DCP_T&, const stdair::BlockIndex_T&,
00104      const stdair::NbOfSegments_T&, const stdair::NbOfSegments_T&);
00105     };
00106 }
00107 #endif // __RMOL_COMMAND_DETRUNCATOR_HPP
00108
00109
```

43.118 rmol/command/Forecaster.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <cmath>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/GuillotineBlock.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/basic/BasConst_Curves.hpp>
#include <rmol/bom/Utilities.hpp>
#include <rmol/bom/GuillotineBlockHelper.hpp>
#include <rmol/bom/HistoricalBookingHolder.hpp>
#include <rmol/bom/HistoricalBooking.hpp>
#include <rmol/bom/EMDetruncator.hpp>
#include <rmol/command/Forecaster.hpp>
#include <rmol/command/Detruncator.hpp>
```

Namespaces

- namespace [RMOL](#)

43.119 Forecaster.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <cmath>
00008 // StdAir
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/basic/RandomGeneration.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/FlightDate.hpp>
00014 #include <stdair/bom/LegDate.hpp>
00015 #include <stdair/bom/SegmentDate.hpp>
00016 #include <stdair/bom/LegCabin.hpp>
00017 #include <stdair/bom/SegmentCabin.hpp>
00018 #include <stdair/bom/GuillotineBlock.hpp>
00019 #include <stdair/bom/BookingClass.hpp>
00020 #include <stdair/service/Logger.hpp>
00021 // RMOL
00022 #include <rmol/basic/BasConst_Curves.hpp>
00023 #include <rmol/bom/Utilities.hpp>
00024 #include <rmol/bom/GuillotineBlockHelper.hpp>
00025 #include <rmol/bom/HistoricalBookingHolder.hpp>
00026 #include <rmol/bom/HistoricalBooking.hpp>
00027 #include <rmol/bom/EMDetruncator.hpp>
00028 #include <rmol/command/Forecaster.hpp>
00029 #include <rmol/command/Detruncator.hpp>
00030
00031 namespace RMOL {
00032
00033 // //////////////////////////////////////
00034 bool Forecaster::
00035 forecastUsingAdditivePickUp (stdair::FlightDate& ioFlightDate,
00036                             const stdair::DateTime_T& iEventTime) {
00037     // Build the offset dates.
00038     const stdair::Date_T& lEventDate = iEventTime.date();
00039     stdair::Date_T lRefDate (2012, boost::gregorian::Jan, 01);
00040
00041     //
00042     bool isSucceeded = true;
00043     const stdair::SegmentDateList_T& lSDList =
00044         stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00045     for (stdair::SegmentDateList_T::const_iterator itSD = lSDList.begin();
00046          itSD != lSDList.end(); ++itSD) {
00047         stdair::SegmentDate* lSD_ptr = *itSD;
00048         assert (lSD_ptr != NULL);
00049
00050         const stdair::Date_T& lBoardingDate = lSD_ptr->getBoardingDate();
00051         const stdair::DateOffset_T lSegmentDateOffset =
00052             lBoardingDate - lEventDate;
00053         const stdair::DTD_T lSegmentDTD = lSegmentDateOffset.days();
00054

```



```

00055     // Build remaining DCP's for the segment-date.
00056     // TODO: treat the case where the segment departure is not the
00057     // same as the flight-date departure.
00058     stdair::DCPList_T lDCPList;
00059
00060     if (lEventDate < lRefDate) {
00061         lDCPList = Utilities::buildRemainingDCPList (lSegmentDTD);
00062     } else {
00063         lDCPList = Utilities::buildRemainingDCPList2 (lSegmentDTD);
00064     }
00065
00066     //
00067     const stdair::SegmentCabinList_T& lSCList =
00068         stdair::BomManager::getList<stdair::SegmentCabin> (*lSD_ptr);
00069     for (stdair::SegmentCabinList_T::const_iterator itSC = lSCList.begin();
00070          itSC != lSCList.end(); ++itSC) {
00071         stdair::SegmentCabin* lSC_ptr = *itSC;
00072         assert (lSC_ptr != NULL);
00073
00074         //
00075         // STDAIR_LOG_NOTIFICATION (ioFlightDate.getDepartureDate()
00076         //                           << ";" << lSegmentDTD);
00077         bool isForecasted = forecastUsingAdditivePickUp (*lSC_ptr, lDCPList,
00078                                                         lEventDate);
00079         if (isForecasted == false) {
00080             isSucceeded = false;
00081         }
00082     }
00083 }
00084
00085 return isSucceeded;
00086 }
00087
00088 // //////////////////////////////////////
00089 bool Forecaster::
00090 forecastUsingAdditivePickUp (stdair::SegmentCabin& ioSegmentCabin,
00091                             const stdair::DCPList_T& iDCPList,
00092                             const stdair::Date_T& iEventDate) {
00093     // Retrieve the number of departed similar segments.
00094     stdair::NbOfSegments_T lNbOfDepartedSegments =
00095         Utilities::getNbOfDepartedSimilarSegments (ioSegmentCabin, iEventDate);
00096     // TODO
00097     if (lNbOfDepartedSegments > 52) lNbOfDepartedSegments = 52;
00098
00099     // DEBUG
00100     STDAIR_LOG_DEBUG ("Nb of similar departed segments: "
00101                      << lNbOfDepartedSegments);
00102
00103     // If the DCP list includes only DTD0 or the number of departed
00104     // segments are less than two, remaining demand for all classes
00105     // will be set to zero.
00106     stdair::DCPList_T::const_iterator itDCP = iDCPList.begin();
00107     assert (itDCP != iDCPList.end());
00108     const stdair::DCP_T& lCurrentDTD = *itDCP;
00109     if (iDCPList.size() == 1 || lNbOfDepartedSegments < 2) {
00110         setRemainingDemandForecastToZero (ioSegmentCabin);
00111         return false;
00112     } else {
00113         // Initialise a holder for the unconstrained demand.
00114         UnconstrainedDemandVector_T lQEquivalentDemandVector (lNbOfDepartedSegments
00115 , 0.0);
00116         BookingClassUnconstrainedDemandVectorMap_T lBkgClassUncDemMap;

```

```

00116     const stdair::BookingClassList_T& lBCList =
00117         stdair::BomManager::getList<stdair::BookingClass> (ioSegmentCabin);
00118     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00119         itBC != lBCList.end(); ++itBC) {
00120         stdair::BookingClass* lBC_ptr = *itBC;
00121         assert (lBC_ptr != NULL);
00122         std::vector<stdair::NbOfRequests_T> lUncDemandVector (lNbOfDepartedSegmen
ts, 0.0);
00123         bool insertionSucceeded = lBkgClassUncDemMap.insert
00124             (BookingClassUnconstrainedDemandVectorMap_T::
00125                 value_type (lBC_ptr, lUncDemandVector)).second;
00126         assert (insertionSucceeded == true);
00127     }
00128
00129     // Build the DCP intervals and unconstrain censored booking figures for
00130     // each interval.
00131     stdair::DCPList_T::const_iterator itNextDCP = itDCP; ++itNextDCP;
00132     for (; itNextDCP != iDCPList.end(); ++itDCP, ++itNextDCP) {
00133         const stdair::DCP_T& lCurrentDCP = *itDCP;
00134         const stdair::DCP_T& lNextDCP = *itNextDCP;
00135
00136         // DEBUG
00137         STDAIR_LOG_DEBUG ("Unconstrain demand for "
00138             << ioSegmentCabin.describeKey()
00139             << " and the DCP's " << lCurrentDCP << ", "
00140             << lNextDCP);
00141         Detruncator::unconstrainUsingAdditivePickUp (ioSegmentCabin,
00142             lBkgClassUncDemMap,
00143             lQEquivalentDemandVector,
00144             lCurrentDCP-l, lNextDCP,
00145             iEventDate);
00146         STDAIR_LOG_DEBUG ("Detruncation successful");
00147     }
00148
00149     // Retrieve the FRAT5 coefficient and compute the sell-up coef.
00150     FRAT5Curve_T::const_iterator itFRAT5 =
00151         DEFAULT_CUMULATIVE_FRAT5_CURVE.lower_bound (lCurrentDTD);
00152     assert (itFRAT5 != DEFAULT_CUMULATIVE_FRAT5_CURVE.end());
00153     const double lFRAT5Coef = itFRAT5->second;
00154     const double lSellUpCoef = -log(0.5) / (lFRAT5Coef - 1);
00155
00156     forecastUsingAdditivePickUp (ioSegmentCabin, lBkgClassUncDemMap,
00157         lQEquivalentDemandVector, lSellUpCoef);
00158     return true;
00159 }
00160 }
00161
00162 // //////////////////////////////////////
00163 void Forecaster::
00164     forecastUsingAdditivePickUp (stdair::SegmentCabin& ioSegmentCabin,
00165         const BookingClassUnconstrainedDemandVectorMap_T&
iClassUncDemMap,
00166         const UnconstrainedDemandVector_T& iUncDemVector,
00167         const double& iSellUpFactor) {
00168     double lPriceOriMean; double lPriceOriStdDev;
00169     Utilities::computeDistributionParameters (iUncDemVector, lPriceOriMean,
00170         lPriceOriStdDev);
00171
00172     // DEBUG
00173     //STDAIR_LOG_NOTIFICATION (lPriceOriMean << ";" << lPriceOriStdDev);
00174
00175     // Retrieve the classes from low to high and compute the distributions of

```

```

00176 // product-oriented and price-oriented demand.
00177 // Retrieve the lowest class.
00178 const stdair::BookingClassList_T& lBCList =
00179     stdair::BomManager::getList<stdair::BookingClass> (ioSegmentCabin);
00180 stdair::BookingClassList_T::const_reverse_iterator itCurrentClass =
00181     lBCList.rbegin();
00182 assert (itCurrentClass != lBCList.rend());
00183 stdair::BookingClassList_T::const_reverse_iterator itNextClass =
00184     itCurrentClass;
00185 ++itNextClass;
00186 // If there is only one class in the cabin, the demand distribution of this
00187 // class is equal to the price-oriented demand distribution of the cabin.
00188 if (itNextClass == lBCList.rend()) {
00189     stdair::BookingClass* lLowestBC_ptr = *itCurrentClass;
00190     lLowestBC_ptr->setMean (lPriceOriMean);
00191     lLowestBC_ptr->setStdDev (lPriceOriStdDev);
00192 } else {
00193     // Compute the demand for higher class using the formula
00194     //  $\text{Pro\_sell\_up\_from\_Q\_to\_F} = e^{((y\_F/y\_Q - 1) * \ln(0.5) / (\text{FRAT5} - 1))}$ 
00195     for (; itNextClass != lBCList.rend(); ++itCurrentClass, ++itNextClass) {
00196         stdair::BookingClass* lCurrentBC_ptr = *itCurrentClass;
00197         assert (lCurrentBC_ptr != NULL);
00198         const stdair::Yield_T& lCurrentYield = lCurrentBC_ptr->getYield();
00199         stdair::BookingClass* lNextBC_ptr = *itNextClass;
00200         assert (lNextBC_ptr != NULL);
00201         const stdair::Yield_T& lNextYield = lNextBC_ptr->getYield();
00202
00203         // Compute the part of price-oriented demand distributed to the
00204         // current class.
00205         const double lSellUp =
00206             exp ((1.0 - lNextYield/lCurrentYield) * lSellUpFactor);
00207         const double lPriceOriDemMeanFrac = lPriceOriMean * (1.0 - lSellUp);
00208         const double lPriceOriDemStdDevFrac = lPriceOriStdDev * (1.0 - lSellUp);
00209
00210         // Compute the product-oriented demand distribution for the
00211         // current class.
00212         BookingClassUnconstrainedDemandVectorMap_T::const_iterator itBCUD =
00213             iClassUncDemMap.find (lCurrentBC_ptr);
00214         assert (itBCUD != iClassUncDemMap.end());
00215         const UnconstrainedDemandVector_T& lDemandVector = itBCUD->second;
00216         double lMean; double lStdDev;
00217         Utilities::computeDistributionParameters(lDemandVector, lMean, lStdDev);
00218
00219         // Compute the demand distribution for the current class;
00220         lMean += lPriceOriDemMeanFrac;
00221         lStdDev = sqrt (lStdDev * lStdDev +
00222             lPriceOriDemStdDevFrac * lPriceOriDemStdDevFrac);
00223         lCurrentBC_ptr->setMean (lMean);
00224         lCurrentBC_ptr->setStdDev (lStdDev);
00225
00226         // DEBUG
00227         // STDAIR_LOG_NOTIFICATION ("Class " << lCurrentBC_ptr->describeKey()
00228         //                             << ", mean = " << lMean
00229         //                             << ", stddev = " << lStdDev);
00230
00231         // Update the price-oriented demand
00232         lPriceOriMean *= lSellUp;
00233         lPriceOriStdDev *= lSellUp;
00234     }
00235
00236     // Compute the demand distribution for the highest class (which is the
00237     // "current class")

```

```

00238     stdair::BookingClass* lCurrentBC_ptr = *itCurrentClass;
00239     assert (lCurrentBC_ptr != NULL);
00240     BookingClassUnconstrainedDemandVectorMap_T::const_iterator itBCUD =
00241         iClassUncDemMap.find (lCurrentBC_ptr);
00242     assert (itBCUD != iClassUncDemMap.end());
00243     const UnconstrainedDemandVector_T& lDemandVector = itBCUD->second;
00244     double lMean; double lStdDev;
00245     Utilities::computeDistributionParameters(lDemandVector, lMean, lStdDev);
00246
00247     // Compute the demand distribution for the current class;
00248     lMean += lPriceOriMean;
00249     lStdDev = sqrt (lStdDev * lStdDev + lPriceOriStdDev * lPriceOriStdDev);
00250     lCurrentBC_ptr->setMean (lMean);
00251     lCurrentBC_ptr->setStdDev (lStdDev);
00252
00253     // DEBUG
00254     // STDAIR_LOG_NOTIFICATION ("Class " << lCurrentBC_ptr->describeKey()
00255     //                          << ", mean = " << lMean
00256     //                          << ", stddev = " << lStdDev);
00257 }
00258 }
00259
00260 // //////////////////////////////////////
00261 void Forecaster::
00262 setRemainingDemandForecastToZero (const stdair::SegmentCabin& iSegmentCabin) {
00263     // Set the demand forecast for all classes to zero.
00264     const stdair::BookingClassList_T& lBCList =
00265         stdair::BomManager::getList<stdair::BookingClass> (iSegmentCabin);
00266     for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00267          itBC != lBCList.end(); ++itBC) {
00268         stdair::BookingClass* lBC_ptr = *itBC;
00269         assert (lBC_ptr != NULL);
00270         lBC_ptr->setMean (0.0);
00271     }
00272 }
00273
00274 // //////////////////////////////////////
00275 bool Forecaster::
00276 forecastUsingMultiplicativePickUp (stdair::FlightDate& ioFlightDate,
00277                                   const stdair::DateTime_T& iEventTime) {
00278     // Build the offset dates.
00279     const stdair::Date_T& lEventDate = iEventTime.date();
00280
00281     //
00282     bool isSucceeded = true;
00283     const stdair::SegmentDateList_T& lSDList =
00284         stdair::BomManager::getList<stdair::SegmentDate> (ioFlightDate);
00285     for (stdair::SegmentDateList_T::const_iterator itSD = lSDList.begin();
00286          itSD != lSDList.end(); ++itSD) {
00287         stdair::SegmentDate* lSD_ptr = *itSD;
00288         assert (lSD_ptr != NULL);
00289
00290         const stdair::Date_T& lBoardingDate = lSD_ptr->getBoardingDate();
00291         const stdair::DateOffset_T lSegmentDateOffset =
00292             lBoardingDate - lEventDate;
00293         const stdair::DTD_T lSegmentDTD = lSegmentDateOffset.days();
00294
00295         //
00296         const stdair::SegmentCabinList_T& lSCList =
00297             stdair::BomManager::getList<stdair::SegmentCabin> (*lSD_ptr);
00298         for (stdair::SegmentCabinList_T::const_iterator itSC = lSCList.begin();
00299              itSC != lSCList.end(); ++itSC) {

```

```

00300         stdair::SegmentCabin* lSC_ptr = *itSC;
00301         assert (lSC_ptr != NULL);
00302
00303         bool isForecasted = forecastUsingMultiplicativePickUp (*lSC_ptr,
00304                                                                lEventDate,
00305                                                                lSegmentDTD);
00306         if (isForecasted == false) {
00307             isSucceeded = false;
00308         }
00309     }
00310 }
00311 return isSucceeded;
00312 }
00313
00314 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00315 bool Forecaster::
00316 forecastUsingMultiplicativePickUp (stdair::SegmentCabin& ioSegmentCabin,
00317                                    const stdair::Date_T& iEventDate,
00318                                    const stdair::DTD_T& iSegmentDTD) {
00319     // Retrieving the number of anterior similar segments.
00320     const stdair::GuillotineBlock& lGuillotineBlock =
00321         ioSegmentCabin.getGuillotineBlock();
00322     stdair::NbOfSegments_T lNbOfAnteriorSimilarSegments =
00323         GuillotineBlockHelper::
00324         getNbOfSegmentAlreadyPassedThisDTD (lGuillotineBlock, iSegmentDTD,
00325                                             iEventDate) - 1;
00326     // Retrieve the number of departed similar segments.
00327     stdair::NbOfSegments_T lNbOfDepartedSegments =
00328         Utilities::getNbOfDepartedSimilarSegments (ioSegmentCabin, iEventDate);
00329     // TODO:
00330     if (lNbOfDepartedSegments > 52) {
00331         lNbOfAnteriorSimilarSegments =
00332             lNbOfAnteriorSimilarSegments - lNbOfDepartedSegments + 52;
00333     }
00334
00335     // DEBUG
00336     STDAIR_LOG_DEBUG ("Nb of anterior similar segments: "
00337                      << lNbOfAnteriorSimilarSegments);
00338
00339     // If the iSegmentDTD is the last DCP or there is no anterior similar
00340     // segment, remaining demand for all classes will be set to zero
00341     stdair::DCPLIST_T::const_reverse_iterator itLastDCP =
00342         stdair::DEFAULT_DCP_LIST.rbegin();
00343     assert (itLastDCP != stdair::DEFAULT_DCP_LIST.rend());
00344     const stdair::DCP_T& lLastDCP = *itLastDCP;
00345     if (lNbOfAnteriorSimilarSegments < 1.0 || iSegmentDTD <= lLastDCP) {
00346         setRemainingDemandForecastToZero (ioSegmentCabin);
00347         return false;
00348     } else {
00349         // Retrieve the booking figures of the first DCP and consider them
00350         // as unconstrained demand figures.
00351         stdair::DCPLIST_T::const_iterator itDCP = stdair::DEFAULT_DCP_LIST.begin();
00352         assert (itDCP != stdair::DEFAULT_DCP_LIST.end());
00353         const stdair::DCP_T& lFirstDCP = *itDCP;
00354
00355         // Initialise the unconstrained demand for classes.
00356         stdair::NbOfSegments_T lNbOfUsableSegments =
00357             GuillotineBlockHelper::
00358             getNbOfSegmentAlreadyPassedThisDTD (lGuillotineBlock, lFirstDCP,
00359                                                 iEventDate);
00360         // TODO
00361         unsigned short lSize = lNbOfUsableSegments;

```

```

00362         if (lNbOfDepartedSegments > 52) {
00363             lSize = lNbOfUsableSegments - lNbOfDepartedSegments + 52;
00364         }
00365
00366         STDAIR_LOG_DEBUG ("Nb of usable similar segments: "
00367             << lNbOfUsableSegments);
00368
00369         UnconstrainedDemandVector_T lQEquivalentDemandVector (lSize, 0.0);
00370         stdair::NbOfBookings_T lCurrentSegmentQEquivalentDemand = 0.0;
00371         BookingClassUnconstrainedDemandVectorMap_T lBkgClassUncDemVectorMap;
00372         BookingClassUnconstrainedDemandMap_T lCurrentSegmentBkgClassDemMap;
00373         const stdair::BookingClassList_T& lBCList =
00374             stdair::BomManager::getList<stdair::BookingClass> (ioSegmentCabin);
00375         for (stdair::BookingClassList_T::const_iterator itBC = lBCList.begin();
00376             itBC != lBCList.end(); ++itBC) {
00377             stdair::BookingClass* lBC_ptr = *itBC;
00378             assert (lBC_ptr != NULL);
00379
00380             UnconstrainedDemandVector_T lUncDemandVector (lSize, 0.0);
00381             bool insertionSucceeded = lBkgClassUncDemVectorMap.
00382                 insert (BookingClassUnconstrainedDemandVectorMap_T::
00383                     value_type (lBC_ptr, lUncDemandVector)).second;
00384             assert (insertionSucceeded == true);
00385             insertionSucceeded =
00386                 lCurrentSegmentBkgClassDemMap.
00387                 insert (BookingClassUnconstrainedDemandMap_T::
00388                     value_type (lBC_ptr, 0.0)).second;
00389             assert (insertionSucceeded == true);
00390         }
00391         Detruncator::
00392             retrieveUnconstrainedDemandForFirstDCP (ioSegmentCabin,
00393                 lBkgClassUncDemVectorMap,
00394                 lQEquivalentDemandVector,
00395                 lFirstDCP, lNbOfUsableSegments,
00396                 lSize);
00397
00398         // Unconstrain the booking figures.
00399         stdair::DCPList_T::const_iterator itNextDCP = itDCP; ++itNextDCP;
00400         while (itNextDCP != stdair::DEFAULT_DCP_LIST.end()) {
00401             const stdair::DCP_T& lCurrentDCP = *itDCP;
00402             const stdair::DCP_T& lNextDCP = *itNextDCP;
00403             if (lCurrentDCP <= iSegmentDTD) {
00404                 break;
00405             }
00406             Detruncator::
00407                 unconstrainUsingMultiplicativePickUp (ioSegmentCabin,
00408                     lBkgClassUncDemVectorMap,
00409                     lQEquivalentDemandVector,
00410                     lCurrentDCP-1, lNextDCP,
00411                     iEventDate,
00412                     lNbOfDepartedSegments);
00413             ++itNextDCP; ++itDCP;
00414         }
00415
00416         // Update the unconstrained demand for all the classes of the current
00417         // segment.
00418         lCurrentSegmentQEquivalentDemand =
00419             lQEquivalentDemandVector.at (lNbOfAnteriorSimilarSegments);
00420         BookingClassUnconstrainedDemandMap_T::iterator itBCUD =
00421             lCurrentSegmentBkgClassDemMap.begin();
00422         for (BookingClassUnconstrainedDemandVectorMap_T::iterator itBCUDV =
00423             lBkgClassUncDemVectorMap.begin();

```

```

00424         itBCUDV != lBkgClassUncDemVectorMap.end(); ++itBCUDV, ++itBCUD) {
00425     assert (itBCUD != lCurrentSegmentBkgClassDemMap.end());
00426     assert (itBCUD->first == itBCUDV->first);
00427     stdair::NbOfRequests_T& lUncDem = itBCUD->second;
00428     UnconstrainedDemandVector_T& lUncDemVector = itBCUDV->second;
00429     lUncDem = lUncDemVector.at (lNbOfAnteriorSimilarSegments);
00430 }
00431
00432 // Forecast the remaining demand for all classes
00433 const stdair::DCP_T& lCurrentDTD = *itDCP;
00434 for (; itNextDCP != stdair::DEFAULT_DCP_LIST.end(); ++itNextDCP, ++itDCP){
00435     const stdair::DCP_T& lCurrentDCP = *itDCP;
00436     const stdair::DCP_T& lNextDCP = *itNextDCP;
00437     forecastUsingMultiplicativePickUp (ioSegmentCabin,
00438         lBkgClassUncDemVectorMap,
00439         lQEquivalentDemandVector,
00440         lCurrentDCP-l, lNextDCP, iEventDate,
00441         lNbOfAnteriorSimilarSegments,
00442         lNbOfDepartedSegments);
00443 }
00444
00445 // Update the remaining demand for all classes
00446 lCurrentSegmentQEquivalentDemand =
00447     lQEquivalentDemandVector.at (lNbOfAnteriorSimilarSegments)
00448     - lCurrentSegmentQEquivalentDemand;
00449 itBCUD = lCurrentSegmentBkgClassDemMap.begin();
00450 for (BookingClassUnconstrainedDemandVectorMap_T::iterator itBCUDV =
00451     lBkgClassUncDemVectorMap.begin();
00452     itBCUDV != lBkgClassUncDemVectorMap.end(); ++itBCUDV, ++itBCUD) {
00453     assert (itBCUD != lCurrentSegmentBkgClassDemMap.end());
00454     assert (itBCUD->first == itBCUDV->first);
00455     stdair::NbOfRequests_T& lUncDem = itBCUD->second;
00456     UnconstrainedDemandVector_T& lUncDemVector = itBCUDV->second;
00457     lUncDem = lUncDemVector.at (lNbOfAnteriorSimilarSegments) - lUncDem;
00458 }
00459
00460 // Retrieve the FRAT5 coefficient and compute the sell-up coef.
00461 FRAT5Curve_T::const_iterator itFRAT5 =
00462     DEFAULT_CUMULATIVE_FRAT5_CURVE.lower_bound (lCurrentDTD);
00463 assert (itFRAT5 != DEFAULT_CUMULATIVE_FRAT5_CURVE.end());
00464 const double lFRAT5Coef = itFRAT5->second;
00465 const double lSellUpCoef = -log(0.5) / (lFRAT5Coef - 1);
00466
00467 return forecastUsingMultiplicativePickUp(ioSegmentCabin,
00468     lCurrentSegmentBkgClassDemMap,
00469     lCurrentSegmentQEquivalentDemand,
00470     lSellUpCoef);
00471 }
00472 }
00473 }
00474
00475 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00476 void Forecaster::forecastUsingMultiplicativePickUp
00477 (const stdair::SegmentCabin& iSegmentCabin,
00478 BookingClassUnconstrainedDemandVectorMap_T& ioBkgClassUncDemMap,
00479 UnconstrainedDemandVector_T& ioQEquivalentDemandVector,
00480 const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,
00481 const stdair::Date_T& iCurrentDate,
00482 const stdair::NbOfSegments_T& iNbOfAnteriorSimilarSegments,
00483 const stdair::NbOfSegments_T& iNbOfDepartedSegments) {
00484
00485     // Retrieve the guillotine block.

```

```

00486     const stdair::GuillotineBlock& lGuillotineBlock =
00487         iSegmentCabin.getGuillotineBlock();
00488
00489     // Build the historical booking holders for the product-oriented bookings
00490     // of the casses and the Q-equivalent (price-oriented) bookings of the cabin
00491     const stdair::NbOfSegments_T lNbOfUsableSegments = GuillotineBlockHelper::
00492         getNbOfSegmentAlreadyPassedThisDTD (lGuillotineBlock, iDCPEnd,
00493             iCurrentDate);
00494
00495     STDAIR_LOG_DEBUG ("Nb of usable similar segments: "
00496         << lNbOfUsableSegments);
00497
00498     if (lNbOfUsableSegments > 0) {
00499
00500         // Parse the booking class list and unconstrain historical bookings.
00501         for (BookingClassUnconstrainedDemandVectorMap_T::iterator itBCUDV =
00502             ioBkgClassUncDemMap.begin(); itBCUDV != ioBkgClassUncDemMap.end();
00503             ++itBCUDV) {
00504             stdair::BookingClass* lBC_ptr = itBCUDV->first;
00505             assert (lBC_ptr != NULL);
00506             const stdair::MapKey_T& lBCKKey = lBC_ptr->describeKey();
00507             const stdair::BlockIndex_T& lBlockIdx =
00508                 lGuillotineBlock.getBlockIndex (lBCKKey);
00509             UnconstrainedDemandVector_T& lUncDemVector = itBCUDV->second;
00510
00511             STDAIR_LOG_DEBUG ("Unconstrain product-oriented bookings for "<<lBCKKey);
00512             forecastUsingMultiplicativePickUp (lGuillotineBlock, lUncDemVector,
00513                 iDCPBegin, iDCPEnd,
00514                 lNbOfUsableSegments, lBlockIdx,
00515                 iNbOfAnteriorSimilarSegments,
00516                 iNbOfDepartedSegments);
00517         }
00518
00519         // Unconstrain the Q-equivalent bookings.
00520         // Retrieve the block index of the segment-cabin.
00521         std::ostream lSCMapKey;
00522         lSCMapKey << stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE
00523             << iSegmentCabin.describeKey();
00524         const stdair::BlockIndex_T& lCabinIdx =
00525             lGuillotineBlock.getBlockIndex (lSCMapKey.str());
00526
00527         STDAIR_LOG_DEBUG ("Unconstrain price-oriented bookings");
00528         forecastUsingMultiplicativePickUp (lGuillotineBlock,
00529             ioQEquivalentDemandVector,
00530             iDCPBegin, iDCPEnd,
00531             lNbOfUsableSegments, lCabinIdx,
00532             iNbOfAnteriorSimilarSegments,
00533             iNbOfDepartedSegments,
00534             iSegmentCabin, iCurrentDate);
00535     }
00536 }
00537
00538 // //////////////////////////////////////
00539 void Forecaster::forecastUsingMultiplicativePickUp
00540 (const stdair::GuillotineBlock& iGuillotineBlock,
00541     UnconstrainedDemandVector_T& ioUncDemVector,
00542     const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,
00543     const stdair::NbOfSegments_T& iNbOfUsableSegments,
00544     const stdair::BlockIndex_T& iBlockIdx,
00545     const stdair::NbOfSegments_T& iNbOfAnteriorSimilarSegments,
00546     const stdair::NbOfSegments_T& iNbOfDepartedSegments) {
00547     // TODO

```



```

00548     stdair::NbOfSegments_T lSegBegin = 0;
00549     if (iNbOfDepartedSegments > 52) {
00550         lSegBegin = iNbOfDepartedSegments - 52;
00551     }
00552     // Retrieve the gross daily booking and availability snapshots.
00553     stdair::ConstSegmentCabinDTRangeSnapshotView_T lBookingView =
00554         iGuillotineBlock.getConstSegmentCabinDTRangeProductAndPriceOrientedBooking
00555         SnapshotView (lSegBegin, iNbOfUsableSegments -1, iDCPEnd, iDCPBegin);
00556     stdair::ConstSegmentCabinDTRangeSnapshotView_T lAvlView =
00557         iGuillotineBlock.getConstSegmentCabinDTRangeAvailabilitySnapshotView (lSeg
00558         Begin, iNbOfUsableSegments -1, iDCPEnd, iDCPBegin);
00559
00560     // Browse the list of segments and build the historical booking holder.
00561     const stdair::ValueTypeIndexMap_T& lVTIdxMap =
00562         iGuillotineBlock.getValueTypeIndexMap();
00563     const unsigned int lNbOfValueTypes = lVTIdxMap.size();
00564     HistoricalBookingHolder lHBHolder;
00565     std::vector<short> lDataIndexList;
00566     for (short i = 0; i < iNbOfUsableSegments-lSegBegin; ++i) {
00567         stdair::Flag_T lCensorshipFlag = false;
00568         stdair::NbOfBookings_T lNbOfHistoricalBkgs = 0.0;
00569         const short lNbOfDTDs = iDCPBegin - iDCPEnd + 1;
00570
00571         // Parse the DTDs during the period
00572         for (short j = 0; j < lNbOfDTDs; ++j) {
00573             // Check if the data has been censored during this day.
00574             // STDAIR_LOG_DEBUG ("i: " << i << ", NbOfValues: " << lNbOfValueTypes
00575             // << ", BlockIdx: " << iBlockIdx << ", j: " << j);
00576             if (lCensorshipFlag == false) {
00577                 if (lAvlView[i*lNbOfValueTypes + iBlockIdx][j] < 1.0) {
00578                     lCensorshipFlag = true;
00579                 }
00580             }
00581
00582             // Get the bookings of the day.
00583             // STDAIR_LOG_DEBUG ("Bookings of the day: " << lBookingView[i*lNbOfValue
00584             Types + iBlockIdx][j]);
00585             lNbOfHistoricalBkgs += lBookingView[i*lNbOfValueTypes + iBlockIdx][j];
00586         }
00587
00588         // If there is no booking till now for this class and for this segment,
00589         // there will be no unconstraining process.
00590         stdair::NbOfRequests_T& lUncDemand = ioUncDemVector.at (i);
00591         if (lUncDemand < 1.0) {
00592             lUncDemand += lNbOfHistoricalBkgs;
00593         } else {
00594             double lBkgDemandFactor = lNbOfHistoricalBkgs / lUncDemand;
00595             HistoricalBooking lHistoricalBkg (lBkgDemandFactor, lCensorshipFlag);
00596             lHBHolder.addHistoricalBooking (lHistoricalBkg);
00597             lDataIndexList.push_back (i);
00598         }
00599
00600         // DEBUG
00601         STDAIR_LOG_DEBUG ("Historical bkgs: " << lNbOfHistoricalBkgs
00602         << ", censored: " << lCensorshipFlag);
00603     }
00604
00605     // DEBUG
00606     STDAIR_LOG_DEBUG ("Unconstrain by multiplicative pick-up using EM");
00607
00608     // Unconstrain the booking figures
00609     Detruncator::unconstrainUsingMultiplicativePickUp (lHBHolder);

```

```

00607
00608     // Update the unconstrained demand vector.
00609     short i = 0;
00610     for (std::vector<short>::iterator itIdx = lDataIndexList.begin();
00611          itIdx != lDataIndexList.end(); ++itIdx, ++i) {
00612         short lIdx = *itIdx;
00613         stdair::NbOfRequests_T& lPastDemand = ioUncDemVector.at (lIdx);
00614         const stdair::NbOfRequests_T& lUncDemandFactorOfThisPeriod =
00615             lHBHolder.getUnconstrainedDemand (i);
00616         lPastDemand *= (1+lUncDemandFactorOfThisPeriod);
00617     }
00618
00619     // Update the unconstrained demand for the current segment.
00620     if (lHBHolder.getNbOfFlights() > 0) {
00621         const stdair::NbOfRequests_T& lUncDemandFactorMean =
00622             lHBHolder.getDemandMean();
00623         stdair::NbOfRequests_T& lPastDemand =
00624             ioUncDemVector.at (iNbOfAnteriorSimilarSegments);
00625         lPastDemand *= (1+lUncDemandFactorMean);
00626     }
00627 }
00628
00629 // //////////////////////////////////////
00630 void Forecaster::forecastUsingMultiplicativePickUp
00631 (const stdair::GuillotineBlock& iGuillotineBlock,
00632  UnconstrainedDemandVector_T& ioUncDemVector,
00633  const stdair::DCP_T& iDCPBegin, const stdair::DCP_T& iDCPEnd,
00634  const stdair::NbOfSegments_T& iNbOfUsableSegments,
00635  const stdair::BlockIndex_T& iBlockIdx,
00636  const stdair::NbOfSegments_T& iNbOfAnteriorSimilarSegments,
00637  const stdair::NbOfSegments_T& iNbOfDepartedSegments,
00638  const stdair::SegmentCabin& iSegmentCabin,
00639  const stdair::Date_T& iCurrentDate) {
00640     // TODO
00641     stdair::NbOfSegments_T lSegBegin = 0;
00642     if (iNbOfDepartedSegments > 52) {
00643         lSegBegin = iNbOfDepartedSegments - 52;
00644     }
00645     // Retrieve the gross daily booking and availability snapshots.
00646     stdair::ConstSegmentCabinDTDRangeSnapshotView_T lBookingView =
00647         iGuillotineBlock.getConstSegmentCabinDTDRangeProductAndPriceOrientedBooking
00648         SnapshotView (lSegBegin, iNbOfUsableSegments - 1, iDCPEnd, iDCPBegin);
00649     stdair::ConstSegmentCabinDTDRangeSnapshotView_T lAvlView =
00650         iGuillotineBlock.getConstSegmentCabinDTDRangeAvailabilitySnapshotView (lSeg
00651         Begin, iNbOfUsableSegments - 1, iDCPEnd, iDCPBegin);
00652
00653     // Browse the list of segments and build the historical booking holder.
00654     const stdair::ValueTypeIndexMap_T& lVTIdxMap =
00655         iGuillotineBlock.getValueTypeIndexMap();
00656     const unsigned int lNbOfValueTypes = lVTIdxMap.size();
00657     HistoricalBookingHolder lHBHolder;
00658     std::vector<short> lDataIndexList;
00659     for (short i = 0; i < iNbOfUsableSegments-lSegBegin; ++i) {
00660         stdair::Flag_T lCensorshipFlag = false;
00661         stdair::NbOfBookings_T lNbOfHistoricalBkgs = 0.0;
00662         const short lNbOfDTDs = iDCPBegin - iDCPEnd + 1;
00663
00664         // Parse the DTDs during the period
00665         for (short j = 0; j < lNbOfDTDs; ++j) {
00666             // Check if the data has been censored during this day.
00667             // STDAIR_LOG_DEBUG ("i: " << i << ", NbOfValues: " << lNbOfValueTypes
00668             //                    << ", BlockIdx: " << iBlockIdx << ", j: " << j);

```

```

00667         if (lCensorshipFlag == false) {
00668             if (lAvlView[i*lNbOfValueTypes + iBlockIdx][j] < 1.0) {
00669                 lCensorshipFlag = true;
00670             }
00671         }
00672     }
00673     // Get the bookings of the day.
00674     // STDAIR_LOG_DEBUG ("Bookings of the day: " << lBookingView[i*lNbOfValue
Types + iBlockIdx][j]);
00675     lNbOfHistoricalBkgs += lBookingView[i*lNbOfValueTypes + iBlockIdx][j];
00676 }
00677
00678 // If there is no booking till now for this class and for this segment,
00679 // there will be no unconstraining process.
00680 stdair::NbOfRequests_T& lUncDemand = ioUncDemVector.at (i);
00681 if (lUncDemand < 1.0) {
00682     lUncDemand += lNbOfHistoricalBkgs;
00683 } else {
00684     double lBkgDemandFactor = lNbOfHistoricalBkgs / lUncDemand;
00685     HistoricalBooking lHistoricalBkg (lBkgDemandFactor, lCensorshipFlag);
00686     lHBHolder.addHistoricalBooking (lHistoricalBkg);
00687     lDataIndexList.push_back (i);
00688 }
00689
00690 // DEBUG
00691 STDAIR_LOG_DEBUG ("Historical bkgs: " << lNbOfHistoricalBkgs
<< ", censored: " << lCensorshipFlag);
00692 }
00693
00694 // DEBUG
00695 STDAIR_LOG_DEBUG ("Unconstrain by multiplicative pick-up using EM");
00696
00697 // Unconstrain the booking figures
00698 Detruncator::unconstrainUsingMultiplicativePickUp (lHBHolder);
00699
00700 // Update the unconstrained demand vector.
00701 // LOG
00702 const stdair::SegmentDate& lSegmentDate = stdair::BomManager::
getParent<stdair::SegmentDate, stdair::SegmentCabin> (iSegmentCabin);
00703 const stdair::FlightDate& lFlightDate = stdair::BomManager::
getParent<stdair::FlightDate, stdair::SegmentDate> (lSegmentDate);
00704 const stdair::Date_T& lDepDate = lFlightDate.getDepartureDate();
00705 const boost::gregorian::date_duration lDD = lDepDate - iCurrentDate;
00706 const long lDTD = lDD.days();
00707 stdair::Date_T lRefDate (2012, boost::gregorian::Jan, 01);
00708 short i = 0;
00709 for (std::vector<short>::iterator itIdx = lDataIndexList.begin();
itIdx != lDataIndexList.end(); ++itIdx, ++i) {
00710     short lIdx = *itIdx;
00711     stdair::NbOfRequests_T& lPastDemand = ioUncDemVector.at (lIdx);
00712     const stdair::NbOfRequests_T& lUncDemandFactorOfThisPeriod =
lHBHolder.getUnconstrainedDemand (i);
00713     const double lUncDemThisPeriod =
lPastDemand * lUncDemandFactorOfThisPeriod;
00714     const double lQEBkgThisPeriod =
lPastDemand * lHBHolder.getHistoricalBooking (i);
00715     lPastDemand *= (1+lUncDemandFactorOfThisPeriod);
00716     if (lDepDate > lRefDate) {
00717         const stdair::DateOffset_T lDateOffset (7 *(52 - i) + 420);
00718         const stdair::Date_T lHDate = lDepDate - lDateOffset;
00719         STDAIR_LOG_NOTIFICATION (boost::gregorian::to_iso_string(lDepDate)
<< ";" << lDTD << ";" << iDCPBegin << ";")

```

```

00728                                     << iDCPEnd << ";";
00729                                     << boost::gregorian::to_iso_string (lHDate)
00730                                     << ";" << lUncDemThisPeriod);
00731             STDAIR_LOG_NOTIFICATION (boost::gregorian::to_iso_string (lDepDate)
00732                                     << ";" << lDTD << ";" << iDCPBegin << ";"
00733                                     << iDCPEnd << ";";
00734                                     << boost::gregorian::to_iso_string (lHDate)
00735                                     << ";" << lQEBkgThisPeriod);
00736     }
00737 }
00738
00739 // Update the unconstrained demand for the current segment.
00740 if (lHBHolder.getNbOfFlights() > 0) {
00741     const stdair::NbOfRequests_T& lUncDemandFactorMean =
00742         lHBHolder.getDemandMean();
00743     stdair::NbOfRequests_T& lPastDemand =
00744         ioUncDemVector.at (iNbOfAnteriorSimilarSegments);
00745     lPastDemand *= (1+lUncDemandFactorMean);
00746 }
00747 }
00748
00749 // //////////////////////////////////////
00750 bool Forecaster::
00751 forecastUsingMultiplicativePickUp (stdair::SegmentCabin& ioSegmentCabin,
00752                                   const BookingClassUnconstrainedDemandMap_T&
00753                                   iClassUncDemMap,
00754                                   const stdair::NbOfRequests_T& iUncDem,
00755                                   const double& iSellUpFactor) {
00756     double lPriceOriMean = iUncDem;
00757     double lPriceOriStdDev = sqrt (iUncDem);
00758     // DEBUG
00759     STDAIR_LOG_DEBUG ("Price-oriented demand: mean = " << lPriceOriMean
00760                     << ", stddev = " << lPriceOriStdDev);
00761
00762     // Retrieve the classes from low to high and compute the distributions of
00763     // product-oriented and price-oriented demand.
00764     // Retrieve the lowest class.
00765     const stdair::BookingClassList_T& lBCList =
00766         stdair::BomManager::getList<stdair::BookingClass> (ioSegmentCabin);
00767     stdair::BookingClassList_T::const_reverse_iterator itCurrentClass =
00768         lBCList.rbegin();
00769     assert (itCurrentClass != lBCList.rend());
00770     stdair::BookingClassList_T::const_reverse_iterator itNextClass =
00771         itCurrentClass;
00772     ++itNextClass;
00773     // If there is only one class in the cabin, the demand distribution of this
00774     // class is equal to the price-oriented demand distribution of the cabin.
00775     if (itNextClass == lBCList.rend()) {
00776         stdair::BookingClass* lLowestBC_ptr = *itCurrentClass;
00777         lLowestBC_ptr->setMean (lPriceOriMean);
00778         lLowestBC_ptr->setStdDev (lPriceOriStdDev);
00779         if (lPriceOriMean > 0) {
00780             return true;
00781         } else {
00782             return false;
00783         }
00784     } else {
00785         bool isSucceeded = false;
00786         // Compute the demand for higher class using the formula
00787         // Pro_sell_up_from_Q_to_F = e ^ ((y_F/y_Q - 1) * ln (0.5) / (FRAT5 - 1))
00788         for (; itNextClass != lBCList.rend(); ++itCurrentClass, ++itNextClass) {

```

```

00789         stdair::BookingClass* lCurrentBC_ptr = *itCurrentClass;
00790         assert (lCurrentBC_ptr != NULL);
00791         const stdair::Yield_T& lCurrentYield = lCurrentBC_ptr->getYield();
00792         stdair::BookingClass* lNextBC_ptr = *itNextClass;
00793         assert (lNextBC_ptr != NULL);
00794         const stdair::Yield_T& lNextYield = lNextBC_ptr->getYield();
00795
00796         // Compute the part of price-oriented demand distributed to the
00797         // current class.
00798         const double lSellUp =
00799             exp ((1.0 - lNextYield/lCurrentYield) * iSellUpFactor);
00800         const double lPriceOriDemMeanFrac = lPriceOriMean * (1.0 - lSellUp);
00801         const double lPriceOriDemStdDevFrac = lPriceOriStdDev * (1.0 - lSellUp);
00802
00803         // Compute the product-oriented demand distribution for the
00804         // current class.
00805         BookingClassUnconstrainedDemandMap_T::const_iterator itBCUD =
00806             iClassUncDemMap.find (lCurrentBC_ptr);
00807         assert (itBCUD != iClassUncDemMap.end());
00808         double lMean = itBCUD->second;
00809         double lStdDev = sqrt (lMean);
00810
00811         // Compute the demand distribution for the current class;
00812         lMean += lPriceOriDemMeanFrac;
00813         lStdDev = sqrt (lStdDev * lStdDev +
00814             lPriceOriDemStdDevFrac * lPriceOriDemStdDevFrac);
00815         lCurrentBC_ptr->setMean (lMean);
00816         lCurrentBC_ptr->setStdDev (lStdDev);
00817
00818         if (lMean > 0) {
00819             isSucceeded = true;
00820         }
00821
00822         // DEBUG
00823         STDAIR_LOG_DEBUG ("Class " << lCurrentBC_ptr->describeKey()
00824             << ", mean = " << lMean
00825             << ", stddev = " << lStdDev);
00826
00827         // Update the price-oriented demand
00828         lPriceOriMean *= lSellUp;
00829         lPriceOriStdDev *= lSellUp;
00830     }
00831
00832     // Compute the demand distribution for the highest class (which is the
00833     // "current class")
00834     stdair::BookingClass* lCurrentBC_ptr = *itCurrentClass;
00835     assert (lCurrentBC_ptr != NULL);
00836     BookingClassUnconstrainedDemandMap_T::const_iterator itBCUD =
00837         iClassUncDemMap.find (lCurrentBC_ptr);
00838     assert (itBCUD != iClassUncDemMap.end());
00839     double lMean = itBCUD->second;
00840     double lStdDev = sqrt (lMean);
00841
00842     // Compute the demand distribution for the current class;
00843     lMean += lPriceOriMean;
00844     lStdDev = sqrt (lStdDev * lStdDev + lPriceOriStdDev * lPriceOriStdDev);
00845     lCurrentBC_ptr->setMean (lMean);
00846     lCurrentBC_ptr->setStdDev (lStdDev);
00847
00848     if (lMean > 0) {
00849         isSucceeded = true;
00850     }

```

```

00851
00852         // DEBUG
00853         STDAIR_LOG_DEBUG ("Class " << lCurrentBC_ptr->describeKey()
00854                         << ", mean = " << lMean
00855                         << ", stddev = " << lStdDev);
00856         return isSucceeded;
00857     }
00858 }
00859
00860 }
```

43.120 rmol/command/Forecaster.hpp File Reference

```

#include <map>
#include <stdair/stdair_inventory_types.hpp>
#include <rmol/RMOL_Types.hpp>
```

Classes

- class [RMOL::Forecaster](#)

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.121 Forecaster.hpp

```

00001 #ifndef __RMOL_COMMAND_FORECASTER_HPP
00002 #define __RMOL_COMMAND_FORECASTER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <map>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 // RMOL
00012 #include <rmol/RMOL_Types.hpp>
00013
00014 // Forward declarations
00015 namespace stdair {
00016     class FlightDate;
00017     class SegmentCabin;
00018     class GuillotineBlock;
00019 }
00020
00021 namespace RMOL {
00022     class Forecaster {
00023     public:
00024         static bool forecastUsingAdditivePickUp (stdair::FlightDate&
```

```

00029                                     const stdair::DateTime_T&);
00030
00034     static bool forecastUsingMultiplicativePickUp (stdair::FlightDate&,
00035                                                     const stdair::DateTime_T&);
00036
00037
00038     private:
00042     static bool forecastUsingAdditivePickUp (stdair::SegmentCabin&,
00043                                               const stdair::DCPList_T&,
00044                                               const stdair::Date_T&);
00045
00050     static void forecastUsingAdditivePickUp (stdair::SegmentCabin&,
00051                                               const
BookingClassUnconstrainedDemandVectorMap_T&, const UnconstrainedDemandVector_T&,
const double&);
00052
00056     static bool forecastUsingMultiplicativePickUp (stdair::SegmentCabin&,
00057                                                     const stdair::Date_T&,
00058                                                     const stdair::DTD_T&);
00062     static void forecastUsingMultiplicativePickUp (const stdair::SegmentCabin&,
00063 BookingClassUnconstrainedDemandVectorMap_T&,
UnconstrainedDemandVector_T&,
const stdair::DCP_T&,
const stdair::DCP_T&,
const stdair::Date_T&,
const stdair::NbOfSegments_T&,
const stdair::NbOfSegments_T&);
00070
00074     static void forecastUsingMultiplicativePickUp (const stdair::GuillotineBlock&,
UnconstrainedDemandVector_T&,
const stdair::DCP_T&,
const stdair::DCP_T&,
const stdair::NbOfSegments_T&,
const stdair::BlockIndex_T&,
const stdair::NbOfSegments_T&,
const stdair::NbOfSegments_T&);
00082
00086     static void forecastUsingMultiplicativePickUp (const stdair::GuillotineBlock&,
UnconstrainedDemandVector_T&,
const stdair::DCP_T&,
const stdair::DCP_T&,
const stdair::NbOfSegments_T&,
const stdair::BlockIndex_T&,
const stdair::NbOfSegments_T&,
const stdair::NbOfSegments_T&,
const stdair::SegmentCabin&,
const stdair::Date_T&);
00096
00101     static bool forecastUsingMultiplicativePickUp (stdair::SegmentCabin&,
00102                                                     const
BookingClassUnconstrainedDemandMap_T&,
const stdair::NbOfRequests_T&,
const double&);
00103
00104
00105
00109     static void setRemainingDemandForecastToZero (const stdair::SegmentCabin&);
00110 };
00111 }

```

```
00112 #endif // __RMOL_COMMAND_FORECASTER_HPP
```

43.122 rmol/command/InventoryParser.cpp File Reference

```
#include <sstream>
#include <fstream>
#include <cassert>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/command/InventoryParser.hpp>
```

Namespaces

- namespace [RMOL](#)

43.123 InventoryParser.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
```



```

00003 // //////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <fstream>
00007 #include <cassert>
00008 // StdAir
00009 #include <stdair/stdair_inventory_types.hpp>
00010 #include <stdair/stdair_maths_types.hpp>
00011 #include <stdair/stdair_exceptions.hpp>
00012 #include <stdair/basic/BasConst_DefaultObject.hpp>
00013 #include <stdair/basic/BasConst_Inventory.hpp>
00014 #include <stdair/basic/BasFileMgr.hpp>
00015 #include <stdair/bom/BomRetriever.hpp>
00016 #include <stdair/bom/BomManager.hpp>
00017 #include <stdair/bom/BomRoot.hpp>
00018 #include <stdair/bom/Inventory.hpp>
00019 #include <stdair/bom/FlightDate.hpp>
00020 #include <stdair/bom/SegmentDate.hpp>
00021 #include <stdair/bom/SegmentCabin.hpp>
00022 #include <stdair/bom/LegDate.hpp>
00023 #include <stdair/bom/LegCabin.hpp>
00024 #include <stdair/bom/BookingClass.hpp>
00025 #include <stdair/bom/VirtualClassStruct.hpp>
00026 #include <stdair/factory/FacBom.hpp>
00027 #include <stdair/factory/FacBomManager.hpp>
00028 #include <stdair/service/Logger.hpp>
00029 // RMOL
00030 #include <rmol/command/InventoryParser.hpp>
00031
00032 namespace RMOL {
00033
00034 // //////////////////////////////////////
00035 bool InventoryParser::
00036     parseInputFileAndBuildBom (const std::string& iInputFileName,
00037                               stdair::BomRoot& ioBomRoot) {
00038     bool hasReadBeenSuccessful = false;
00039
00040     // Check that the file path given as input corresponds to an actual file
00041     const bool doesExistAndIsReadable =
00042         stdair::BasFileMgr::doesExistAndIsReadable (iInputFileName);
00043     if (doesExistAndIsReadable == false) {
00044         std::ostringstream oMessage;
00045         oMessage << "The input file, '" << iInputFileName
00046                 << "', can not be retrieved on the file-system";
00047         throw stdair::FileNotFoundException (oMessage.str());
00048     }
00049
00050     // Retrieve the (sample) leg-cabin
00051     stdair::LegCabin& lLegCabin =
00052         stdair::BomRetriever::retrieveDummyLegCabin (ioBomRoot);
00053
00054     // Retrieve the (sample) segment-cabin
00055     stdair::SegmentCabin& lSegmentCabin =
00056         stdair::BomRetriever::retrieveDummySegmentCabin (ioBomRoot);
00057
00058     // Open the input file
00059     std::ifstream inputFile (iInputFileName.c_str());
00060     if (! inputFile) {
00061         STDAIR_LOG_ERROR ("Can not open input file '" << iInputFileName << "'");
00062         throw new stdair::FileNotFoundException ("Can not open input file '"
00063             + iInputFileName + "'");
00064     }

```

```

00065
00066     char buffer[80];
00067     double dval;
00068     short i = 1;
00069     bool hasAllPArms = true;
00070     stdair::Yield_T lYield;
00071     stdair::MeanValue_T lMean;
00072     stdair::StdDevValue_T lStdDev;
00073     stdair::BookingClassKey lBCKey (stdair::DEFAULT_CLASS_CODE);
00074
00075     while (inputFile.getline (buffer, sizeof (buffer), ';')) {
00076         std::istringstream iStringStr (buffer);
00077
00078         if (i == 1) {
00079             hasAllPArms = true;
00080         }
00081
00082         if (iStringStr >> dval) {
00083             if (i == 1) {
00084                 lYield = dval;
00085                 // std::cout << "Yield[" << i << "]" = '" << dval << "'" << std::endl;
00086             } else if (i == 2) {
00087                 lMean = dval;
00088                 // std::cout << "Mean[" << i << "]" = '" << dval << "'" << std::endl;
00089             } else if (i == 3) {
00090                 lStdDev = dval;
00091                 //std::cout << "stdDev[" << i << "]" = '" << dval << "'" << std::endl;
00092                 i = 0;
00093             }
00094             i++;
00095         } else {
00096             hasAllPArms = false;
00097         }
00098
00099         if (hasAllPArms && i == 1) {
00100             stdair::BookingClass& lBookingClass =
00101                 stdair::FacBom<stdair::BookingClass>::instance().create (lBCKey);
00102             stdair::FacBomManager::addToList (lSegmentCabin, lBookingClass);
00103             lBookingClass.setYield (lYield);
00104             lBookingClass.setMean (lMean);
00105             lBookingClass.setStdDev (lStdDev);
00106
00107             stdair::VirtualClassStruct lVirtualClass (lBookingClass);
00108             lVirtualClass.setYield (lYield);
00109             lVirtualClass.setMean (lMean);
00110             lVirtualClass.setStdDev (lStdDev);
00111             lLegCabin.addVirtualClass (lVirtualClass);
00112         }
00113     }
00114
00115     //
00116     if (!inputFile.eof()) {
00117         STDAIR_LOG_ERROR ("Problem when reading input file '" << iInputFileName
00118             << "'");
00119         return hasReadBeenSuccessful;
00120     }
00121
00122     //
00123     hasReadBeenSuccessful = true;

```

```

00127     return hasReadBeenSuccessful;
00128 }
00129
00130 }
```

43.124 rmol/command/InventoryParser.hpp File Reference

```

#include <string>
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class [RMOL::InventoryParser](#)
Class filling the virtual class list (representing a list of classes/buckets) from a given input inventory.

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.125 InventoryParser.hpp

```

00001 #ifndef __RMOL_CMD_INVENTORYPARSER_HPP
00002 #define __RMOL_CMD_INVENTORYPARSER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011
00013 namespace stdair {
00014     class BomRoot;
00015     class LegCabin;
00016     class SegmentCabin;
00017 }
00018
00019 namespace RMOL {
00020
00025     class InventoryParser : public stdair::CmdAbstract {
00026     public:
00027
00035         static bool parseInputFileAndBuildBom (const std::string& iInputFileName,
00036                                                 stdair::BomRoot&);
00037     };
00038 }
00039 #endif // __RMOL_CMD_INVENTORYPARSER_HPP
```

43.126 rmol/command/Optimiser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/service/Logger.hpp>
#include <rmol/bom/MCOptimiser.hpp>
#include <rmol/bom/Emsr.hpp>
#include <rmol/bom/DPOptimiser.hpp>
#include <rmol/command/Optimiser.hpp>
```

Namespaces

- namespace [RMOL](#)

43.127 Optimiser.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/basic/RandomGeneration.hpp>
00010 #include <stdair/bom/BomManager.hpp>
00011 #include <stdair/bom/FlightDate.hpp>
00012 #include <stdair/bom/LegDate.hpp>
00013 #include <stdair/bom/SegmentDate.hpp>
00014 #include <stdair/bom/LegCabin.hpp>
00015 #include <stdair/bom/SegmentCabin.hpp>
00016 #include <stdair/bom/FareFamily.hpp>
00017 #include <stdair/bom/BookingClass.hpp>
00018 #include <stdair/service/Logger.hpp>
```

```

00019 // RMOL
00020 #include <rmol/bom/MCOptimiser.hpp>
00021 #include <rmol/bom/Emsr.hpp>
00022 #include <rmol/bom/DPOptimiser.hpp>
00023 #include <rmol/command/Optimiser.hpp>
00024
00025 namespace RMOL {
00026
00027 // //////////////////////////////////////
00028 void Optimiser::
00029 optimalOptimisationByMCIntegration (const int K,
00030                                     stdair::LegCabin& ioLegCabin) {
00031     // Retrieve the segment-cabin
00032     const stdair::SegmentCabinList_T lSegmentCabinList =
00033         stdair::BomManager::getList<stdair::SegmentCabin> (ioLegCabin);
00034     stdair::SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00035     assert (itSC != lSegmentCabinList.end());
00036     const stdair::SegmentCabin* lSegmentCabin_ptr = *itSC;
00037     assert (lSegmentCabin_ptr != NULL);
00038
00039     // Retrieve the class list.
00040     const stdair::BookingClassList_T lBookingClassList =
00041         stdair::BomManager::getList<stdair::BookingClass> (*lSegmentCabin_ptr);
00042     stdair::RandomGeneration lSeedGenerator (stdair::DEFAULT_RANDOM_SEED);
00043
00044     // Generate the demand samples for the booking classes.
00045     for (stdair::BookingClassList_T::const_iterator itBC =
00046         lBookingClassList.begin(); itBC != lBookingClassList.end(); ++itBC) {
00047         stdair::RandomSeed_T lRandomSeed =
00048             lSeedGenerator.generateUniform01 () * 1e9;
00049         stdair::BookingClass* lBookingClass_ptr = *itBC;
00050         assert (lBookingClass_ptr != NULL);
00051         lBookingClass_ptr->generateDemandSamples (K, lRandomSeed);
00052
00053         // DEBUG
00054         //STDAIR_LOG_DEBUG ("Generating " << K << " demand samples for the class "
00055         //                  << lBookingClass_ptr->describeKey());
00056     }
00057
00058     // Call the class performing the actual algorithm
00059     MCOptimiser::optimalOptimisationByMCIntegration (ioLegCabin);
00060 }
00061
00062 // //////////////////////////////////////
00063 void Optimiser::optimalOptimisationByDP (stdair::LegCabin& ioLegCabin) {
00064     DPOptimiser::optimalOptimisationByDP (ioLegCabin);
00065 }
00066
00067 // //////////////////////////////////////
00068 void Optimiser::heuristicOptimisationByEmsr (stdair::LegCabin& ioLegCabin) {
00069     Emsr::heuristicOptimisationByEmsr (ioLegCabin);
00070 }
00071
00072 // //////////////////////////////////////
00073 void Optimiser::heuristicOptimisationByEmsrA (stdair::LegCabin& ioLegCabin) {
00074     Emsr::heuristicOptimisationByEmsrA (ioLegCabin);
00075 }
00076
00077 // //////////////////////////////////////
00078 void Optimiser::heuristicOptimisationByEmsrB (stdair::LegCabin& ioLegCabin) {
00079     Emsr::heuristicOptimisationByEmsrB (ioLegCabin);
00080 }

```

```

00081
00082 ///////////////////////////////////////////////////////////////////
00083 void Optimiser::optimise (stdair::FlightDate& ioFlightDate) {
00084     // Browse the leg-cabin list and build the virtual class list for
00085     // each cabin.
00086     const stdair::LegDateList_T& lLDList =
00087         stdair::BomManager::getList<stdair::LegDate> (ioFlightDate);
00088     for (stdair::LegDateList_T::const_iterator itLD = lLDList.begin();
00089         itLD != lLDList.end(); ++itLD) {
00090         stdair::LegDate* lLD_ptr = *itLD;
00091         assert (lLD_ptr != NULL);
00092
00093         //
00094         const stdair::LegCabinList_T& lLCList =
00095             stdair::BomManager::getList<stdair::LegCabin> (*lLD_ptr);
00096         for (stdair::LegCabinList_T::const_iterator itLC = lLCList.begin();
00097             itLC != lLCList.end(); ++itLC) {
00098             stdair::LegCabin* lLC_ptr = *itLC;
00099             assert (lLC_ptr != NULL);
00100
00101             // Build the virtual class list.
00102             buildVirtualClassListForLegBasedOptimisation (*lLC_ptr);
00103
00104             // Optimise using Monte-Carlo Integration method.
00105             optimalOptimisationByMCIntegration (10000, *lLC_ptr);
00106         }
00107     }
00108 }
00109
00110 ///////////////////////////////////////////////////////////////////
00111 void Optimiser::
00112 buildVirtualClassListForLegBasedOptimisation (stdair::LegCabin& ioLegCabin) {
00113     // The map holding all virtual classes to be created.
00114     stdair::VirtualClassMap_T lVirtualClassMap;
00115
00116     // Retrieve the segment-cabin
00117     const stdair::SegmentCabinList_T lSegmentCabinList =
00118         stdair::BomManager::getList<stdair::SegmentCabin> (ioLegCabin);
00119     stdair::SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00120     assert (itSC != lSegmentCabinList.end());
00121     const stdair::SegmentCabin* lSegmentCabin_ptr = *itSC;
00122     assert (lSegmentCabin_ptr != NULL);
00123
00124     // Retrieve the class list.
00125     const stdair::BookingClassList_T lBookingClassList =
00126         stdair::BomManager::getList<stdair::BookingClass> (*lSegmentCabin_ptr);
00127
00128     // Generate the demand samples for the booking classes.
00129     for (stdair::BookingClassList_T::const_iterator itBC =
00130         lBookingClassList.begin(); itBC != lBookingClassList.end(); ++itBC) {
00131         stdair::BookingClass* lBookingClass_ptr = *itBC;
00132         assert (lBookingClass_ptr != NULL);
00133
00134         const stdair::Yield_T& lYield = lBookingClass_ptr->getYield();
00135         stdair::VirtualClassStruct lVirtualClass (*lBookingClass_ptr);
00136         lVirtualClass.setYield (lYield);
00137         lVirtualClass.setMean (lBookingClass_ptr->getMean());
00138         lVirtualClass.setStdDev (lBookingClass_ptr->getStdDev());
00139
00140         lVirtualClassMap.insert (stdair::VirtualClassMap_T::
00141             value_type (lYield, lVirtualClass));
00142     }

```

```

00143
00144     // Browse the virtual class map from high to low yield.
00145     ioLegCabin.emptyVirtualClassList();
00146     for (stdair::VirtualClassMap_T::reverse_iterator itVC =
00147         lVirtualClassMap.rbegin(); itVC != lVirtualClassMap.rend(); ++itVC) {
00148         stdair::VirtualClassStruct& lVC = itVC->second;
00149
00150         ioLegCabin.addVirtualClass (lVC);
00151     }
00152 }
00153
00154 // //////////////////////////////////////
00155 double Optimiser::
00156 optimiseUsingOnDForecast (stdair::FlightDate& ioFlightDate,
00157     const bool& iReduceFluctuations) {
00158     double lMaxBPVariation = 0.0;
00159     // Check if the flight date holds a list of leg dates.
00160     // If so, retrieve it and optimise the cabins.
00161     if (stdair::BomManager::hasList<stdair::LegDate> (ioFlightDate)) {
00162         STDAIR_LOG_DEBUG ("Optimisation for the flight date: "
00163             << ioFlightDate.toString());
00164         const stdair::LegDateList_T& lLDList =
00165             stdair::BomManager::getList<stdair::LegDate> (ioFlightDate);
00166         for (stdair::LegDateList_T::const_iterator itLD = lLDList.begin();
00167             itLD != lLDList.end(); ++itLD) {
00168             stdair::LegDate* lLD_ptr = *itLD;
00169             assert (lLD_ptr != NULL);
00170
00171             //
00172             const stdair::LegCabinList_T& lLCList =
00173                 stdair::BomManager::getList<stdair::LegCabin> (*lLD_ptr);
00174             for (stdair::LegCabinList_T::const_iterator itLC = lLCList.begin();
00175                 itLC != lLCList.end(); ++itLC) {
00176                 stdair::LegCabin* lLC_ptr = *itLC;
00177                 assert (lLC_ptr != NULL);
00178                 MCOptimiser::optimisationByMCIntegration (*lLC_ptr);
00179                 const stdair::BidPrice_T& lCurrentBidPrice =
00180                     lLC_ptr->getCurrentBidPrice();
00181                 const stdair::BidPrice_T& lPreviousBidPrice =
00182                     lLC_ptr->getPreviousBidPrice();
00183                 assert (lPreviousBidPrice != 0);
00184                 const double lBPVariation =
00185                     std::abs((lCurrentBidPrice - lPreviousBidPrice)/lPreviousBidPrice);
00186                 lMaxBPVariation = std::max(lMaxBPVariation, lBPVariation);
00187             }
00188         }
00189     }
00190     return lMaxBPVariation;
00191 }
00192
00193 }

```

43.128 rmol/command/Optimiser.hpp File Reference

```
#include <rmol/RMOL_Types.hpp>
```

Classes

- class [RMOL::Optimiser](#)

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.129 Optimiser.hpp

```

00001 #ifndef __RMOL_COMMAND_OPTIMISER_HPP
00002 #define __RMOL_COMMAND_OPTIMISER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // RMOL
00008 #include <rmol/RMOL_Types.hpp>
00009
00010 // Forward declarations
00011 namespace stdair {
00012     class FlightDate;
00013     class LegCabin;
00014 }
00015
00016 namespace RMOL {
00017     class Optimiser {
00018     public:
00019
00020         static void optimalOptimisationByMCIntegration (const int K,
00021                                                         stdair::LegCabin&);
00022
00023         static void optimalOptimisationByDP (stdair::LegCabin&);
00024
00025         static void heuristicOptimisationByEmsr (stdair::LegCabin&);
00026
00027         static void heuristicOptimisationByEmsrA (stdair::LegCabin&);
00028
00029         static void heuristicOptimisationByEmsrB (stdair::LegCabin&);
00030
00031         static void optimise (stdair::FlightDate&);
00032
00033         static void buildVirtualClassListForLegBasedOptimisation (stdair::LegCabin&);
00034
00035         static double optimiseUsingOnDForecast (stdair::FlightDate&,
00036                                                 const bool& iReduceFluctuations = fal
00037 se);
00038     };
00039 }
00040
00041 #endif // __RMOL_COMMAND_OPTIMISER_HPP

```

43.130 rmol/config/rmol-paths.hpp File Reference

Defines

- #define [PACKAGE](#) "rmol"
- #define [PACKAGE_NAME](#) "RMOL"
- #define [PACKAGE_VERSION](#) "0.25.2"

- `#define PREFIXDIR "/usr"`
- `#define EXEC_PREFIX "/usr"`
- `#define BINDIR "/usr/bin"`
- `#define LIBDIR "/usr/lib64"`
- `#define LIBEXECDIR "/usr/libexec"`
- `#define SBINDIR "/usr/sbin"`
- `#define SYSCONFDIR "/usr/etc"`
- `#define INCLUDEDIR "/usr/include"`
- `#define DATAROOTDIR "/usr/share"`
- `#define DATADIR "/usr/share"`
- `#define DOCDIR "/usr/share/doc/rmol-0.25.2"`
- `#define MANDIR "/usr/share/man"`
- `#define INFODIR "/usr/share/info"`
- `#define HTMLDIR "/usr/share/doc/rmol-0.25.2/html"`
- `#define PDFDIR "/usr/share/doc/rmol-0.25.2/html"`
- `#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"`

43.130.1 Define Documentation

43.130.1.1 `#define PACKAGE "rmol"`

Definition at line 4 of file [rmol-paths.hpp](#).

43.130.1.2 `#define PACKAGE_NAME "RMOL"`

Definition at line 5 of file [rmol-paths.hpp](#).

Referenced by [readConfiguration\(\)](#).

43.130.1.3 `#define PACKAGE_VERSION "0.25.2"`

Definition at line 6 of file [rmol-paths.hpp](#).

Referenced by [readConfiguration\(\)](#).

43.130.1.4 `#define PREFIXDIR "/usr"`

Definition at line 7 of file [rmol-paths.hpp](#).

Referenced by [readConfiguration\(\)](#).

43.130.1.5 `#define EXEC_PREFIX "/usr"`

Definition at line 8 of file [rmol-paths.hpp](#).

43.130.1.6 `#define BINDIR "/usr/bin"`

Definition at line 9 of file [rmol-paths.hpp](#).

43.130.1.7 `#define LIBDIR "/usr/lib64"`

Definition at line 10 of file [rmol-paths.hpp](#).

43.130.1.8 `#define LIBEXECDIR "/usr/libexec"`

Definition at line 11 of file [rmol-paths.hpp](#).

43.130.1.9 `#define SBINDIR "/usr/sbin"`

Definition at line 12 of file [rmol-paths.hpp](#).

43.130.1.10 `#define SYSCONFDIR "/usr/etc"`

Definition at line 13 of file [rmol-paths.hpp](#).

43.130.1.11 `#define INCLUDEDIR "/usr/include"`

Definition at line 14 of file [rmol-paths.hpp](#).

43.130.1.12 `#define DATAROOTDIR "/usr/share"`

Definition at line 15 of file [rmol-paths.hpp](#).

43.130.1.13 `#define DATADIR "/usr/share"`

Definition at line 16 of file [rmol-paths.hpp](#).

43.130.1.14 `#define DOCDIR "/usr/share/doc/rmol-0.25.2"`

Definition at line 17 of file [rmol-paths.hpp](#).

43.130.1.15 `#define MANDIR "/usr/share/man"`

Definition at line 18 of file [rmol-paths.hpp](#).

43.130.1.16 `#define INFODIR "/usr/share/info"`

Definition at line 19 of file [rmol-paths.hpp](#).

43.130.1.17 `#define HTMLDIR "/usr/share/doc/rmol-0.25.2/html"`

Definition at line 20 of file [rmol-paths.hpp](#).

43.130.1.18 `#define PDFDIR "/usr/share/doc/rmol-0.25.2/html"`

Definition at line 21 of file [rmol-paths.hpp](#).

43.130.1.19 `#define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"`

Definition at line 22 of file [rmol-paths.hpp](#).

43.131 rmol-paths.hpp

```
00001 #ifndef __RMOL_PATHS_HPP__
00002 #define __RMOL_PATHS_HPP__
```

```

00003
00004 #define PACKAGE "rmol"
00005 #define PACKAGE_NAME "RMOL"
00006 #define PACKAGE_VERSION "0.25.2"
00007 #define PREFIXDIR "/usr"
00008 #define EXEC_PREFIX "/usr"
00009 #define BINDIR "/usr/bin"
00010 #define LIBDIR "/usr/lib64"
00011 #define LIBEXECDIR "/usr/libexec"
00012 #define SBINDIR "/usr/sbin"
00013 #define SYSCONFDIR "/usr/etc"
00014 #define INCLUDEDIR "/usr/include"
00015 #define DATAROOTDIR "/usr/share"
00016 #define DATADIR "/usr/share"
00017 #define DOCDIR "/usr/share/doc/rmol-0.25.2"
00018 #define MANDIR "/usr/share/man"
00019 #define INFODIR "/usr/share/info"
00020 #define HTMLDIR "/usr/share/doc/rmol-0.25.2/html"
00021 #define PDFDIR "/usr/share/doc/rmol-0.25.2/html"
00022 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"
00023
00024 #endif // __RMOL_PATHS_HPP__

```

43.132 rmol/factory/FacRmolServiceContext.cpp File Reference

```

#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <rmol/factory/FacRmolServiceContext.hpp>
#include <rmol/service/RMOL_ServiceContext.hpp>

```

Namespaces

- namespace [RMOL](#)

43.133 FacRmolServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // RMOL
00009 #include <rmol/factory/FacRmolServiceContext.hpp>
00010 #include <rmol/service/RMOL_ServiceContext.hpp>
00011
00012 namespace RMOL {
00013
00014     FacRmolServiceContext* FacRmolServiceContext::_instance = NULL;
00015
00016     // //////////////////////////////////////
00017     FacRmolServiceContext::~FacRmolServiceContext() {
00018         _instance = NULL;

```

```

00019     }
00020
00021     // //////////////////////////////////////
00022     FacRmolServiceContext& FacRmolServiceContext::instance() {
00023
00024         if (_instance == NULL) {
00025             _instance = new FacRmolServiceContext();
00026             assert (_instance != NULL);
00027
00028             stdair::FacSupervisor::instance().registerServiceFactory (_instance);
00029         }
00030         return *_instance;
00031     }
00032
00033     // //////////////////////////////////////
00034     RMOL_ServiceContext& FacRmolServiceContext::create() {
00035         RMOL_ServiceContext* aServiceContext_ptr = NULL;
00036
00037         aServiceContext_ptr = new RMOL_ServiceContext();
00038         assert (aServiceContext_ptr != NULL);
00039
00040         // The new object is added to the Bom pool
00041         _pool.push_back (aServiceContext_ptr);
00042
00043         return *aServiceContext_ptr;
00044     }
00045
00046 }

```

43.134 rmol/factory/FacRmolServiceContext.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/service/FacServiceAbstract.hpp>

```

Classes

- class [RMOL::FacRmolServiceContext](#)
Factory for the service context.

Namespaces

- namespace [RMOL](#)

43.135 FacRmolServiceContext.hpp

```

00001 #ifndef __RMOL_FAC_FACRMOLSERVICECONTEXT_HPP
00002 #define __RMOL_FAC_FACRMOLSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////

```

```

00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/service/FacServiceAbstract.hpp>
00012
00013 namespace RMOL {
00014
00016     class RMOL_ServiceContext;
00017
00018
00022     class FacRmolServiceContext : public stdair::FacServiceAbstract {
00023     public:
00024
00031         static FacRmolServiceContext& instance();
00032
00039         ~FacRmolServiceContext();
00040
00048         RMOL_ServiceContext& create();
00049
00050
00051     protected:
00057         FacRmolServiceContext() {}
00058
00059
00060     private:
00064         static FacRmolServiceContext* _instance;
00065     };
00066
00067 }
00068 #endif // __RMOL_FAC_FACRMOLSERVICECONTEXT_HPP

```

43.136 rmol/RMOL_Service.hpp File Reference

```

#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <rmol/RMOL_Types.hpp>

```

Classes

- class [RMOL::RMOL_Service](#)
Interface for the [RMOL](#) Services.

Namespaces

- namespace `stdair`
- namespace `RMOL`

43.137 RMOL_Service.hpp

```

00001 #ifndef __RMOL_SVC_RMOL_SERVICE_HPP
00002 #define __RMOL_SVC_RMOL_SERVICE_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_service_types.hpp>
00013 #include <stdair/stdair_maths_types.hpp>
00014 #include <stdair/basic/ForecastingMethod.hpp>
00015 #include <stdair/basic/PartnershipTechnique.hpp>
00016 // RMOL
00017 #include <rmol/RMOL_Types.hpp>
00018
00020 namespace stdair {
00021     class FlightDate;
00022     struct BasLogParams;
00023     struct BasDBParams;
00024     class BomRoot;
00025     class AirlineClassList;
00026     class YieldFeatures;
00027     class Inventory;
00028     class OnDDate;
00029 }
00030
00031 namespace RMOL {
00032
00034     class RMOL_ServiceContext;
00035
00039     class RMOL_Service {
00040     public:
00041         // ////////////////////////////////// Constructors and destructors //////////////////////////////////
00057         RMOL_Service (const stdair::BasLogParams&, const stdair::BasDBParams&);
00058
00070         RMOL_Service (const stdair::BasLogParams&);
00071
00087         RMOL_Service (stdair::STDAIR_ServicePtr_T);
00088
00112         void parseAndLoad (const stdair::CabinCapacity_T& iCabinCapacity,
00113                             const stdair::Filename_T& iDemandAndClassDataFile);
00114
00118         void setUpStudyStatManager();
00119
00123         ~RMOL_Service();
00124
00125
00126     public:
00127         // ////////////////////////////////// Business Methods //////////////////////////////////
00133         void buildSampleBom();

```

```

00134
00138     void optimalOptimisationByMCIntegration (const int K);
00139
00143     void optimalOptimisationByDP();
00144
00148     void heuristicOptimisationByEmsr();
00149
00153     void heuristicOptimisationByEmsrA();
00154
00158     void heuristicOptimisationByEmsrB();
00159
00163     bool optimise (stdair::FlightDate&, const stdair::DateTime_T&,
00164                   const stdair::ForecastingMethod&, const stdair::PartnershipTec
hnique&);
00165
00170     // O&D based forecast
00171     void forecastOnD (const stdair::DateTime_T&);
00172
00173     stdair::YieldFeatures* getYieldFeatures(const stdair::OnDDate&, const stdair:
:CabinCode_T&,
00174                                           stdair::BomRoot&);
00175
00176     void forecastOnD (const stdair::YieldFeatures&, stdair::OnDDate&,
00177                     const stdair::CabinCode_T&, const stdair::DTD_T&,
00178                     stdair::BomRoot&);
00179
00180     void setOnDForecast (const stdair::AirlineClassList&, const stdair::MeanValue
_T&,
00181                         const stdair::StdDevValue_T&, stdair::OnDDate&, const st
dair::CabinCode_T&,
00182                         stdair::BomRoot&);
00183
00184     // Single segment O&D
00185     void setOnDForecast (const stdair::AirlineCode_T&, const stdair::Date_T&, con
st stdair::AirportCode_T&,
00186                         const stdair::AirportCode_T&, const stdair::CabinCode_T&
, const stdair::ClassCode_T&,
00187                         const stdair::MeanValue_T&, const stdair::StdDevValue_T&
, const stdair::Yield_T&, stdair::BomRoot&);
00188
00189     // Multiple segment O&D
00190     void setOnDForecast (const stdair::AirlineCodeList_T&, const stdair::AirlineC
ode_T&, const stdair::Date_T&,
00191                         const stdair::AirportCode_T&, const stdair::AirportCode_
T&, const stdair::CabinCode_T&,
00192                         const stdair::ClassCodeList_T&, const stdair::MeanValue_
T&, const stdair::StdDevValue_T&,
00193                         const stdair::Yield_T&, stdair::BomRoot&);
00194
00195     // Initialise (or re-initialise) the demand projections in all leg cabins
00196     void resetDemandInformation (const stdair::DateTime_T&);
00197
00198     void resetDemandInformation (const stdair::DateTime_T&, const stdair::Invento
ry&);
00199
00200     /* Projection of demand */
00201
00202     // Aggregated demand at booking class level.
00203     void projectAggregatedDemandOnLegCabins(const stdair::DateTime_T&);
00204
00205     // Static rule prorated yield
00206     void projectOnDDemandOnLegCabinsUsingYP(const stdair::DateTime_T&);

```

```

00207
00208     // Displacement-adjusted yield
00209     void projectOnDDemandOnLegCabinsUsingDA(const stdair::DateTime_T&);
00210
00211     // Dynamic yield proration (PF = BP_i/BP_{total}, where BP_{total} = sum(BP_i
00212 ))
00213     void projectOnDDemandOnLegCabinsUsingDYP(const stdair::DateTime_T&);
00214     void projectOnDDemandOnLegCabinsUsingDYP(const stdair::DateTime_T&, const std
00215 air::Inventory&);
00216
00217     // O&D-based optimisation (using demand aggregation or demand aggregation).
00218     void optimiseOnD (const stdair::DateTime_T&);
00219
00220     // O&D-based optimisation using displacement-adjusted yield.
00221     void optimiseOnDUsingRMCooperation (const stdair::DateTime_T&);
00222
00223     // Advanced version of O&D-based optimisation using displacement-adjusted yie
00224 ld.
00225     // Network optimisation instead of separate inventory optimisation.
00226     void optimiseOnDUsingAdvancedRMCooperation (const stdair::DateTime_T&);
00227
00228     // Update bid price and send to partners
00229     void updateBidPrice (const stdair::DateTime_T&);
00230     void updateBidPrice (const stdair::FlightDate&, stdair::BomRoot&);
00231
00232 public:
00233     // //////////// Export support methods ////////////
00234     std::string jsonExport (const stdair::AirlineCode_T&,
00235                             const stdair::FlightNumber_T&,
00236                             const stdair::Date_T& iDepartureDate) const;
00237
00238 public:
00239     // //////////// Display support methods ////////////
00240     std::string csvDisplay() const;
00241
00242 private:
00243     // ////////// Construction and Destruction helper methods //////////
00244     RMOL_Service();
00245
00246     RMOL_Service (const RMOL_Service&);
00247
00248     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00249                                                     const stdair::BasDBParams&);
00250
00251     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&);
00252
00253     void addStdAirService (stdair::STDAIR_ServicePtr_T,
00254                           const bool iOwnStdairService);
00255
00256     void initServiceContext();
00257
00258     void initRmolService();
00259
00260     void finalise();
00261
00262 private:
00263     // ////////// Service Context //////////
00264     RMOL_ServiceContext* _rmolServiceContext;

```



```

00331
00332     stdair::Date_T _previousForecastDate;
00333 };
00334 };
00335 }
00336 #endif // __RMOL_SVC_RMOL_SERVICE_HPP

```

43.138 rmol/RMOL_Types.hpp File Reference

```

#include <map>
#include <vector>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_exceptions.hpp>

```

Classes

- class [RMOL::OverbookingException](#)
Overbooking-related exception.
- class [RMOL::UnconstrainingException](#)
Unconstraining-related exception.
- class [RMOL::ForecastException](#)
Forecast-related exception.
- class [RMOL::OptimisationException](#)
Optimisation-related exception.

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

Typedefs

- typedef boost::shared_ptr< RMOL_Service > [RMOL::RMOL_ServicePtr_T](#)
- typedef std::vector< stdair::NbOfRequests_T > [RMOL::UnconstrainedDemandVector_T](#)
- typedef std::vector< stdair::NbOfBookings_T > [RMOL::BookingVector_T](#)
- typedef std::vector< stdair::Flag_T > [RMOL::FlagVector_T](#)
- typedef std::map< stdair::BookingClass *, UnconstrainedDemandVector_T > [RMOL::BookingClassUnconstrainedDemandVector_T](#)
- typedef std::map< stdair::BookingClass *, stdair::NbOfRequests_T > [RMOL::BookingClassUnconstrainedDemandVector_T](#)
- typedef std::map< const stdair::DTD_T, double > [RMOL::FRAT5Curve_T](#)

43.139 RMOL_Types.hpp

```

00001 #ifndef __RMOL_RMOL_TYPES_HPP
00002 #define __RMOL_RMOL_TYPES_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <map>
00009 #include <vector>
00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012 // StdAir
00013 #include <stdair/stdair_inventory_types.hpp>
00014 #include <stdair/stdair_exceptions.hpp>
00015
00016 // Forward declarations.
00017 namespace stdair {
00018     class BookingClass;
00019 }
00020
00021 namespace RMOL {
00022
00023     // Forward declarations
00024     class RMOL_Service;
00025
00026     // ////////// Exceptions //////////
00031     class OverbookingException : public stdair::RootException {
00032     public:
00034         OverbookingException (const std::string& iWhat)
00035             : stdair::RootException (iWhat) {}
00036     };
00037
00041     class UnconstrainingException : public stdair::RootException {
00042     public:
00044         UnconstrainingException (const std::string& iWhat)
00045             : stdair::RootException (iWhat) {}
00046     };
00047
00051     class ForecastException : public stdair::RootException {
00052     public:
00054         ForecastException (const std::string& iWhat)
00055             : stdair::RootException (iWhat) {}
00056     };
00057
00061     class OptimisationException : public stdair::RootException {
00062     public:
00064         OptimisationException (const std::string& iWhat)
00065             : stdair::RootException (iWhat) {}
00066     };
00067
00068     // ////////// Type definitions //////////
00073     typedef boost::shared_ptr<RMOL_Service> RMOL_ServicePtr_T;
00074
00076     typedef std::vector<stdair::NbOfRequests_T> UnconstrainedDemandVector_T;
00077
00079     typedef std::vector<stdair::NbOfBookings_T> BookingVector_T;
00080
00082     typedef std::vector<stdair::Flag_T> FlagVector_T;

```

```
00083
00086     typedef std::map<stdair::BookingClass*, UnconstrainedDemandVector_T>
        BookingClassUnconstrainedDemandVectorMap_T;
00087
00090     typedef std::map<stdair::BookingClass*, stdair::NbOfRequests_T>
        BookingClassUnconstrainedDemandMap_T;
00091
00093     typedef std::map<const stdair::DTD_T, double> FRAT5Curve_T;
00094
00095 }
00096 #endif // __RMOL_RMOL_TYPES_HPP
```

43.140 rmol/service/RMOL_Service.cpp File Reference

```
#include <cassert>
#include <boost/make_shared.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/basic/ContinuousAttributeLite.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/OnDDate.hpp>
#include <stdair/bom/OnDDateTypes.hpp>
```

```

#include <stdair/command/CmdBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <rmol/basic/BasConst_RMOL_Service.hpp>
#include <rmol/factory/FacRmolServiceContext.hpp>
#include <rmol/command/InventoryParser.hpp>
#include <rmol/command/Optimiser.hpp>
#include <rmol/command/Forecaster.hpp>
#include <rmol/service/RMOL_ServiceContext.hpp>
#include <rmol/RMOL_Service.hpp>

```

Namespaces

- namespace [RMOL](#)

43.141 RMOL_Service.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // Boost
00007 #include <boost/make_shared.hpp>
00008 // StdAir
00009 #include <stdair/stdair_inventory_types.hpp>
00010 #include <stdair/basic/BasChronometer.hpp>
00011 #include <stdair/basic/ContinuousAttributeLite.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/BomRetriever.hpp>
00014 #include <stdair/bom/BomRoot.hpp>
00015 #include <stdair/bom/Inventory.hpp>
00016 #include <stdair/bom/FlightDate.hpp>
00017 #include <stdair/bom/LegCabin.hpp>
00018 #include <stdair/bom/LegDate.hpp>
00019 #include <stdair/bom/YieldFeatures.hpp>
00020 #include <stdair/bom/AirportPair.hpp>
00021 #include <stdair/bom/PosChannel.hpp>
00022 #include <stdair/bom/DatePeriod.hpp>
00023 #include <stdair/bom/TimePeriod.hpp>
00024 #include <stdair/bom/AirlineClassList.hpp>
00025 #include <stdair/basic/BasConst_Request.hpp>
00026 #include <stdair/basic/BasConst_Inventory.hpp>
00027 #include <stdair/bom/Inventory.hpp>
00028 #include <stdair/bom/FlightDate.hpp>
00029 #include <stdair/bom/SegmentDate.hpp>
00030 #include <stdair/bom/SegmentCabin.hpp>
00031 #include <stdair/bom/BookingClass.hpp>
00032 #include <stdair/bom/OnDDate.hpp>
00033 #include <stdair/bom/OnDDateTypes.hpp>
00034 #include <stdair/command/CmdBomManager.hpp>

```

```

00035 #include <stdair/service/Logger.hpp>
00036 #include <stdair/STDAIR_Service.hpp>
00037 // RMOL
00038 #include <rmol/basic/BasConst_RMOL_Service.hpp>
00039 #include <rmol/factory/FacRmolServiceContext.hpp>
00040 #include <rmol/command/InventoryParser.hpp>
00041 #include <rmol/command/Optimiser.hpp>
00042 #include <rmol/command/Forecaster.hpp>
00043 #include <rmol/service/RMOL_ServiceContext.hpp>
00044 #include <rmol/RMOL_Service.hpp>
00045
00046 namespace RMOL {
00047
00048 // //////////////////////////////////////
00049 RMOL_Service::RMOL_Service()
00050 : _rmolServiceContext (NULL),
00051   _previousForecastDate (stdair::Date_T (2000, 1, 1)) {
00052     assert (false);
00053 }
00054
00055 // //////////////////////////////////////
00056 RMOL_Service::RMOL_Service (const RMOL_Service& iService) :
00057   _rmolServiceContext (NULL),
00058   _previousForecastDate (stdair::Date_T (2000, 1, 1)) {
00059     assert (false);
00060 }
00061
00062 // //////////////////////////////////////
00063 RMOL_Service::RMOL_Service (const stdair::BasLogParams& iLogParams) :
00064   _rmolServiceContext (NULL),
00065   _previousForecastDate (stdair::Date_T (2000, 1, 1)) {
00066
00067     // Initialise the STDAIR service handler
00068     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00069       initStdAirService (iLogParams);
00070
00071     // Initialise the service context
00072     initServiceContext();
00073
00074     // Add the StdAir service context to the RMOL service context
00075     // \note RMOL owns the STDAIR service resources here.
00076     const bool ownStdairService = true;
00077     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00078
00079     // Initialise the (remaining of the) context
00080     initRmolService();
00081 }
00082
00083 // //////////////////////////////////////
00084 RMOL_Service::RMOL_Service (const stdair::BasLogParams& iLogParams,
00085                             const stdair::BasDBParams& iDBParams) :
00086   _rmolServiceContext (NULL),
00087   _previousForecastDate (stdair::Date_T (2000, 1, 1)) {
00088
00089     // Initialise the STDAIR service handler
00090     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00091       initStdAirService (iLogParams, iDBParams);
00092
00093     // Initialise the service context
00094     initServiceContext();
00095
00096     // Add the StdAir service context to the RMOL service context

```

```

00097     // \note RMOL owns the STDAIR service resources here.
00098     const bool ownStdairService = true;
00099     addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00100
00101     // Initialise the (remaining of the) context
00102     initRmolService();
00103 }
00104
00105 // //////////////////////////////////////
00106 RMOL_Service::RMOL_Service (stdair::STDAIR_ServicePtr_T ioSTDAIRServicePtr)
00107     : _rmolServiceContext (NULL),
00108       _previousForecastDate (stdair::Date_T (2000, 1, 1)) {
00109
00110     // Initialise the context
00111     initServiceContext();
00112
00113     // Add the StdAir service context to the RMOL service context.
00114     // \note RMOL does not own the STDAIR service resources here.
00115     const bool doesNotOwnStdairService = false;
00116     addStdAirService (ioSTDAIRServicePtr, doesNotOwnStdairService);
00117
00118     // Initialise the (remaining of the) context
00119     initRmolService();
00120 }
00121
00122 // //////////////////////////////////////
00123 RMOL_Service::~RMOL_Service() {
00124     // Delete/Clean all the objects from memory
00125     finalise();
00126 }
00127
00128 // //////////////////////////////////////
00129 void RMOL_Service::finalise() {
00130     assert (_rmolServiceContext != NULL);
00131     // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00132     _rmolServiceContext->reset();
00133 }
00134
00135 // //////////////////////////////////////
00136 void RMOL_Service::initServiceContext() {
00137     // Initialise the service context
00138     RMOL_ServiceContext& lRMOL_ServiceContext =
00139         FacRmolServiceContext::instance().create();
00140     _rmolServiceContext = &lRMOL_ServiceContext;
00141 }
00142
00143 // //////////////////////////////////////
00144 void RMOL_Service::
00145 addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00146                  const bool iOwnStdairService) {
00147
00148     // Retrieve the RMOL service context
00149     assert (_rmolServiceContext != NULL);
00150     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
00151
00152     // Store the STDAIR service object within the (AIRINV) service context
00153     lRMOL_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00154                                             iOwnStdairService);
00155 }
00156
00157 // //////////////////////////////////////
00158 stdair::STDAIR_ServicePtr_T RMOL_Service::

```

```

00159     initStdAirService (const stdair::BasLogParams& iLogParams) {
00160
00168         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00169             boost::make_shared<stdair::STDAIR_Service> (iLogParams);
00170
00171         return lSTDAIR_Service_ptr;
00172     }
00173
00174     // ////////////////////////////////////////
00175     stdair::STDAIR_ServicePtr_T RMOL_Service::
00176     initStdAirService (const stdair::BasLogParams& iLogParams,
00177         const stdair::BasDBParams& iDBParams) {
00178
00186         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00187             boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00188
00189         return lSTDAIR_Service_ptr;
00190     }
00191
00192     // ////////////////////////////////////////
00193     void RMOL_Service::initRmolService() {
00194         // Do nothing at this stage. A sample BOM tree may be built by
00195         // calling the buildSampleBom() method
00196     }
00197
00198     // ////////////////////////////////////////
00199     void RMOL_Service::
00200     parseAndLoad (const stdair::CabinCapacity_T& iCabinCapacity,
00201         const stdair::Filename_T& iInputFileName) {
00202
00203         // Retrieve the RMOL service context
00204         if (_rmolServiceContext == NULL) {
00205             throw stdair::NonInitialisedServiceException ("The RMOL service has not"
00206                 " been initialised");
00207         }
00208         assert (_rmolServiceContext != NULL);
00209         RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
00210
00211         // Retrieve the StdAir service object from the (RMOL) service context
00212         stdair::STDAIR_Service& lSTDAIR_Service =
00213             lRMOL_ServiceContext.getSTDAIR_Service();
00214         stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00215
00216         // Build a dummy inventory with a leg-cabin which has the given capacity.
00217         lSTDAIR_Service.buildDummyInventory (iCabinCapacity);
00218
00219         // Complete the BOM tree with the optimisation problem specification
00220         InventoryParser::parseInputFileAndBuildBom (iInputFileName, lBomRoot);
00221     }
00222
00223     // ////////////////////////////////////////
00224     void RMOL_Service::buildSampleBom() {
00225
00226         // Retrieve the RMOL service context
00227         if (_rmolServiceContext == NULL) {
00228             throw stdair::NonInitialisedServiceException ("The RMOL service has not"
00229                 " been initialised");
00230         }
00231         assert (_rmolServiceContext != NULL);
00232
00233         // Retrieve the RMOL service context and whether it owns the Stdair
00234         // service

```

```

00235     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
00236     const bool doesOwnStdairService =
00237         lRMOL_ServiceContext.getOwnStdairServiceFlag();
00238
00239     // Retrieve the StdAir service object from the (RMOL) service context
00240     stdair::STDAIR_Service& lSTDAIR_Service =
00241         lRMOL_ServiceContext.getSTDAIR_Service();
00242
00243     if (doesOwnStdairService == true) {
00244         //
00245         lSTDAIR_Service.buildSampleBom();
00246     }
00247 }
00248
00249 // //////////////////////////////////////
00250 void RMOL_Service::optimalOptimisationByMCIntegration (const int K) {
00251     assert (_rmolServiceContext != NULL);
00252     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
00253
00254     // Retrieve the StdAir service
00255     stdair::STDAIR_Service& lSTDAIR_Service =
00256         lRMOL_ServiceContext.getSTDAIR_Service();
00257     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00258
00259     //
00260     stdair::LegCabin& lLegCabin =
00261         stdair::BomRetriever::retrieveDummyLegCabin (lBomRoot);
00262
00263     stdair::BasChronometer lOptimisationChronometer;
00264     lOptimisationChronometer.start();
00265
00266     Optimiser::optimalOptimisationByMCIntegration (K, lLegCabin);
00267
00268     const double lOptimisationMeasure = lOptimisationChronometer.elapsed();
00269
00270     // DEBUG
00271     STDAIR_LOG_DEBUG ("Optimisation by Monte-Carlo performed in "
00272         << lOptimisationMeasure);
00273     STDAIR_LOG_DEBUG ("Result: " << lLegCabin.displayVirtualClassList());
00274
00275     std::ostream logStream;
00276     stdair::BidPriceVector_T lBidPriceVector = lLegCabin.getBidPriceVector();
00277     logStream << "Bid-Price Vector (BPV): ";
00278     unsigned int size = lBidPriceVector.size();
00279
00280     for (unsigned int i = 0; i < size - 1; ++i) {
00281         const double bidPrice = lBidPriceVector.at(i);
00282         logStream << std::fixed << std::setprecision (2) << bidPrice << ", ";
00283     }
00284     const double bidPrice = lBidPriceVector.at(size - 1);
00285     logStream << std::fixed << std::setprecision (2) << bidPrice;
00286     STDAIR_LOG_DEBUG (logStream.str());
00287 }
00288
00289 // //////////////////////////////////////
00290 void RMOL_Service::optimalOptimisationByDP() {
00291 }
00292
00293 // //////////////////////////////////////
00294 void RMOL_Service::heuristicOptimisationByEmsr() {
00295     assert (_rmolServiceContext != NULL);

```



```

00318     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
00319
00320     // Retrieve the StdAir service
00321     stdair::STDAIR_Service& lSTDAIR_Service =
00322         lRMOL_ServiceContext.getSTDAIR_Service();
00323     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00324
00325     //
00326     stdair::LegCabin& lLegCabin =
00327         stdair::BomRetriever::retrieveDummyLegCabin (lBomRoot);
00328
00329     stdair::BasChronometer lOptimisationChronometer;
00330     lOptimisationChronometer.start();
00331
00332     Optimiser::heuristicOptimisationByEmsr (lLegCabin);
00333
00334     const double lOptimisationMeasure = lOptimisationChronometer.elapsed();
00335     // DEBUG
00336     STDAIR_LOG_DEBUG ("Optimisation EMSR performed in "
00337         << lOptimisationMeasure);
00338     STDAIR_LOG_DEBUG ("Result: " << lLegCabin.displayVirtualClassList());
00339
00340     stdair::BidPriceVector_T lBidPriceVector = lLegCabin.getBidPriceVector();
00341     std::ostringstream logStream;
00342     logStream << "Bid-Price Vector (BPV): ";
00343     unsigned int idx = 0;
00344     for (stdair::BidPriceVector_T::const_iterator itBP = lBidPriceVector.begin();
00345
00346         itBP != lBidPriceVector.end(); ++itBP) {
00347         if (idx != 0) {
00348             logStream << ", ";
00349         }
00350         const stdair::BidPrice_T& lBidPrice = *itBP;
00351         logStream << std::fixed << std::setprecision (2) << lBidPrice;
00352     }
00353     // DEBUG
00354     STDAIR_LOG_DEBUG (logStream.str());
00355 }
00356
00357 // //////////////////////////////////////
00358 void RMOL_Service::heuristicOptimisationByEmsrA() {
00359     assert (_rmolServiceContext != NULL);
00360     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
00361
00362     // Retrieve the StdAir service
00363     stdair::STDAIR_Service& lSTDAIR_Service =
00364         lRMOL_ServiceContext.getSTDAIR_Service();
00365     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00366
00367     //
00368     stdair::LegCabin& lLegCabin =
00369         stdair::BomRetriever::retrieveDummyLegCabin (lBomRoot);
00370
00371     Optimiser::heuristicOptimisationByEmsrA (lLegCabin);
00372
00373     // DEBUG
00374     STDAIR_LOG_DEBUG ("Result: " << lLegCabin.displayVirtualClassList());
00375 }
00376
00377 // //////////////////////////////////////
00378 void RMOL_Service::heuristicOptimisationByEmsrB() {

```

```

00379     assert (_rmolServiceContext != NULL);
00380     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
00381
00382     // Retrieve the StdAir service
00383     stdair::STDAIR_Service& lSTDAIR_Service =
00384         lRMOL_ServiceContext.getSTDAIR_Service();
00385     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00386
00387     //
00388     stdair::LegCabin& lLegCabin =
00389         stdair::BomRetriever::retrieveDummyLegCabin (lBomRoot);
00390
00391     Optimiser::heuristicOptimisationByEmsrB (lLegCabin);
00392
00393     // DEBUG
00394     STDAIR_LOG_DEBUG ("Result: " << lLegCabin.displayVirtualClassList());
00395 }
00396
00397 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00398 bool RMOL_Service::
00399 optimise (stdair::FlightDate& ioFlightDate,
00400          const stdair::DateTime_T& iRMEEventTime,
00401          const stdair::ForecastingMethod& iForecastingMethod,
00402          const stdair::PartnershipTechnique& iPartnershipTechnique) {
00403
00404
00405     STDAIR_LOG_DEBUG ("Forecast & Optimisation");
00406
00407     const stdair::PartnershipTechnique::EN_PartnershipTechnique& lPartnershipTech
nique =
00408         iPartnershipTechnique.getTechnique();
00409
00410     switch (lPartnershipTechnique) {
00411     case stdair::PartnershipTechnique::RAE_DA:
00412     case stdair::PartnershipTechnique::IBP_DA: {
00413         if (_previousForecastDate < iRMEEventTime.date()) {
00414             forecastOnD (iRMEEventTime);
00415             resetDemandInformation (iRMEEventTime);
00416             projectAggregatedDemandOnLegCabins (iRMEEventTime);
00417             optimiseOnD (iRMEEventTime);
00418         }
00419         break;
00420     }
00421     case stdair::PartnershipTechnique::RAE_YP:
00422     case stdair::PartnershipTechnique::IBP_YP:
00423     case stdair::PartnershipTechnique::IBP_YP_U: {
00424         if (_previousForecastDate < iRMEEventTime.date()) {
00425             forecastOnD (iRMEEventTime);
00426             resetDemandInformation (iRMEEventTime);
00427             projectOnDDemandOnLegCabinsUsingYP (iRMEEventTime);
00428             optimiseOnD (iRMEEventTime);
00429         }
00430         break;
00431     }
00432     case stdair::PartnershipTechnique::RMC: {
00433         if (_previousForecastDate < iRMEEventTime.date()) {
00434             forecastOnD (iRMEEventTime);
00435             resetDemandInformation (iRMEEventTime);
00436             updateBidPrice (iRMEEventTime);
00437             projectOnDDemandOnLegCabinsUsingDYP (iRMEEventTime);
00438             optimiseOnDUsingRMCooperation (iRMEEventTime);
00439         }

```

```

00440         break;
00441     }
00442     case stdair::PartnershipTechnique::A_RMC:{
00443         if (_previousForecastDate < iRMEventTime.date()) {
00444             forecastOnD (iRMEventTime);
00445             resetDemandInformation (iRMEventTime);
00446             updateBidPrice (iRMEventTime);
00447             projectOnDDemandOnLegCabinsUsingDYP (iRMEventTime);
00448             optimiseOnDUsingAdvancedRMCooperation (iRMEventTime);
00449         }
00450         break;
00451     }
00452     case stdair::PartnershipTechnique::NONE:{
00453         // DEBUG
00454         STDAIR_LOG_DEBUG ("Forecast");
00455
00456         // 1. Forecast
00457         bool isForecasted = false;
00458         const stdair::ForecastingMethod& lForecastingMethod=
00459             iForecastingMethod.getMethod();
00460         switch (lForecastingMethod) {
00461             case stdair::ForecastingMethod::ADD_PK: {
00462                 isForecasted = Forecaster::forecastUsingAdditivePickUp (ioFlightDate,
00463                                                                         iRMEventTime);
00464                 break;
00465             }
00466             case stdair::ForecastingMethod::MUL_PK: {
00467                 isForecasted =
00468                     Forecaster::forecastUsingMultiplicativePickUp (ioFlightDate,
00469                                                                     iRMEventTime);
00470                 break;
00471             }
00472             default: {
00473                 assert (false);
00474                 break;
00475             }
00476         }
00477
00478         // DEBUG
00479         STDAIR_LOG_DEBUG ("Forecast successful: " << isForecasted);
00480
00481         // 2. Optimisation
00482         if (isForecasted == true) {
00483             // DEBUG
00484             STDAIR_LOG_DEBUG ("Optimise");
00485
00486             Optimiser::optimise (ioFlightDate);
00487             return true;
00488         }
00489         break;
00490     }
00491     default:{
00492         assert (false);
00493         break;
00494     }
00495 }
00496 return false;
00497 }
00498
00499 // //////////////////////////////////////
00500 void RMOL_Service::forecastOnD (const stdair::DateTime_T& iRMEventTime) {
00501

```

```

00502     if (_rmolServiceContext == NULL) {
00503         throw stdair::NonInitialisedServiceException ("The Rmol service "
00504                                                     "has not been initialised");
00505     }
00506     assert (_rmolServiceContext != NULL);
00507     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
00508
00509     // Retrieve the bom root
00510     stdair::STDAIR_Service& lSTDAIR_Service =
00511         lRMOL_ServiceContext.getSTDAIR_Service();
00512     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00513
00514     // Retrieve the date from the RM event
00515     const stdair::Date_T lDate = iRMEventTime.date();
00516
00517     _previousForecastDate = lDate;
00518
00519     const stdair::InventoryList_T& lInventoryList =
00520         stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
00521     assert (!lInventoryList.empty());
00522     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin();
00523          itInv != lInventoryList.end(); ++itInv) {
00524         const stdair::Inventory* lInventory_ptr = *itInv;
00525         assert (lInventory_ptr != NULL);
00526         if (stdair::BomManager::hasList<stdair::OnDDate> (*lInventory_ptr)) {
00527             const stdair::OnDDateList_T lOnDDateList =
00528                 stdair::BomManager::getList<stdair::OnDDate> (*lInventory_ptr);
00529
00530             for (stdair::OnDDateList_T::const_iterator itOD = lOnDDateList.begin();
00531                  itOD != lOnDDateList.end(); ++itOD) {
00532                 stdair::OnDDate* lOnDDate_ptr = *itOD;
00533                 assert (lOnDDate_ptr != NULL);
00534
00535                 const stdair::Date_T& lDepartureDate = lOnDDate_ptr->getDate();
00536                 stdair::DateOffset_T lDateOffset = lDepartureDate - lDate;
00537                 stdair::DTD_T lDTD = short (lDateOffset.days());
00538
00539                 stdair::DCPList_T::const_iterator itDCP =
00540                     std::find (stdair::DEFAULT_DCP_LIST.begin(),
00541                                stdair::DEFAULT_DCP_LIST.end(), lDTD);
00542                 // Check if the forecast for this O&D date needs to be forecasted.
00543                 if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
00544                     // Retrieve the total forecast map.
00545                     const stdair::CabinForecastMap_T& lTotalForecastMap =
00546                         lOnDDate_ptr->getTotalForecastMap();
00547
00548                     // Browse the map and make a forecast for every cabin.
00549                     for (stdair::CabinForecastMap_T::const_iterator itCF =
00550                          lTotalForecastMap.begin();
00551                          itCF != lTotalForecastMap.end(); ++itCF) {
00552                         const stdair::CabinCode_T lCabinCode = itCF->first;
00553                         stdair::YieldFeatures* lYieldFeatures_ptr =
00554                             getYieldFeatures(*lOnDDate_ptr, lCabinCode, lBomRoot);
00555                         if (lYieldFeatures_ptr == NULL) {
00556                             STDAIR_LOG_ERROR ("Cannot find yield corresponding to "
00557                                                 "<< "the O&D date"
00558                                                 << lOnDDate_ptr->toString()
00559                                                 << " Cabin " << lCabinCode);
00560                             assert (false);
00561                         }
00562                         forecastOnD (*lYieldFeatures_ptr, *lOnDDate_ptr, lCabinCode, lDTD,
00563                                     lBomRoot);

```

```

00564         }
00565     }
00566 }
00567 }
00568 }
00569 }
00570
00571 // //////////////////////////////////////
00572 stdair::YieldFeatures* RMOL_Service::
00573 getYieldFeatures(const stdair::OnDDate& iOnDDate,
00574                 const stdair::CabinCode_T& iCabinCode,
00575                 stdair::BomRoot& iBomRoot) {
00576
00577     const stdair::AirportCode_T& lOrigin = iOnDDate.getOrigin();
00578     const stdair::AirportCode_T& lDestination = iOnDDate.getDestination();
00579
00580     const stdair::Date_T& lDepartureDate = iOnDDate.getDate();
00581
00582     // Build the airport pair key out of O&D and get the airport pair object
00583     const stdair::AirportPairKey lAirportPairKey(lOrigin, lDestination);
00584     stdair::AirportPair* lAirportPair_ptr = stdair::BomManager::
00585         getObjectPtr<stdair::AirportPair> (iBomRoot,
00586                                           lAirportPairKey.toString());
00587     if (lAirportPair_ptr == NULL) {
00588         STDAIR_LOG_ERROR ("Cannot find yield corresponding to the airport "
00589                         << "pair: " << lAirportPairKey.toString());
00590         assert (false);
00591     }
00592
00593     // Retrieve the corresponding date period to lDepartureDate.
00594     const stdair::DatePeriodList_T lDatePeriodList =
00595         stdair::BomManager::getList<stdair::DatePeriod> (*lAirportPair_ptr);
00596     for (stdair::DatePeriodList_T::const_iterator itDatePeriod =
00597         lDatePeriodList.begin();
00598         itDatePeriod != lDatePeriodList.end(); ++itDatePeriod) {
00599         const stdair::DatePeriod* lDatePeriod_ptr = *itDatePeriod;
00600         assert (lDatePeriod_ptr != NULL);
00601
00602         const bool isDepartureDateValid =
00603             lDatePeriod_ptr->isDepartureDateValid (lDepartureDate);
00604
00605         if (isDepartureDateValid == true) {
00606             // Retrieve the PoS-Channel.
00607             // TODO: Use POS and Channel from demand instead of default
00608             const stdair::PosChannelKey lPosChannelKey (stdair::DEFAULT_POS,
00609                                                         stdair::DEFAULT_CHANNEL);
00610             stdair::PosChannel* lPosChannel_ptr = stdair::BomManager::
00611                 getObjectPtr<stdair::PosChannel> (*lDatePeriod_ptr,
00612                                                    lPosChannelKey.toString());
00613             if (lPosChannel_ptr == NULL) {
00614                 STDAIR_LOG_ERROR ("Cannot find yield corresponding to the PoS-"
00615                                 << "Channel: " << lPosChannelKey.toString());
00616                 assert (false);
00617             }
00618             // Retrieve the yield features.
00619             const stdair::TimePeriodList_T lTimePeriodList = stdair::
00620                 BomManager::getList<stdair::TimePeriod> (*lPosChannel_ptr);
00621             for (stdair::TimePeriodList_T::const_iterator itTimePeriod =
00622                 lTimePeriodList.begin();
00623                 itTimePeriod != lTimePeriodList.end(); ++itTimePeriod) {
00624                 const stdair::TimePeriod* lTimePeriod_ptr = *itTimePeriod;
00625                 assert (lTimePeriod_ptr != NULL);

```

```

00626
00627         // TODO: Use trip type from demand instead of default value.
00628         const stdair::YieldFeaturesKey lYieldFeaturesKey (stdair::TRIP_TYPE_ONE
_WAY,
00629                                                         iCabinCode);
00630         stdair::YieldFeatures* oYieldFeatures_ptr = stdair::BomManager::
00631             getObjectPtr<stdair::YieldFeatures>(*lTimePeriod_ptr,
00632                                                 lYieldFeaturesKey.toString());
00633         if (oYieldFeatures_ptr != NULL) {
00634             return oYieldFeatures_ptr;
00635         }
00636     }
00637 }
00638 }
00639 return NULL;
00640
00641 }
00642
00643 // ////////////////////////////////////////
00644 void RMOL_Service::
00645 forecastOnD (const stdair::YieldFeatures& iYieldFeatures,
00646             stdair::OnDDate& iOnDDate,
00647             const stdair::CabinCode_T& iCabinCode,
00648             const stdair::DTD_T& iDTD,
00649             stdair::BomRoot& iBomRoot) {
00650
00651     const stdair::AirlineClassListList_T lAirlineClassListList =
00652         stdair::BomManager::getList<stdair::AirlineClassList> (iYieldFeatures);
00653     assert (lAirlineClassListList.begin() != lAirlineClassListList.end());
00654
00655     // Yield order check
00656     stdair::AirlineClassListList_T::const_iterator itACL =
00657         lAirlineClassListList.begin();
00658     stdair::Yield_T lPreviousYield ((*itACL)->getYield());
00659     ++itACL;
00660     for (; itACL != lAirlineClassListList.end(); ++itACL) {
00661         const stdair::AirlineClassList* lAirlineClassList = *itACL;
00662         const stdair::Yield_T& lYield = lAirlineClassList->getYield();
00663         if (lYield <= lPreviousYield) {
00664             lPreviousYield = lYield;
00665         }
00666         else{
00667             STDAIR_LOG_ERROR ("Yields should be given in a descendant order"
00668                             << " in the yield input file" );
00669             assert (false);
00670         }
00671     }
00672 }
00673 // Proportion factor list initialisation
00674 // Each element corresponds to a yield rule
00675 stdair::ProportionFactorList_T lProportionFactorList;
00676 stdair::ProportionFactor_T lPreviousProportionFactor = 0;
00677
00678 // Retrieve the minimal willingness to pay associated to the demand
00679 const stdair::WTPDemandPair_T& lTotalForecast =
00680     iOnDDate.getTotalForecast (iCabinCode);
00681 const stdair::WTP_T& lMinWTP = lTotalForecast.first;
00682
00683 // Retrieve the remaining percentage of booking requests
00684 const stdair::ContinuousAttributeLite<stdair::FloatDuration_T>
00685     lArrivalPattern (stdair::DEFAULT_DTD_PROB_MAP);
00686

```

```

00687     STDAIR_LOG_DEBUG (lArrivalPattern.displayCumulativeDistribution());
00688     const stdair::Probability_T lRemainingProportion =
00689         lArrivalPattern.getRemainingProportion(-float(iDTD));
00690
00691     // Compute the characteristics (mean and std dev) of the total
00692     // forecast demand to come
00693     const stdair::MeanStdDevPair_T lForecatsMeanStdDevPair =
00694         lTotalForecast.second;
00695     const stdair::MeanValue_T& lMeanValue =
00696         lForecatsMeanStdDevPair.first;
00697     const stdair::MeanValue_T& lRemainingMeanValue =
00698         lRemainingProportion*lMeanValue;
00699     const stdair::StdDevValue_T& lStdDevValue =
00700         lForecatsMeanStdDevPair.second;
00701     const stdair::StdDevValue_T& lRemainingStdDevValue =
00702         lRemainingProportion*lStdDevValue;
00703
00704     // Retrieve the frat5 coef corresponding to the input dtd
00705     stdair::DTDFratMap_T::const_iterator itDFC =
00706         stdair::DEFAULT_DTD_FRAT5COEF_MAP.find(iDTD);
00707     if (itDFC == stdair::DEFAULT_DTD_FRAT5COEF_MAP.end()) {
00708         STDAIR_LOG_ERROR ("Cannot find frat5 coef for DTD = " << iDTD );
00709         assert (false);
00710     }
00711     stdair::RealNumber_T lFrat5Coef =
00712         stdair::DEFAULT_DTD_FRAT5COEF_MAP.at(iDTD);
00713
00714     STDAIR_LOG_DEBUG ("Remaining proportion " << lRemainingProportion
00715         << " Total " << lMeanValue
00716         << " StdDev " << lStdDevValue
00717         << "Frat5 Coef " << lFrat5Coef);
00718
00719     std::ostringstream oStr;
00720     // Compute the "forecast demand to come" proportion by class
00721     itACL = lAirlineClassListList.begin();
00722     for (; itACL != lAirlineClassListList.end(); ++itACL) {
00723         const stdair::AirlineClassList* lAirlineClassList_ptr = *itACL;
00724         const stdair::Yield_T& lYield = lAirlineClassList_ptr->getYield();
00725         stdair::ProportionFactor_T lProportionFactor =
00726             exp ((lYield - lMinWTP)*log(0.5)/(lMinWTP*(lFrat5Coef-1.0)));
00727         // If the yield is smaller than minimal WTP, the factor is greater than 1.
00728         // In that case it should be modified and put to 1.
00729         lProportionFactor = std::min (lProportionFactor, 1.0);
00730         lProportionFactorList.push_back(lProportionFactor - lPreviousProportionFact
00731 or);
00732         lPreviousProportionFactor = lProportionFactor;
00733         oStr << lAirlineClassList_ptr->toString() << lProportionFactor << " ";
00734     }
00735     STDAIR_LOG_DEBUG (oStr.str());
00736
00737     // Sanity check
00738     assert (lAirlineClassListList.size() == lProportionFactorList.size());
00739
00740     STDAIR_LOG_DEBUG ("Forecast for " << iOnDDate.describeKey()
00741         << " " << iDTD << " days to departure");
00742
00743     // store the forecast demand to come characteristics in the booking classes
00744     stdair::ProportionFactorList_T::const_iterator itPF =
00745         lProportionFactorList.begin();
00746     itACL = lAirlineClassListList.begin();
00747     for (; itACL != lAirlineClassListList.end(); ++itACL, ++itPF) {

```

```

00748     const stdair::AirlineClassList* lAirlineClassList_ptr = *itACL;
00749     const stdair::ProportionFactor_T& lProportionFactor = *itPF;
00750     stdair::MeanValue_T lMeanValue = lProportionFactor*lRemainingMeanValue;
00751     stdair::StdDevValue_T lStdDevValue =
00752         lProportionFactor*lRemainingStdDevValue;
00753     setOnDForecast(*lAirlineClassList_ptr, lMeanValue, lStdDevValue,
00754         iOnDDate, iCabinCode, iBomRoot);
00755 }
00756
00757 }
00758
00759 // //////////////////////////////////////
00760 void RMOL_Service::
00761 setOnDForecast (const stdair::AirlineClassList& iAirlineClassList,
00762     const stdair::MeanValue_T& iMeanValue,
00763     const stdair::StdDevValue_T& iStdDevValue,
00764     stdair::OnDDate& iOnDDate,
00765     const stdair::CabinCode_T& iCabinCode,
00766     stdair::BomRoot& iBomRoot) {
00767
00768     const stdair::AirportCode_T& lOrigin = iOnDDate.getOrigin();
00769     const stdair::AirportCode_T& lDestination = iOnDDate.getDestination();
00770
00771     const stdair::Date_T& lDepartureDate = iOnDDate.getDate();
00772
00773     const stdair::AirlineCodeList_T& lAirlineCodeList =
00774         iAirlineClassList.getAirlineCodeList();
00775
00776     // Retrieve the class list (one class per airline)
00777     const stdair::ClassList_StringList_T& lClassList_StringList =
00778         iAirlineClassList.getClassCodeList();
00779     assert (!lClassList_StringList.empty());
00780     stdair::ClassCodeList_T lClassCodeList;
00781     for (stdair::ClassList_StringList_T::const_iterator itCL =
00782         lClassList_StringList.begin();
00783         itCL != lClassList_StringList.end(); ++itCL){
00784         const stdair::ClassList_String_T& lClassList_String = *itCL;
00785         assert (lClassList_String.size() > 0);
00786         stdair::ClassCode_T lFirstClass;
00787         lFirstClass.append (lClassList_String, 0, 1);
00788         lClassCodeList.push_back(lFirstClass);
00789     }
00790
00791     // Sanity check
00792     assert (lAirlineCodeList.size() == lClassCodeList.size());
00793     assert (!lAirlineCodeList.empty());
00794
00795     if (lAirlineCodeList.size() == 1) {
00796         // Store the forecast information in the case of a single segment
00797         stdair::AirlineCode_T lAirlineCode = lAirlineCodeList.front();
00798         stdair::ClassCode_T lClassCode = lClassCodeList.front();
00799         stdair::Yield_T lYield = iAirlineClassList.getYield();
00800         setOnDForecast(lAirlineCode, lDepartureDate, lOrigin,
00801             lDestination, iCabinCode, lClassCode,
00802             iMeanValue, iStdDevValue, lYield, iBomRoot);
00803     } else {
00804         // Store the forecast information in the case of a multiple segment
00805
00806         stdair::Yield_T lYield = iAirlineClassList.getYield();
00807         for (stdair::AirlineCodeList_T::const_iterator itAC =
00808             lAirlineCodeList.begin();
00809             itAC != lAirlineCodeList.end(); ++itAC) {

```



```

00810         const stdair::AirlineCode_T& lAirlineCode = *itAC;
00811         setOnDForecast(lAirlineCodeList, lAirlineCode, lDepartureDate, lOrigin,
00812             lDestination, iCabinCode, lClassCodeList,
00813             iMeanValue, iStdDevValue, lYield, iBomRoot);
00814     }
00815 }
00816 }
00817
00818 // //////////////////////////////////////
00819 void RMOL_Service::
00820 setOnDForecast (const stdair::AirlineCode_T& iAirlineCode,
00821     const stdair::Date_T& iDepartureDate,
00822     const stdair::AirportCode_T& iOrigin,
00823     const stdair::AirportCode_T& iDestination,
00824     const stdair::CabinCode_T& iCabinCode,
00825     const stdair::ClassCode_T& iClassCode,
00826     const stdair::MeanValue_T& iMeanValue,
00827     const stdair::StdDevValue_T& iStdDevValue,
00828     const stdair::Yield_T& iYield,
00829     stdair::BomRoot& iBomRoot) {
00830     stdair::Inventory* lInventory_ptr = iBomRoot.getInventory(iAirlineCode);
00831     if (lInventory_ptr == NULL) {
00832         STDAIR_LOG_ERROR ("Cannot find the inventory corresponding"
00833             << " to the airline" << iAirlineCode) ;
00834         assert(false);
00835     }
00836     const stdair::OnDDateList_T lOnDDateList =
00837         stdair::BomManager::getList<stdair::OnDDate> (*lInventory_ptr);
00838     assert (!lOnDDateList.empty());
00839     bool lFoundOnDDate = false;
00840     for (stdair::OnDDateList_T::const_iterator itOD = lOnDDateList.begin();
00841         itOD != lOnDDateList.end(); ++itOD) {
00842         stdair::OnDDate* lOnDDate_ptr = *itOD;
00843         assert (lOnDDate_ptr != NULL);
00844         const stdair::Date_T& lDepartureDate = lOnDDate_ptr->getDate();
00845         const stdair::AirportCode_T& lOrigin = lOnDDate_ptr->getOrigin();
00846         const stdair::AirportCode_T& lDestination = lOnDDate_ptr->getDestination();
00847
00848         if (!stdair::BomManager::hasList<stdair::SegmentDate> (*lOnDDate_ptr)) {
00849             STDAIR_LOG_ERROR ("The O&D date " << lOnDDate_ptr->describeKey()
00850                 << "has not been correctly initialized : SegmentDate li
00851 st is missing");
00852             assert (false);
00853         }
00854         const stdair::SegmentDateList_T& lSegmentDateList =
00855             stdair::BomManager::getList<stdair::SegmentDate> (*lOnDDate_ptr);
00856         // Check if the the O&D date is the one we are looking for
00857         if (lDepartureDate == iDepartureDate && lOrigin == iOrigin &&
00858             lDestination == iDestination && lSegmentDateList.size() == 1) {
00859             stdair::CabinClassPair_T lCabinClassPair (iCabinCode, iClassCode);
00860             stdair::CabinClassPairList_T lCabinClassPairList;
00861             lCabinClassPairList.push_back(lCabinClassPair);
00862             const stdair::MeanStdDevPair_T lMeanStdDevPair (iMeanValue, iStdDevValue)
00863 ;
00864             const stdair::WTPDemandPair_T lWTPDemandPair (iYield, lMeanStdDevPair);
00865             lOnDDate_ptr->setDemandInformation(lCabinClassPairList, lWTPDemandPair);
00866             lFoundOnDDate = true;
00867             STDAIR_LOG_DEBUG (iAirlineCode << " Class " << iClassCode
00868                 << " Mean " << iMeanValue
00869                 << " Std Dev " << iStdDevValue);
00870             break;
00871         }
00872     }

```

```

00869     }
00870
00871     if (!lFoundOnDDate) {
00872         STDAIR_LOG_ERROR ("Cannot find class " << iClassCode << " in cabin "
00873             << iCabinCode << " for the segment "
00874             << iOrigin << "-" << iDestination << " with"
00875             << " the airline " << iAirlineCode);
00876         assert(false);
00877     }
00878 }
00879
00880 // //////////////////////////////////////
00881 void RMOL_Service::
00882 setOnDForecast (const stdair::AirlineCodeList_T& iAirlineCodeList,
00883     const stdair::AirlineCode_T& iAirlineCode,
00884     const stdair::Date_T& iDepartureDate,
00885     const stdair::AirportCode_T& iOrigin,
00886     const stdair::AirportCode_T& iDestination,
00887     const stdair::CabinCode_T& iCabinCode,
00888     const stdair::ClassCodeList_T& iClassCodeList,
00889     const stdair::MeanValue_T& iMeanValue,
00890     const stdair::StdDevValue_T& iStdDevValue,
00891     const stdair::Yield_T& iYield,
00892     stdair::BomRoot& iBomRoot) {
00893     stdair::Inventory* lInventory_ptr = iBomRoot.getInventory(iAirlineCode);
00894     if (lInventory_ptr == NULL) {
00895         STDAIR_LOG_ERROR ("Cannot find the inventory corresponding"
00896             << " to the airline" << iAirlineCode) ;
00897         assert(false);
00898     }
00899     const stdair::OnDDateList_T lOnDDateList =
00900         stdair::BomManager::getList<stdair::OnDDate> (*lInventory_ptr);
00901     assert (!lOnDDateList.empty());
00902     bool lFoundOnDDate = false;
00903     for (stdair::OnDDateList_T::const_iterator itOD = lOnDDateList.begin();
00904         itOD != lOnDDateList.end(); ++itOD) {
00905         stdair::OnDDate* lOnDDate_ptr = *itOD;
00906         assert (lOnDDate_ptr != NULL);
00907         const stdair::Date_T& lDepartureDate = lOnDDate_ptr->getDate();
00908         const stdair::AirportCode_T& lOrigin = lOnDDate_ptr->getOrigin();
00909         const stdair::AirportCode_T& lDestination = lOnDDate_ptr->getDestination();
00910
00911         if (!stdair::BomManager::hasList<stdair::SegmentDate> (*lOnDDate_ptr)) {
00912             STDAIR_LOG_ERROR ("The O&D date " << lOnDDate_ptr->describeKey()
00913                 << "has not been correctly initialized : SegmentDate li
00914 st is missing");
00915             assert (false);
00916         }
00917         const stdair::SegmentDateList_T& lSegmentDateList =
00918             stdair::BomManager::getList<stdair::SegmentDate> (*lOnDDate_ptr);
00919         // Check if the O&D date might be the one we are looking for.
00920         // There still is a test to go through to see if the combination of airline
00921 s is right.
00922         if (lDepartureDate == iDepartureDate && lOrigin == iOrigin &&
00923             lDestination == iDestination && lSegmentDateList.size() == iAirlineCode
00924 List.size()) {
00925             const stdair::SegmentDateList_T& lSegmentDateList =
00926                 stdair::BomManager::getList<stdair::SegmentDate> (*lOnDDate_ptr);
00927
00928             stdair::AirlineCodeList_T::const_iterator itAC = iAirlineCodeList.begin()
00929 ;
00930             stdair::SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin()

```

```

;
00925     for (; itAC != iAirlineCodeList.end(); ++itAC, ++itSD) {
00926         const stdair::AirlineCode_T lForecastAirlineCode = *itAC;
00927         const stdair::SegmentDate* lSegmentDate_ptr = *itSD;
00928         // Check if the operating airline is a different one and check if it
00929         // is the airline that we are looking for.
00930         const stdair::SegmentDate* lOperatingSegmentDate_ptr =
00931             lSegmentDate_ptr->getOperatingSegmentDate ();
00932         if (lOperatingSegmentDate_ptr != NULL) {
00933             const stdair::FlightDate* lOperatingFD_ptr =
00934                 stdair::BomManager::getParentPtr<stdair::FlightDate>(*lOperatingSeg
mentDate_ptr);
00935             const stdair::AirlineCode_T lOperatingAirlineCode =
00936                 lOperatingFD_ptr->getAirlineCode();
00937             if (lOperatingAirlineCode != lForecastAirlineCode) {
00938                 break;
00939             }
00940         } else {
00941             const stdair::AirlineCode_T lOperatingAirlineCode =
00942                 lOnDDate_ptr->getAirlineCode();
00943             if (lOperatingAirlineCode != lForecastAirlineCode) {
00944                 break;
00945             }
00946         }
00947     }
00948     if (itAC == iAirlineCodeList.end()) {lFoundOnDDate = true;}
00949 }
00950 if (lFoundOnDDate) {
00951     stdair::CabinClassPairList_T lCabinClassPairList;
00952     for (stdair::ClassCodeList_T::const_iterator itCC = iClassCodeList.begin(
);
00953         itCC != iClassCodeList.end(); ++itCC) {
00954         const stdair::ClassCode_T lClassCode = *itCC;
00955         stdair::CabinClassPair_T lCabinClassPair (iCabinCode, lClassCode);
00956         lCabinClassPairList.push_back(lCabinClassPair);
00957     }
00958     const stdair::MeanStdDevPair_T lMeanStdDevPair (iMeanValue, iStdDevValue)
;
00959     const stdair::YieldDemandPair_T lYieldDemandPair (iYield, lMeanStdDevPair
);
00960     lOnDDate_ptr->setDemandInformation(lCabinClassPairList, lYieldDemandPair)
;
00961     lFoundOnDDate = true;
00962     std::ostringstream oACStr;
00963     for (stdair::AirlineCodeList_T::const_iterator itAC = iAirlineCodeList.be
gin();
00964         itAC != iAirlineCodeList.end(); ++itAC) {
00965         if (itAC == iAirlineCodeList.begin()) {
00966             oACStr << *itAC;
00967         }
00968         else {
00969             oACStr << "-" << *itAC;
00970         }
00971     }
00972     std::ostringstream oCCStr;
00973     for (stdair::ClassCodeList_T::const_iterator itCC = iClassCodeList.begin(
);
00974         itCC != iClassCodeList.end(); ++itCC) {
00975         if (itCC == iClassCodeList.begin()) {
00976             oCCStr << *itCC;
00977         }
00978         else {

```

```

00979         oCCStr << "-" << *itCC;
00980     }
00981 }
00982
00983     STDAIR_LOG_DEBUG (oACStr.str() << " Classes " << oCCStr.str()
00984                     << " Mean " << iMeanValue << " Std Dev " << iStdDevValu
e);
00985     break;
00986 }
00987 }
00988 if (!lFoundOnDDate) {
00989     STDAIR_LOG_ERROR ("Cannot find the required multi-segment O&D date: "
00990                     << iOrigin << "-" << iDestination << " " << iDepartureDat
e);
00991     assert(false);
00992 }
00993 }
00994
00995 ///////////////////////////////////////////////////////////////////
00996 void RMOL_Service::
00997 resetDemandInformation (const stdair::DateTime_T& iRMEventTime) {
00998     if (_rmolServiceContext == NULL) {
00999         throw stdair::NonInitialisedServiceException ("The Rmol service "
10000                                                         "has not been initialised");
10001     }
10002     assert (_rmolServiceContext != NULL);
10003     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
10004
10005     // Retrieve the bom root
10006     stdair::STDAIR_Service& lSTDAIR_Service =
10007         lRMOL_ServiceContext.getSTDAIR_Service();
10008     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
10009
10010     const stdair::InventoryList_T lInventoryList =
10011         stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
10012     assert (!lInventoryList.empty());
10013     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin();
10014          itInv != lInventoryList.end(); ++itInv) {
10015         const stdair::Inventory* lInventory_ptr = *itInv;
10016         assert (lInventory_ptr != NULL);
10017         resetDemandInformation (iRMEventTime, *lInventory_ptr);
10018     }
10019 }
10020
10021 ///////////////////////////////////////////////////////////////////
10022 void RMOL_Service::
10023 resetDemandInformation (const stdair::DateTime_T& iRMEventTime,
10024                         const stdair::Inventory& iInventory) {
10025
10026     const stdair::FlightDateList_T lFlightDateList =
10027         stdair::BomManager::getList<stdair::FlightDate> (iInventory);
10028     assert (!lFlightDateList.empty());
10029     for (stdair::FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
10030          itFD != lFlightDateList.end(); ++itFD) {
10031         const stdair::FlightDate* lFlightDate_ptr = *itFD;
10032         assert (lFlightDate_ptr != NULL);
10033
10034         // Retrieve the date from the RM event
10035         const stdair::Date_T lDate = iRMEventTime.date();
10036
10037         const stdair::Date_T& lDepartureDate = lFlightDate_ptr->getDepartureDate();

```

```

01038     stdair::DateOffset_T lDateOffset = lDepartureDate - lDate;
01039     stdair::DTD_T lDTD = short (lDateOffset.days());
01040
01041     stdair::DCPLIST_T::const_iterator itDCP =
01042         std::find (stdair::DEFAULT_DCP_LIST.begin(), stdair::DEFAULT_DCP_LIST.end
01043             (), lDTD);
01044     // Check if the demand forecast info corresponding to this flight date need
01045     // s to be reset.
01046     if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01047         // Check if the flight date holds a list of leg dates.
01048         // If so, find all leg cabin and reset the forecast they are holding.
01049         if (stdair::BomManager::hasList<stdair::LegDate> (*lFlightDate_ptr)) {
01050             const stdair::LegDateList_T lLegDateList =
01051                 stdair::BomManager::getList<stdair::LegDate> (*lFlightDate_ptr);
01052             assert (!lLegDateList.empty());
01053             for (stdair::LegDateList_T::const_iterator itLD = lLegDateList.begin();
01054                 itLD != lLegDateList.end(); ++itLD) {
01055                 const stdair::LegDate* lLegDate_ptr = *itLD;
01056                 assert (lLegDate_ptr != NULL);
01057                 const stdair::LegCabinList_T lLegCabinList =
01058                     stdair::BomManager::getList<stdair::LegCabin> (*lLegDate_ptr);
01059                 assert (!lLegCabinList.empty());
01060                 for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.begi
01061                     n());
01062                 itLC != lLegCabinList.end(); ++itLC) {
01063                     stdair::LegCabin* lLegCabin_ptr = *itLC;
01064                     assert (lLegCabin_ptr != NULL);
01065                     lLegCabin_ptr->emptyYieldLevelDemandMap();
01066                 }
01067             }
01068         }
01069     }
01070
01071     // //////////////////////////////////////
01072     void RMOL_Service::projectAggregatedDemandOnLegCabins(const stdair::DateTime_T&
01073         iRMEEventTime) {
01074         if (_rmolServiceContext == NULL) {
01075             throw stdair::NonInitialisedServiceException ("The Rmol service "
01076                 "has not been initialised");
01077         }
01078         assert (_rmolServiceContext != NULL);
01079         RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
01080
01081         // Retrieve the bom root
01082         stdair::STDAIR_Service& lSTDAIR_Service =
01083             lRMOL_ServiceContext.getSTDAIR_Service();
01084         stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01085
01086         // Retrieve the date from the RM event
01087         const stdair::Date_T lDate = iRMEEventTime.date();
01088
01089         const stdair::InventoryList_T lInventoryList =
01090             stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01091         assert (!lInventoryList.empty());
01092         for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin();
01093             itInv != lInventoryList.end(); ++itInv) {
01094             const stdair::Inventory* lInventory_ptr = *itInv;

```

```

01094     assert (lInventory_ptr != NULL);
01095     const stdair::OnDDateList_T lOnDDateList =
01096         stdair::BomManager::getList<stdair::OnDDate> (*lInventory_ptr);
01097     assert (!lOnDDateList.empty());
01098     for (stdair::OnDDateList_T::const_iterator itOD = lOnDDateList.begin();
01099          itOD != lOnDDateList.end(); ++itOD) {
01100         stdair::OnDDate* lOnDDate_ptr = *itOD;
01101         assert (lOnDDate_ptr != NULL);
01102
01103         const stdair::Date_T& lDepartureDate = lOnDDate_ptr->getDate();
01104         stdair::DateOffset_T lDateOffset = lDepartureDate - lDate;
01105         stdair::DTD_T lDTD = short (lDateOffset.days());
01106
01107         stdair::DCPList_T::const_iterator itDCP =
01108             std::find (stdair::DEFAULT_DCP_LIST.begin(), stdair::DEFAULT_DCP_LIST.e
01109 nd(), lDTD);
01109         // Check if the forecast for this O&D date needs to be projected.
01110         if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01111
01112             // Browse the demand info map.
01113             const stdair::StringDemandStructMap_T& lStringDemandStructMap =
01114                 lOnDDate_ptr->getDemandInfoMap ();
01115             for (stdair::StringDemandStructMap_T::const_iterator itStrDS = lStringD
01116 emandStructMap.begin();
01117                  itStrDS != lStringDemandStructMap.end(); ++itStrDS) {
01118                 std::string lCabinClassPath = itStrDS->first;
01119                 const stdair::YieldDemandPair_T& lYieldDemandPair =
01120                     itStrDS->second;
01121                 const stdair::CabinClassPairList_T& lCabinClassPairList =
01122                     lOnDDate_ptr->getCabinClassPairList (lCabinClassPath);
01123                 const unsigned int lNbOfSegments = lOnDDate_ptr->getNbOfSegments();
01124                 // Sanity check
01125                 assert (lCabinClassPairList.size() == lNbOfSegments);
01126
01127                 const stdair::SegmentDateList_T lOnDSegmentDateList =
01128                     stdair::BomManager::getList<stdair::SegmentDate> (*lOnDDate_ptr);
01129                 // Sanity check
01130                 assert (lOnDSegmentDateList.size() == lNbOfSegments);
01131                 stdair::CabinClassPairList_T::const_iterator itCCP = lCabinClassPairL
01132 ist.begin();
01133                 stdair::SegmentDateList_T::const_iterator itSD = lOnDSegmentDateList.
01134 begin();
01135                 for (; itSD != lOnDSegmentDateList.end(); ++itCCP, ++itSD) {
01136                     const stdair::SegmentDate* lSegmentDate_ptr = *itSD;
01137                     const stdair::SegmentDate* lOperatingSegmentDate_ptr =
01138                         lSegmentDate_ptr->getOperatingSegmentDate ();
01139                     assert (lSegmentDate_ptr != NULL);
01140                     // Only operated legs receive the demand information.
01141                     if (lOperatingSegmentDate_ptr == NULL) {
01142                         const stdair::CabinCode_T lCabinCode = itCCP->first;
01143                         const stdair::ClassCode_T lClassCode = itCCP->second;
01144                         const stdair::SegmentCabin* lSegmentCabin_ptr =
01145                             stdair::BomManager::getObjectPtr<stdair::SegmentCabin> (*lSegme
01146 ntDate_ptr,
01147                                                                 lCabinC
01148 ode);
01149                         assert (lSegmentCabin_ptr != NULL);
01150                         // Retrieve the booking class (level of aggregation of demand).
01151                         // The yield of the class is assigned to all types of demand for
01152 it.
01153                         const stdair::BookingClass* lBookingClass_ptr =
01154                             stdair::BomManager::getObjectPtr<stdair::BookingClass> (*lSegme

```

```

        ntCabin_ptr,
01149
                                                    lClassC
        ode);
01150
        assert (lBookingClass_ptr != NULL);
01151
        const stdair::LegCabinList_T lLegCabinList =
01152
            stdair::BomManager::getList<stdair::LegCabin> (*lSegmentCabin_ptr);
01153
        assert (!lLegCabinList.empty());
01154
        const int lNbOfLegs = lLegCabinList.size();
01155
        // Determine the yield (equally distributed over legs).
01156
        const stdair::Yield_T& lYield = lBookingClass_ptr->getYield()/lNb
        OfLegs;
01157
        const stdair::MeanStdDevPair_T& lMeanStdDevPair =
01158
            lYieldDemandPair.second;
01159
        const stdair::MeanValue_T& lMeanValue = lMeanStdDevPair.first;
01160
        const stdair::StdDevValue_T& lStdDevValue = lMeanStdDevPair.secon
        d;
01161
        for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.
        begin();
01162
            itLC != lLegCabinList.end(); ++itLC) {
01163
            stdair::LegCabin* lLegCabin_ptr = *itLC;
01164
            assert (lLegCabin_ptr != NULL);
01165
            lLegCabin_ptr->addDemandInformation (lYield, lMeanValue, lStdDe
        vValue);
01166
        }
01167
    }
01168
}
01169
}
01170
}
01171
}
01172
}
01173
}
01174
}
01175
// //////////////////////////////////////
01176
void RMOL_Service::projectOnDDemandOnLegCabinsUsingYP(const stdair::DateTime_T&
iRMEEventTime) {
01177
01178
    if (_rmolServiceContext == NULL) {
01179
        throw stdair::NonInitialisedServiceException ("The Rmol service "
01180
            "has not been initialised");
01181
    }
01182
    assert (_rmolServiceContext != NULL);
01183
    RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
01184
01185
    // Retrieve the bom root
01186
    stdair::STDAIR_Service& lSTDAIR_Service =
01187
        lRMOL_ServiceContext.getSTDAIR_Service();
01188
    stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01189
01190
    // Retrieve the date from the RM event
01191
    const stdair::Date_T lDate = iRMEEventTime.date();
01192
01193
    const stdair::InventoryList_T lInventoryList =
01194
        stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01195
    assert (!lInventoryList.empty());
01196
    for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin();
01197
        itInv != lInventoryList.end(); ++itInv) {
01198
        const stdair::Inventory* lInventory_ptr = *itInv;
01199
        assert (lInventory_ptr != NULL);
01200
        const stdair::OnDDateList_T lOnDDateList =
01201
            stdair::BomManager::getList<stdair::OnDDate> (*lInventory_ptr);
01202
        assert (!lOnDDateList.empty());

```

```

01203         for (stdair::OnDDateList_T::const_iterator itOD = lOnDDateList.begin();
01204              itOD != lOnDDateList.end(); ++itOD) {
01205             stdair::OnDDate* lOnDDate_ptr = *itOD;
01206             assert (lOnDDate_ptr != NULL);
01207
01208             const stdair::Date_T& lDepartureDate = lOnDDate_ptr->getDate();
01209             stdair::DateOffset_T lDateOffset = lDepartureDate - lDate;
01210             stdair::DTD_T lDTD = short (lDateOffset.days());
01211
01212             stdair::DCPLIST_T::const_iterator itDCP =
01213             std::find (stdair::DEFAULT_DCP_LIST.begin(), stdair::DEFAULT_DCP_LIST.e
01214 nd(), lDTD);
01215             // Check if the forecast for this O&D date needs to be projected.
01216             if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01217                 // Browse the demand info map.
01218                 const stdair::StringDemandStructMap_T& lStringDemandStructMap =
01219                 lOnDDate_ptr->getDemandInfoMap ();
01220                 for (stdair::StringDemandStructMap_T::const_iterator itStrDS = lStringD
01221 emandStructMap.begin();
01222                      itStrDS != lStringDemandStructMap.end(); ++itStrDS) {
01223                     std::string lCabinClassPath = itStrDS->first;
01224                     const stdair::YieldDemandPair_T& lYieldDemandPair =
01225                     itStrDS->second;
01226                     const stdair::CabinClassPairList_T& lCabinClassPairList =
01227                     lOnDDate_ptr->getCabinClassPairList (lCabinClassPath);
01228                     const unsigned int lNbOfSegments = lOnDDate_ptr->getNbOfSegments();
01229                     // Sanity check
01230                     assert (lCabinClassPairList.size() == lNbOfSegments);
01231
01232                     const stdair::SegmentDateList_T lOnDSegmentDateList =
01233                     stdair::BomManager::getList<stdair::SegmentDate> (*lOnDDate_ptr);
01234                     // Sanity check
01235                     assert (lOnDSegmentDateList.size() == lNbOfSegments);
01236                     stdair::CabinClassPairList_T::const_iterator itCCP = lCabinClassPairL
01237 ist.begin();
01238                     stdair::SegmentDateList_T::const_iterator itSD = lOnDSegmentDateList.
01239 begin();
01240                     for (; itSD != lOnDSegmentDateList.end(); ++itCCP, ++itSD) {
01241                         const stdair::SegmentDate* lSegmentDate_ptr = *itSD;
01242                         assert (lSegmentDate_ptr != NULL);
01243                         const stdair::SegmentDate* lOperatingSegmentDate_ptr =
01244                         lSegmentDate_ptr->getOperatingSegmentDate ();
01245                         // Only operated legs receive the demand information.
01246                         if (lOperatingSegmentDate_ptr == NULL) {
01247                             const stdair::CabinCode_T lCabinCode = itCCP->first;
01248                             const stdair::ClassCode_T lClassCode = itCCP->second;
01249                             const stdair::SegmentCabin* lSegmentCabin_ptr =
01250                             stdair::BomManager::getObjectPtr<stdair::SegmentCabin> (*lSegme
01251 ntDate_ptr,
01252                                     lCabinC
01253 ode);
01254                             assert (lSegmentCabin_ptr != NULL);
01255                             const stdair::LegCabinList_T lLegCabinList =
01256                             stdair::BomManager::getList<stdair::LegCabin> (*lSegmentCabin_p
01257 tr);
01258                             assert (!lLegCabinList.empty());
01259                             const int lNbOfLegs = lLegCabinList.size();
01260                             // Determine the yield (equally distributed over segments and the
01261 n legs).
01262                             const stdair::MeanStdDevPair_T& lMeanStdDevPair =
01263                             lYieldDemandPair.second;

```



```

01257         const stdair::Yield_T& lYield = lYieldDemandPair.first/(lNbOfLegs
*1NbOfSegments);
01258         const stdair::MeanValue_T& lMeanValue = lMeanStdDevPair.first;
01259         const stdair::StdDevValue_T& lStdDevValue = lMeanStdDevPair.secon
d;
01260         for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.
begin();
01261             itLC != lLegCabinList.end(); ++itLC) {
01262             stdair::LegCabin* lLegCabin_ptr = *itLC;
01263             assert (lLegCabin_ptr != NULL);
01264             lLegCabin_ptr->addDemandInformation (lYield, lMeanValue, lStdDe
vValue);
01265         }
01266     }
01267 }
01268 }
01269 }
01270 }
01271 }
01272 }
01273
01274 // //////////////////////////////////////
01275 void RMOL_Service::optimiseOnD (const stdair::DateTime_T& iRMEventTime) {
01276
01277     if (_rmolServiceContext == NULL) {
01278         throw stdair::NonInitialisedServiceException ("The Rmol service "
01279                                                         "has not been initialised");
01280     }
01281     assert (_rmolServiceContext != NULL);
01282     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
01283
01284     // Retrieve the bom root
01285     stdair::STDAIR_Service& lSTDAIR_Service =
01286         lRMOL_ServiceContext.getSTDAIR_Service();
01287     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01288
01289     // Retrieve the date from the RM event
01290     const stdair::Date_T lDate = iRMEventTime.date();
01291
01292     const stdair::InventoryList_T& lInvList =
01293         stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01294     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
01295         itInv != lInvList.end(); ++itInv) {
01296         stdair::Inventory* lCurrentInv_ptr = *itInv;
01297         assert (lCurrentInv_ptr != NULL);
01298
01299         const stdair::FlightDateList_T& lFlightDateList =
01300             stdair::BomManager::getList<stdair::FlightDate> (*lCurrentInv_ptr);
01301         for (stdair::FlightDateList_T::const_iterator itFlightDate =
01302             lFlightDateList.begin();
01303             itFlightDate != lFlightDateList.end(); ++itFlightDate) {
01304             stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
01305             assert (lCurrentFlightDate_ptr != NULL);
01306
01307             const stdair::Date_T& lCurrentDepartureDate = lCurrentFlightDate_ptr->get
DepartureDate();
01308             stdair::DateOffset_T lDateOffset = lCurrentDepartureDate - lDate;
01309             stdair::DTD_T lDTD = short (lDateOffset.days());
01310
01311             stdair::DCPLList_T::const_iterator itDCP =
01312                 std::find (stdair::DEFAULT_DCP_LIST.begin(), stdair::DEFAULT_DCP_LIST.e
nd(), lDTD);

```

```

01313         // Check if the optimisation is needed.
01314         if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01315             STDAIR_LOG_DEBUG ("Optimisation using O&D forecast: " << lCurrentInv_ptr
01316                             r->getAirlineCode()
01317                             << " Departure " << lCurrentDepartureDate << " DTD "
01318                             << lDTD);
01319             Optimiser::optimiseUsingOnDForecast (*lCurrentFlightDate_ptr);
01320         }
01321     }
01322
01323     // //////////////////////////////////////
01324     void RMOL_Service::updateBidPrice (const stdair::DateTime_T& iRMEventTime) {
01325
01326         if (_rmolServiceContext == NULL) {
01327             throw stdair::NonInitialisedServiceException ("The Rmol service "
01328                                                         "has not been initialised");
01329         }
01330         assert (_rmolServiceContext != NULL);
01331         RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
01332
01333         // Retrieve the bom root
01334         stdair::STDAIR_Service& lSTDAIR_Service =
01335             lRMOL_ServiceContext.getSTDAIR_Service();
01336         stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01337
01338         // Retrieve the date from the RM event
01339         const stdair::Date_T lDate = iRMEventTime.date();
01340
01341         const stdair::InventoryList_T& lInvList =
01342             stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01343
01344         for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
01345             itInv != lInvList.end(); ++itInv) {
01346             stdair::Inventory* lCurrentInv_ptr = *itInv;
01347             assert (lCurrentInv_ptr != NULL);
01348
01349             const stdair::FlightDateList_T& lFlightDateList =
01350                 stdair::BomManager::getList<stdair::FlightDate> (*lCurrentInv_ptr);
01351             for (stdair::FlightDateList_T::const_iterator itFlightDate =
01352                 lFlightDateList.begin();
01353                 itFlightDate != lFlightDateList.end(); ++itFlightDate) {
01354                 stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
01355                 assert (lCurrentFlightDate_ptr != NULL);
01356
01357                 const stdair::Date_T& lCurrentDepartureDate = lCurrentFlightDate_ptr->get
01358                     DepartureDate();
01359                 stdair::DateOffset_T lDateOffset = lCurrentDepartureDate - lDate;
01360                 stdair::DTD_T lDTD = short (lDateOffset.days());
01361
01362                 stdair::DCPList_T::const_iterator itDCP =
01363                     std::find (stdair::DEFAULT_DCP_LIST.begin(), stdair::DEFAULT_DCP_LIST.e
01364                         nd(), lDTD);
01365                 // Check if the operation is needed.
01366                 if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01367                     updateBidPrice (*lCurrentFlightDate_ptr, lBomRoot);
01368                 }
01369             }
01370         }
01371     }

```

```

01371 // //////////////////////////////////////
01372 void RMOL_Service::updateBidPrice (const stdair::FlightDate& iFlightDate,
01373                                   stdair::BomRoot& iBomRoot) {
01374     const stdair::SegmentDateList_T& lSegmentDateList =
01375         stdair::BomManager::getList<stdair::SegmentDate> (iFlightDate);
01376     const stdair::AirlineCode_T& lOptAC = iFlightDate.getAirlineCode();
01377     const std::string lFDKeyStr = iFlightDate.describeKey();
01378
01379     for (stdair::SegmentDateList_T::const_iterator itSegmentDate = lSegmentDateLi
01380 st.begin();
01381          itSegmentDate != lSegmentDateList.end(); ++itSegmentDate) {
01382         stdair::SegmentDate* lSegmentDate_ptr = *itSegmentDate;
01383         assert (lSegmentDate_ptr != NULL);
01384         if (stdair::BomManager::hasList<stdair::SegmentDate>(*lSegmentDate_ptr)) {
01385             const stdair::LegDateList_T& lLegDateList =
01386                 stdair::BomManager::getList<stdair::LegDate>(*lSegmentDate_ptr);
01387             // Get the list of marketing carriers segments.
01388             // These are part of maketing partners inventories images held by the ope
01389             rating airline.
01390             const stdair::SegmentDateList_T& lMktSegmentDateList =
01391                 stdair::BomManager::getList<stdair::SegmentDate>(*lSegmentDate_ptr);
01392             for (stdair::SegmentDateList_T::const_iterator itMktSD = lMktSegmentDateL
01393 ist.begin();
01394                  itMktSD != lMktSegmentDateList.end(); ++itMktSD) {
01395                 // Get the marketing airline code.
01396                 stdair::SegmentDate* lMktSD_ptr = *itMktSD;
01397                 assert (lMktSD_ptr != NULL);
01398                 stdair::FlightDate* lMktFD_ptr =
01399                     stdair::BomManager::getParentPtr<stdair::FlightDate>(*lMktSD_ptr);
01400                 assert (lMktFD_ptr != NULL);
01401                 const stdair::AirlineCode_T& lMktAC = lMktFD_ptr->getAirlineCode();
01402                 // Get the (real) marketer inventory.
01403                 const stdair::Inventory* lMktInv_ptr =
01404                     stdair::BomManager::getObjectPtr<stdair::Inventory>(iBomRoot, lMktAC);
01405
01406                 assert (lMktInv_ptr != NULL);
01407                 // Get the image of the operating airline inventory held by the markete
01408                 r.
01409                 const stdair::Inventory* lOptInv_ptr =
01410                     stdair::BomManager::getObjectPtr<stdair::Inventory>(*lMktInv_ptr, lOpt
01411 AC);
01412                 assert (lOptInv_ptr != NULL);
01413                 // Find the image of the concerned flight date.
01414                 const stdair::FlightDate* lOptFD_ptr =
01415                     stdair::BomManager::getObjectPtr<stdair::FlightDate>(*lOptInv_ptr, lFD
01416 KeyStr);
01417                 assert (lOptFD_ptr != NULL);
01418                 // Browse the list of leg dates in the real operating inventory.
01419                 // Retrieve the image of each leg date.
01420                 for (stdair::LegDateList_T::const_iterator itLD = lLegDateList.begin();
01421                      itLD != lLegDateList.end(); ++itLD) {
01422                     const stdair::LegDate* lLD_ptr = *itLD;
01423                     assert (lLD_ptr != NULL);
01424                     const std::string lLDKeyStr = lLD_ptr->describeKey();
01425                     stdair::LegDate* lOptLD_ptr =
01426                         stdair::BomManager::getObjectPtr<stdair::LegDate>(*lOptFD_ptr, lLDKe
01427 yStr);
01428                     assert (lOptLD_ptr != NULL);
01429                     const stdair::LegCabinList_T& lLegCabinList_T =
01430                         stdair::BomManager::getList<stdair::LegCabin>(*lLD_ptr);
01431                     // Browse the list of leg cabins in the real operating inventory.

```

```

01424         // Retrieve the image of each leg cabin and update the bid price of t
he real and send it to the image.
01425         for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList_T.be
gin();
01426             itLC != lLegCabinList_T.end(); ++itLC) {
01427             stdair::LegCabin* lLC_ptr = *itLC;
01428             assert (lLC_ptr != NULL);
01429             const std::string lLCKeyStr = lLC_ptr->describeKey();
01430             stdair::LegCabin* lOptLC_ptr =
01431                 stdair::BomManager::getObjectPtr<stdair::LegCabin>(*lOptLD_ptr, l
LCKeyStr);
01432             assert (lOptLC_ptr != NULL);
01433             // Update the current bid price of the real leg.
01434             lLC_ptr->updateCurrentBidPrice();
01435             // Update the previous bid price (store the current).
01436             lOptLC_ptr->updatePreviousBidPrice();
01437             // Update the current bid price.
01438             lOptLC_ptr->setCurrentBidPrice (lLC_ptr->getCurrentBidPrice());
01439
01440             STDAIR_LOG_DEBUG ("Update bid price of " << lLC_ptr->getFullerKey()
<< " : " << lOptLC_ptr->getCurrentBidPrice()
01441                             << " Availability pool " << lLC_ptr->getAvailabil
ityPool());
01443         }
01444     }
01445 }
01446 }
01447 }
01448 }
01449
01450 // //////////////////////////////////////
01451 void RMOL_Service::projectOnDDemandOnLegCabinsUsingDA(const stdair::DateTime_T&
iRMEventTime) {
01452
01453     if (_rmolServiceContext == NULL) {
01454         throw stdair::NonInitialisedServiceException ("The Rmol service "
01455                                                         "has not been initialised");
01456     }
01457     assert (_rmolServiceContext != NULL);
01458     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
01459
01460     // Retrieve the bom root
01461     stdair::STDAIR_Service& lSTDAIR_Service =
01462         lRMOL_ServiceContext.getSTDAIR_Service();
01463     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01464
01465     // Retrieve the date from the RM event
01466     const stdair::Date_T lDate = iRMEventTime.date();
01467
01468     const stdair::InventoryList_T lInventoryList =
01469         stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01470     assert (!lInventoryList.empty());
01471     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin();
01472         itInv != lInventoryList.end(); ++itInv) {
01473         const stdair::Inventory* lInventory_ptr = *itInv;
01474         assert (lInventory_ptr != NULL);
01475         const stdair::OnDDateList_T lOnDDateList =
01476             stdair::BomManager::getList<stdair::OnDDate> (*lInventory_ptr);
01477         assert (!lOnDDateList.empty());
01478         for (stdair::OnDDateList_T::const_iterator itOD = lOnDDateList.begin();
01479             itOD != lOnDDateList.end(); ++itOD) {

```

```

01480         stdair::OnDDate* lOnDDate_ptr = *itOD;
01481         assert (lOnDDate_ptr != NULL);
01482
01483         const stdair::Date_T& lDepartureDate = lOnDDate_ptr->getDate();
01484         stdair::DateOffset_T lDateOffset = lDepartureDate - lDate;
01485         stdair::DTD_T lDTD = short (lDateOffset.days());
01486
01487         stdair::DCPLIST_T::const_iterator itDCP =
01488             std::find (stdair::DEFAULT_DCP_LIST.begin(), stdair::DEFAULT_DCP_LIST.e
nd(), lDTD);
01489         // Check if the forecast for this O&D date needs to be projected.
01490         if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01491
01492             // Browse the demand info map.
01493             const stdair::StringDemandStructMap_T& lStringDemandStructMap =
01494                 lOnDDate_ptr->getDemandInfoMap ();
01495             for (stdair::StringDemandStructMap_T::const_iterator itStrDS = lStringD
emandStructMap.begin();
01496                 itStrDS != lStringDemandStructMap.end(); ++itStrDS) {
01497                 std::string lCabinClassPath = itStrDS->first;
01498                 const stdair::YieldDemandPair_T& lYieldDemandPair = itStrDS->second;
01499                 const stdair::CabinClassPairList_T& lCabinClassPairList =
01500                     lOnDDate_ptr->getCabinClassPairList (lCabinClassPath);
01501                 const unsigned int lNbOfSegments = lOnDDate_ptr->getNbOfSegments();
01502                 // Sanity check
01503                 assert (lCabinClassPairList.size() == lNbOfSegments);
01504
01505                 //
01506                 const stdair::SegmentDateList_T lOnDSegmentDateList =
01507                     stdair::BomManager::getList<stdair::SegmentDate> (*lOnDDate_ptr);
01508                 // Sanity check
01509                 assert (lOnDSegmentDateList.size() == lNbOfSegments);
01510                 stdair::CabinClassPairList_T::const_iterator itCCP = lCabinClassPairL
ist.begin();
01511                 stdair::SegmentDateList_T::const_iterator itSD = lOnDSegmentDateList.
begin();
01512                 // List of bid prices that will be used to easily compute displacemen
t-adjusted yields.
01513                 std::list<stdair::BidPrice_T> lBidPriceList;
01514                 // The sum of bid prices that will be stored in the list above.
01515                 stdair::BidPrice_T lTotalBidPrice = 0;
01516                 // Retrieve the bid prices
01517                 for (; itSD != lOnDSegmentDateList.end(); ++itCCP, ++itSD) {
01518                     // Get the operating segment cabin (it holds the bid price informat
ion).
01519                     const stdair::SegmentDate* lSegmentDate_ptr = *itSD;
01520                     assert (lSegmentDate_ptr != NULL);
01521                     // Get the operating airline code and check if it is the airline we
are looking for.
01522                     const stdair::SegmentDate* lOperatingSegmentDate_ptr =
01523                         lSegmentDate_ptr->getOperatingSegmentDate ();
01524                     if (lOperatingSegmentDate_ptr != NULL) {
01525                         lSegmentDate_ptr = lOperatingSegmentDate_ptr;
01526                     }
01527                     const stdair::CabinCode_T lCabinCode = itCCP->first;
01528                     const stdair::SegmentCabin* lSegmentCabin_ptr =
01529                         stdair::BomManager::getObjectPtr<stdair::SegmentCabin> (*lSegment
Date_ptr,
01530                                     lCabinCod
e);
01531                     assert (lSegmentCabin_ptr != NULL);
01532                     stdair::BidPrice_T lBidPrice = 0;

```

```

01533         const stdair::LegCabinList_T lLegCabinList =
01534             stdair::BomManager::getList<stdair::LegCabin>(*lSegmentCabin_ptr)
01535         ;
01536         for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
01537             itLC != lLegCabinList.end(); ++itLC) {
01538             const stdair::LegCabin* lLegCabin_ptr = *itLC;
01539             assert (lLegCabin_ptr != NULL);
01540             lBidPrice += lLegCabin_ptr->getCurrentBidPrice();
01541         }
01542         lBidPriceList.push_back (lBidPrice);
01543         lTotalBidPrice += lBidPrice;
01544     }
01545
01546     itCCP = lCabinClassPairList.begin();
01547     itSD = lOnDSegmentDateList.begin();
01548     std::list<stdair::BidPrice_T>::const_iterator itBP = lBidPriceList.begin();
01549     for (; itSD != lOnDSegmentDateList.end(); ++itCCP, ++itSD, ++itBP) {
01550         stdair::BidPrice_T lBidPrice = *itBP;
01551         stdair::BidPrice_T lComplementaryBidPrice = lTotalBidPrice - lBidPrice;
01552
01553         const stdair::SegmentDate* lSegmentDate_ptr = *itSD;
01554         assert (lSegmentDate_ptr != NULL);
01555         const stdair::SegmentDate* lOperatingSegmentDate_ptr =
01556             lSegmentDate_ptr->getOperatingSegmentDate ();
01557         // Only operated legs receive the demand information.
01558         if (lOperatingSegmentDate_ptr == NULL) {
01559             const stdair::CabinCode_T lCabinCode = itCCP->first;
01560             const stdair::ClassCode_T lClassCode = itCCP->second;
01561             const stdair::SegmentCabin* lSegmentCabin_ptr =
01562                 stdair::BomManager::getObjectPtr<stdair::SegmentCabin> (*lSegmentDate_ptr,
01563                                     lCabinCode);
01564             assert (lSegmentCabin_ptr != NULL);
01565             const stdair::LegCabinList_T lLegCabinList =
01566                 stdair::BomManager::getList<stdair::LegCabin> (*lSegmentCabin_ptr);
01567             assert (!lLegCabinList.empty());
01568             // Determine the displacement-adjusted yield.
01569             // It is set to 100 (positive small value), if the computed value
01570             // is negative.
01571             const stdair::Yield_T& lDAYield =
01572                 std::max(100., lYieldDemandPair.first - lComplementaryBidPrice);
01573
01574             stdair::Yield_T lYield = lDAYield;
01575             // In order to be protected against important variations of partners' bid price,
01576             // the displacement adjusted yield is not allowed to get out of a certain range.
01577             // This range is here chosen to be from 80% to 100% of the (static rule) prorated yield.
01578             /*
01579             const int lNbOfLegs = lLegCabinList.size();
01580             const stdair::Yield_T& lStaticProrationYield =
01581                 lDemandStruct.getYield() / (lNbOfLegs * lNbOfSegments);
01582             if (lDAYield < 0.8 * lStaticProrationYield) {
01583                 lYield = 0.8 * lStaticProrationYield;
01584             }
01585             */

```

```

01583     }
01584     if (lDAYield > lStaticProrationYield) {
01585         lYield = lStaticProrationYield;
01586     }
01587     */
01588     const stdair::MeanStdDevPair_T& lMeanStdDevPair =
01589         lYieldDemandPair.second;
01590     const stdair::MeanValue_T& lMeanValue = lMeanStdDevPair.first;
01591     const stdair::StdDevValue_T& lStdDevValue = lMeanStdDevPair.secon
d;
01592     for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.
begin();
01593         itLC != lLegCabinList.end(); ++itLC) {
01594         stdair::LegCabin* lLegCabin_ptr = *itLC;
01595         assert (lLegCabin_ptr != NULL);
01596         lLegCabin_ptr->addDemandInformation (lYield, lMeanValue, lStdDe
vValue);
01597     }
01598 }
01599 }
01600 }
01601 }
01602 }
01603 }
01604 }
01605
01606 // //////////////////////////////////////
01607 void RMOL_Service::projectOnDDemandOnLegCabinsUsingDYP(const stdair::DateTime_T
& iRMEEventTime) {
01608
01609     if (_rmolServiceContext == NULL) {
01610         throw stdair::NonInitialisedServiceException ("The Rmol service "
"has not been initialised");
01611     }
01612     assert (_rmolServiceContext != NULL);
01613     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
01614
01615     // Retrieve the bom root
01616     stdair::STDAIR_Service& lSTDAIR_Service =
lRMOL_ServiceContext.getSTDAIR_Service();
01617     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01618
01619     const stdair::InventoryList_T lInventoryList =
stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01620     assert (!lInventoryList.empty());
01621     for (stdair::InventoryList_T::const_iterator itInv = lInventoryList.begin();
itInv != lInventoryList.end(); ++itInv) {
01622         const stdair::Inventory* lInventory_ptr = *itInv;
01623         assert (lInventory_ptr != NULL);
01624         projectOnDDemandOnLegCabinsUsingDYP (iRMEEventTime, *lInventory_ptr);
01625     }
01626 }
01627
01628 // //////////////////////////////////////
01629 void RMOL_Service::projectOnDDemandOnLegCabinsUsingDYP(const stdair::DateTime_T
& iRMEEventTime,
01630 const stdair::Inventory&
iInventory) {
01631
01632     const stdair::OnDDateList_T lOnDDateList =
stdair::BomManager::getList<stdair::OnDDate> (iInventory);
01633     assert (!lOnDDateList.empty());

```

```

01639     for (stdair::OnDDateList_T::const_iterator itOD = lOnDDateList.begin();
01640          itOD != lOnDDateList.end(); ++itOD) {
01641         stdair::OnDDate* lOnDDate_ptr = *itOD;
01642         assert (lOnDDate_ptr != NULL);
01643
01644         // Retrieve the date from the RM event
01645         const stdair::Date_T lDate = iRMEventTime.date();
01646
01647         const stdair::Date_T& lDepartureDate = lOnDDate_ptr->getDate();
01648         stdair::DateOffset_T lDateOffset = lDepartureDate - lDate;
01649         stdair::DTD_T lDTD = short (lDateOffset.days());
01650
01651         stdair::DCPLIST_T::const_iterator itDCP =
01652             std::find (stdair::DEFAULT_DCP_LIST.begin(), stdair::DEFAULT_DCP_LIST.end
01653             ), lDTD);
01654         // Check if the forecast for this O&D date needs to be projected.
01655         if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01656             // Browse the demand info map.
01657             const stdair::StringDemandStructMap_T& lStringDemandStructMap =
01658                 lOnDDate_ptr->getDemandInfoMap ();
01659             for (stdair::StringDemandStructMap_T::const_iterator itStrDS = lStringDem
01660                 andStructMap.begin();
01661                  itStrDS != lStringDemandStructMap.end(); ++itStrDS) {
01662                 std::string lCabinClassPath = itStrDS->first;
01663                 const stdair::YieldDemandPair_T& lYieldDemandPair = itStrDS->second;
01664                 const stdair::CabinClassPairList_T& lCabinClassPairList =
01665                     lOnDDate_ptr->getCabinClassPairList (lCabinClassPath);
01666                 const unsigned int lNbOfSegments = lOnDDate_ptr->getNbOfSegments();
01667                 // Sanity check
01668                 assert (lCabinClassPairList.size() == lNbOfSegments);
01669
01670                 //
01671                 const stdair::SegmentDateList_T lOnDSegmentDateList =
01672                     stdair::BomManager::getList<stdair::SegmentDate> (*lOnDDate_ptr);
01673                 // Sanity check
01674                 assert (lOnDSegmentDateList.size() == lNbOfSegments);
01675                 stdair::CabinClassPairList_T::const_iterator itCCP = lCabinClassPairLis
01676                 t.begin();
01677                 stdair::SegmentDateList_T::const_iterator itSD = lOnDSegmentDateList.be
01678                 gin();
01679                 // The sum of bid prices of all cabins.
01680                 stdair::BidPrice_T lTotalBidPrice = 0;
01681                 for (; itSD != lOnDSegmentDateList.end(); ++itCCP, ++itSD) {
01682                     // Get the operating segment cabin (it holds the bid price informatio
01683                     n).
01684                     const stdair::SegmentDate* lSegmentDate_ptr = *itSD;
01685                     assert (lSegmentDate_ptr != NULL);
01686                     // Get the operating airline code and check if it is the airline we a
01687                     re looking for.
01688                     const stdair::SegmentDate* lOperatingSegmentDate_ptr =
01689                         lSegmentDate_ptr->getOperatingSegmentDate ();
01690                     if (lOperatingSegmentDate_ptr != NULL) {
01691                         lSegmentDate_ptr = lOperatingSegmentDate_ptr;
01692                     }
01693                     const stdair::CabinCode_T lCabinCode = itCCP->first;
01694                     const stdair::SegmentCabin* lSegmentCabin_ptr =
01695                         stdair::BomManager::getObjectPtr<stdair::SegmentCabin> (*lSegmentDa
01696                         te_ptr,
01697                                     lCabinCode)
01698                     ;
01699                     assert (lSegmentCabin_ptr != NULL);

```



```

01693         const stdair::LegCabinList_T lLegCabinList =
01694             stdair::BomManager::getList<stdair::LegCabin>(*lSegmentCabin_ptr);
01695         for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.begin()
n();
01696             itLC != lLegCabinList.end(); ++itLC) {
01697             const stdair::LegCabin* lLegCabin_ptr = *itLC;
01698             assert (lLegCabin_ptr != NULL);
01699             lTotalBidPrice += lLegCabin_ptr->getCurrentBidPrice();
01700         }
01701     }
01702
01703
01704     itCCP = lCabinClassPairList.begin();
01705     itSD = lOnDSegmentDateList.begin();
01706     for (; itSD != lOnDSegmentDateList.end(); ++itCCP, ++itSD) {
01707         const stdair::SegmentDate* lSegmentDate_ptr = *itSD;
01708         assert (lSegmentDate_ptr != NULL);
01709         const stdair::SegmentDate* lOperatingSegmentDate_ptr =
01710             lSegmentDate_ptr->getOperatingSegmentDate ();
01711         // Only operated legs receive the demand information.
01712         if (lOperatingSegmentDate_ptr == NULL) {
01713             const stdair::CabinCode_T lCabinCode = itCCP->first;
01714             const stdair::ClassCode_T lClassCode = itCCP->second;
01715             const stdair::SegmentCabin* lSegmentCabin_ptr =
01716                 stdair::BomManager::getObjectPtr<stdair::SegmentCabin> (*lSegment
Date_ptr,
01717
lCabinCod
e);
01718             assert (lSegmentCabin_ptr != NULL);
01719             const stdair::LegCabinList_T lLegCabinList =
01720                 stdair::BomManager::getList<stdair::LegCabin> (*lSegmentCabin_ptr
);
01721             assert (!lLegCabinList.empty());
01722             const stdair::Yield_T& lYield = lYieldDemandPair.first;

01723             const stdair::MeanStdDevPair_T& lMeanStdDevPair =
01724                 lYieldDemandPair.second;
01725             const stdair::MeanValue_T& lMeanValue = lMeanStdDevPair.first;
01726             const stdair::StdDevValue_T& lStdDevValue = lMeanStdDevPair.second;

01727             for (stdair::LegCabinList_T::const_iterator itLC = lLegCabinList.be
gin();
01728                 itLC != lLegCabinList.end(); ++itLC) {
01729                 stdair::LegCabin* lLegCabin_ptr = *itLC;
01730                 assert (lLegCabin_ptr != NULL);
01731                 const stdair::BidPrice_T& lBidPrice = lLegCabin_ptr->getCurrentBi
dPrice();
01732                 const stdair::RealNumber_T lDynamicYieldProrationFactor = lBidPri
ce / lTotalBidPrice;
01733                 const stdair::Yield_T lProratedYield = lDynamicYieldProrationFact
or*lYield;
01734                 lLegCabin_ptr->addDemandInformation (lProratedYield, lMeanValue,
lStdDevValue);
01735
01736                 // STDAIR_LOG_DEBUG ("Adding demand information to leg-cabin " <
< lLegCabin_ptr->getFullerKey()
01737                 //
<< " Total yield " << lYield << " Proration
factor "
01738                 //
<< lDynamicYieldProrationFactor << " Prorate
d yield " << lProratedYield
01739                 //
<< " Mean demand " << lMeanValue << " StdDev
" << lStdDevValue);

```

```

01740         }
01741     }
01742 }
01743 }
01744 }
01745 }
01746 }
01747
01748 // //////////////////////////////////////
01749 void RMOL_Service::optimiseOnDUsingRMCooperation (const stdair::DateTime_T& iRM
EventTime) {
01750
01751     if (_rmolServiceContext == NULL) {
01752         throw stdair::NonInitialisedServiceException ("The Rmol service "
01753             "has not been initialised");
01754     }
01755     assert (_rmolServiceContext != NULL);
01756     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
01757
01758     // Retrieve the bom root
01759     stdair::STDAIR_Service& lSTDAIR_Service =
01760         lRMOL_ServiceContext.getSTDAIR_Service();
01761     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01762
01763     // Retrieve the date from the RM event
01764     const stdair::Date_T lDate = iRMEventTime.date();
01765
01766     // Browse the list of inventories and optimise within each one independently.
01767
01768     const stdair::InventoryList_T& lInvList =
01769         stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01770     for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
01771         itInv != lInvList.end(); ++itInv) {
01772         stdair::Inventory* lCurrentInv_ptr = *itInv;
01773         assert (lCurrentInv_ptr != NULL);
01774
01775         double lMaxBPVariation = 1.0;
01776         short lIterationCounter = 0;
01777         // Iterate until the variation is under the wanted level or the maximal num
ber of iterations is reached.
01778         while (lMaxBPVariation > 0.01 && lIterationCounter < 10) {
01779             lMaxBPVariation = 0.0;
01780             lIterationCounter++;
01781             const stdair::FlightDateList_T& lFlightDateList =
01782                 stdair::BomManager::getList<stdair::FlightDate> (*lCurrentInv_ptr);
01783             for (stdair::FlightDateList_T::const_iterator itFlightDate =
01784                 lFlightDateList.begin();
01785                 itFlightDate != lFlightDateList.end(); ++itFlightDate) {
01786                 stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
01787                 assert (lCurrentFlightDate_ptr != NULL);
01788                 const stdair::Date_T& lCurrentDepartureDate = lCurrentFlightDate_ptr->g
etDepartureDate();
01789                 stdair::DateOffset_T lDateOffset = lCurrentDepartureDate - lDate;
01790                 stdair::DTD_T lDTD = short (lDateOffset.days());
01791
01792                 stdair::DCPLIST_T::const_iterator itDCP =
01793                     std::find (stdair::DEFAULT_DCP_LIST.begin(),
01794                         stdair::DEFAULT_DCP_LIST.end(), lDTD);
01795                 // Check if the optimisation is needed.
01796                 if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01797                     const double lBPVariation = Optimiser::optimiseUsingOnDForecast (*lCu

```

```

    rrentFlightDate_ptr);
01797         lMaxBPVariation = std::max(lMaxBPVariation, lBPVariation);
01798     }
01799 }
01800 // Update the prorated yields for the current inventory.
01801 resetDemandInformation (iRMEventTime, *lCurrentInv_ptr);
01802 projectOnDDemandOnLegCabinsUsingDYP (iRMEventTime, *lCurrentInv_ptr);
01803 }
01804 }
01805 }
01806
01807
01808 // //////////////////////////////////////
01809 void RMOL_Service::optimiseOnDUsingAdvancedRMCooperation (const stdair::DateTim
e_T& iRMEventTime) {
01810
01811     if (_rmolServiceContext == NULL) {
01812         throw stdair::NonInitialisedServiceException ("The Rmol service "
01813                                                         "has not been initialised");
01814     }
01815     assert (_rmolServiceContext != NULL);
01816     RMOL_ServiceContext& lRMOL_ServiceContext = *_rmolServiceContext;
01817
01818     // Retrieve the bom root
01819     stdair::STDAIR_Service& lSTDAIR_Service =
01820         lRMOL_ServiceContext.getSTDAIR_Service();
01821     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
01822
01823     // Retrieve the date from the RM event
01824     const stdair::Date_T lDate = iRMEventTime.date();
01825
01826     double lMaxBPVariation = 1.0;
01827     short lIterationCounter = 0;
01828     // Iterate until the variation is under the wanted level or the maximal numbe
r of iterations is reached.
01829     // Every iteration corresponds to the optimisation of the whole network. Bid
prices are communicated
01830     // between partners at the end of each iteration.
01831     while (lMaxBPVariation > 0.01 && lIterationCounter < 50) {
01832         lMaxBPVariation = 0.0;
01833         lIterationCounter++;
01834
01835         const stdair::InventoryList_T& lInvList =
01836             stdair::BomManager::getList<stdair::Inventory> (lBomRoot);
01837         for (stdair::InventoryList_T::const_iterator itInv = lInvList.begin();
01838             itInv != lInvList.end(); ++itInv) {
01839             stdair::Inventory* lCurrentInv_ptr = *itInv;
01840             assert (lCurrentInv_ptr != NULL);
01841             const stdair::FlightDateList_T& lFlightDateList =
01842                 stdair::BomManager::getList<stdair::FlightDate> (*lCurrentInv_ptr);
01843             for (stdair::FlightDateList_T::const_iterator itFlightDate =
01844                 lFlightDateList.begin();
01845                 itFlightDate != lFlightDateList.end(); ++itFlightDate) {
01846                 stdair::FlightDate* lCurrentFlightDate_ptr = *itFlightDate;
01847                 assert (lCurrentFlightDate_ptr != NULL);
01848
01849                 const stdair::Date_T& lCurrentDepartureDate = lCurrentFlightDate_ptr->g
etDepartureDate();
01850                 stdair::DateOffset_T lDateOffset = lCurrentDepartureDate - lDate;
01851                 stdair::DTD_T lDTD = short (lDateOffset.days());
01852
01853                 stdair::DCPList_T::const_iterator itDCP =

```

```

01854         std::find (stdair::DEFAULT_DCP_LIST.begin(),
stdair::DEFAULT_DCP_LIST.end(), lDTD);
01855         if (itDCP != stdair::DEFAULT_DCP_LIST.end()) {
01856             const double lBPVariation = Optimiser::optimiseUsingOnDForecast (*lCu
rrentFlightDate_ptr);
01857             lMaxBPVariation = std::max(lMaxBPVariation, lBPVariation);
01858         }
01859     }
01860 }
01861 // At the end of each iteration, communicate bid prices and compute displac
ement adjusted yields.
01862 updateBidPrice (iRMEventTime);
01863 resetDemandInformation (iRMEventTime);
01864 projectOnDDemandOnLegCabinsUsingDYP (iRMEventTime);
01865 }
01866 }
01867
01868 }

```

43.142 rmol/service/RMOL_ServiceContext.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <stdair/STDAIR_Service.hpp>
#include <rmol/basic/BasConst_RMOL_Service.hpp>
#include <rmol/service/RMOL_ServiceContext.hpp>

```

Namespaces

- namespace [RMOL](#)

43.143 RMOL_ServiceContext.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/STDAIR_Service.hpp>
00009 // RMOL
00010 #include <rmol/basic/BasConst_RMOL_Service.hpp>
00011 #include <rmol/service/RMOL_ServiceContext.hpp>
00012
00013 namespace RMOL {
00014
00015 // //////////////////////////////////////
00016 RMOL_ServiceContext::RMOL_ServiceContext() : _ownStdairService (false) {
00017 }
00018
00019 // //////////////////////////////////////
00020 RMOL_ServiceContext::RMOL_ServiceContext (const RMOL_ServiceContext&) {

```

```

00021     assert (false);
00022 }
00023
00024 // //////////////////////////////////////
00025 RMOL_ServiceContext::~RMOL_ServiceContext() {
00026 }
00027
00028 // //////////////////////////////////////
00029 stdair::STDAIR_Service& RMOL_ServiceContext::getSTDAIR_Service() const {
00030     assert (_stdairService != NULL);
00031     return *_stdairService;
00032 }
00033
00034 // //////////////////////////////////////
00035 const std::string RMOL_ServiceContext::shortDisplay() const {
00036     std::ostringstream ostr;
00037     ostr << "RMOL_ServiceContext -- Owns StdAir service: " << _ownStdairService;
00038     return ostr.str();
00039 }
00040
00041 // //////////////////////////////////////
00042 const std::string RMOL_ServiceContext::display() const {
00043     std::ostringstream ostr;
00044     ostr << shortDisplay();
00045     return ostr.str();
00046 }
00047
00048 // //////////////////////////////////////
00049 const std::string RMOL_ServiceContext::describe() const {
00050     return shortDisplay();
00051 }
00052
00053 // //////////////////////////////////////
00054 void RMOL_ServiceContext::reset() {
00055     if (_ownStdairService == true) {
00056         _stdairService.reset();
00057     }
00058 }
00059
00060 }

```

43.144 rmol/service/RMOL_ServiceContext.hpp File Reference

```

#include <string>

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <rmol/RMOL_Types.hpp>

```

Classes

- class [RMOL::RMOL_ServiceContext](#)
Inner class holding the context for the [RMOL](#) Service object.

Namespaces

- namespace [stdair](#)
- namespace [RMOL](#)

43.145 RMOL_ServiceContext.hpp

```

00001 #ifndef __RMOL_SVC_RMOL_SERVICE_CONTEXT_HPP
00002 #define __RMOL_SVC_RMOL_SERVICE_CONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_maths_types.hpp>
00013 #include <stdair/stdair_service_types.hpp>
00014 #include <stdair/service/ServiceAbstract.hpp>
00015 // RMOL
00016 #include <rmol/RMOL_Types.hpp>
00017
00018 namespace stdair {
00019     class STDAIR_Service;
00020     class LegCabin;
00021 }
00022
00023 namespace RMOL {
00024
00025     class RMOL_ServiceContext : public stdair::ServiceAbstract {
00026     friend class RMOL_Service;
00027     friend class FacRmolServiceContext;
00028
00029     private:
00030         // ////////// Getters //////////
00031         stdair::STDAIR_ServicePtr_T getSTDAIR_ServicePtr() const {
00032             return _stdairService;
00033         }
00034
00035         stdair::STDAIR_Service& getSTDAIR_Service() const;
00036
00037         const bool getOwnStdairServiceFlag() const {
00038             return _ownStdairService;
00039         }
00040
00041     private:
00042         // ////////// Setters //////////
00043         void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00044                                 const bool iOwnStdairService) {

```

```

00067     _stdairService = ioSTDAIR_ServicePtr;
00068     _ownStdairService = iOwnStdairService;
00069 }
00070
00074 void reset();
00075
00076
00077 private:
00078     // ////////// Display Methods //////////
00082     const std::string shortDisplay() const;
00083
00087     const std::string display() const;
00088
00092     const std::string describe() const;
00093
00094
00095 private:
00096     // ////////// Construction / initialisation //////////
00100     RMOL_ServiceContext();
00104     RMOL_ServiceContext (const RMOL_ServiceContext&);
00105
00109     ~RMOL_ServiceContext();
00110
00111
00112 private:
00113     // ////////////////////////////////// Children //////////////////////////////////
00117     stdair::STDAIR_ServicePtr_T _stdairService;
00118
00122     bool _ownStdairService;
00123 };
00124
00125 }
00126 #endif // __RMOL_SVC_RMOL_SERVICE_CONTEXT_HPP

```

43.146 test/rmol/bomsforforecaster.cpp File Reference

43.147 bomsforforecaster.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <limits>
00011 #include <sstream>
00012 #include <fstream>
00013 #include <string>
00014 // Boost Unit Test Framework (UTF)
00015 #define BOOST_TEST_DYN_LINK
00016 #define BOOST_TEST_MAIN
00017 #define BOOST_TEST_MODULE OptimiseTestSuite
00018 #include <boost/test/unit_test.hpp>
00019 // StdAir
00020 #include <stdair/basic/BasLogParams.hpp>
00021 #include <stdair/basic/BasDBParams.hpp>
00022 #include <stdair/service/Logger.hpp>
00023 // RMOL
00024 #include <rmol/RMOL_Service.hpp>
00025 #include <rmol/config/rmol-paths.hpp>

```

```

00026
00027 namespace boost_utf = boost::unit_test;
00028
00029 // (Boost) Unit Test XML Report
00030 std::ofstream utfReportStream ("bomsforforecaster_utfresults.xml");
00031
00032 struct UnitTestConfig {
00033     UnitTestConfig() {
00034         boost_utf::unit_test_log.set_stream (utfReportStream);
00035         boost_utf::unit_test_log.set_format (boost_utf::XML);
00036         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00037         //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tes
00038     ts);
00039     }
00040
00041     ~UnitTestConfig() {
00042     }
00043 };
00044
00045 namespace RMOL {
00046
00047     struct BookingClassData {
00048
00049         // Attributes
00050         double _bookingCount;
00051         double _fare;
00052         double _sellupFactor;
00053         bool _censorshipFlag;
00054
00055         // Constructor
00056         BookingClassData (const double iBookingCount, const double iFare,
00057                         const double iSellupFactor, const bool iCensorshipFlag)
00058             : _bookingCount(iBookingCount), _fare(iFare),
00059             _sellupFactor(iSellupFactor), _censorshipFlag(iCensorshipFlag) {
00060         }
00061
00062         // Getters
00063         double getFare () const {
00064             return _fare;
00065         }
00066
00067         bool getCensorshipFlag () const {
00068             return _censorshipFlag;
00069         }
00070
00071         // Display
00072         std::string toString() const {
00073             std::ostringstream ostr;
00074             ostr << std::endl
00075                 << "[Booking class data information]" << std::endl
00076                 << "Booking counter: " << _bookingCount << std::endl
00077                 << "Fare: " << _fare << std::endl
00078                 << "Sell-up Factor: " << _sellupFactor << std::endl
00079                 << "censorshipFlag: " << _censorshipFlag << std::endl;
00080             return ostr.str();
00081         }
00082     };
00083
00084     struct BookingClassDataSet {
00085
00086         typedef std::vector<BookingClassData*> BookingClassDataList_T;

```



```

00094
00095     // Attributes
00096     int _numberOfClass;
00097     double _minimumFare;
00098     bool _censorshipFlag; // true if any of the classes is censored
00099     BookingClassDataList_T _bookingClassDataList;
00100
00101     // Constructor
00102     BookingClassDataSet ()
00103     : _numberOfClass(0), _minimumFare(0),
00104       _censorshipFlag(false) {
00105     }
00106
00107     // Add BookingClassData
00108     void addBookingClassData (BookingClassData& ioBookingClassData) {
00109         _bookingClassDataList.push_back (&ioBookingClassData);
00110     }
00111
00112     // Getters
00113     unsigned int getNumberOfClass () const {
00114         return _bookingClassDataList.size();
00115     }
00116
00117     double getMinimumFare () const {
00118         return _minimumFare;
00119     }
00120
00121     bool getCensorshipFlag () const {
00122         return _censorshipFlag;
00123     }
00124
00125     // Setters
00126     void setMinimumFare (const double iMinFare) {
00127         _minimumFare = iMinFare;
00128     }
00129
00130     void setCensorshipFlag (const bool iCensorshipFlag) {
00131         _censorshipFlag = iCensorshipFlag;
00132     }
00133
00134     // compute minimum fare
00135     void updateMinimumFare() {
00136         double minFare = std::numeric_limits<double>::max();
00137         BookingClassDataList_T::iterator itBookingClassDataList;
00138         for (itBookingClassDataList = _bookingClassDataList.begin();
00139             itBookingClassDataList != _bookingClassDataList.end();
00140             ++itBookingClassDataList) {
00141             BookingClassData* lBookingClassData = *itBookingClassDataList;
00142             assert (lBookingClassData != NULL);
00143
00144             const double lFare = lBookingClassData->getFare();
00145             if (lFare < minFare) {
00146                 minFare = lFare;
00147             }
00148         }
00149         //
00150         setMinimumFare(minFare);
00151     }
00152
00153     // compute censorship flag for the data set
00154     void updateCensorshipFlag () {
00155         bool censorshipFlag = false;

```

```

00156     BookingClassDataList_T::iterator itBookingClassDataList;
00157     for (itBookingClassDataList = _bookingClassDataList.begin();
00158         itBookingClassDataList != _bookingClassDataList.end();
00159         ++itBookingClassDataList) {
00160         BookingClassData* lBookingClassData = *itBookingClassDataList;
00161         assert (lBookingClassData != NULL);
00162
00163         const bool lCensorshipFlagOfAClass =
00164             lBookingClassData->getCensorshipFlag();
00165         if (lCensorshipFlagOfAClass) {
00166             censorshipFlag = true;
00167             break;
00168         }
00169     }
00170     //
00171     setCensorshipFlag(censorshipFlag);
00172 }
00173
00174 // Display
00175 std::string toString() const {
00176     std::ostringstream oStr;
00177     oStr << std::endl
00178         << "[Booking class data set information]" << std::endl
00179         << "Number of classes: " << _numberOfClass << std::endl
00180         << "Minimum fare: " << _minimumFare << std::endl
00181         << "The data of the class set are sensed: " << _censorshipFlag
00182         << std::endl;
00183     return oStr.str();
00184 }
00185
00186 };
00187
00188 // /**----- BOM : Q-Forecaster ----- */
00189 // struct QForecaster {
00190
00191 //     // Function focused BOM
00192
00193 //     // 1. calculate sell up probability for Q-eq
00194
00195 //     // 2. calculate Q-Equivalent Booking
00196 //     double calculateQEqBooking (BookingClassDataSet& iBookingClassDataSet) {
00197 //         double lQEqBooking = 0.0;
00198 //         double lMinFare = iBookingClassDataSet.getMinimumFare();
00199
00200
00201 //         return lQEqBooking;
00202 //     }
00203
00204 //     /* Calculate Q-equivalent demand
00205 //         [<- performed by unconstrainer if necessary (Using ExpMax BOM)]
00206 //     */
00207
00208
00209 //     // 3. Partition to each class
00210
00211 //     //
00212
00213 // };
00214
00215 }
00216
00217 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////

```

```

00218
00219 // Set the UTF configuration (re-direct the output to a specific file)
00220 BOOST_GLOBAL_FIXTURE (UnitTestFixture);
00221
00222 BOOST_AUTO_TEST_SUITE (master_test_suite)
00223
00224 BOOST_AUTO_TEST_CASE (rmol_forecaster) {
00225
00226     // Output log File
00227     std::string lLogFilename ("bomsforforecaster.log");
00228     std::ofstream logOutputFile;
00229
00230     // Open and clean the log outputfile
00231     logOutputFile.open (lLogFilename.c_str());
00232     logOutputFile.clear();
00233
00234     // Initialise the RMOL service
00235     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00236
00237     // Initialise the RMOL service
00238     RMOL::RMOL_Service rmolService (lLogParams);
00239
00240     // Build a sample BOM tree
00241     rmolService.buildSampleBom();
00242
00243     // Register BCDDataSet
00244     RMOL::BookingClassDataSet lBookingClassDataSet;
00245
00246     // Register BookingClassData
00247     RMOL::BookingClassData QClassData (10, 100, 1, false);
00248     RMOL::BookingClassData MClassData (5, 150, 0.8, true);
00249     RMOL::BookingClassData BClassData (0, 200, 0.6, false);
00250     RMOL::BookingClassData YClassData (0, 300, 0.3, false);
00251
00252     // Display
00253     STDAIR_LOG_DEBUG (QClassData.toString());
00254     STDAIR_LOG_DEBUG (MClassData.toString());
00255     STDAIR_LOG_DEBUG (BClassData.toString());
00256     STDAIR_LOG_DEBUG (YClassData.toString());
00257
00258     // Add BookingClassData into the BCDDataSet
00259     lBookingClassDataSet.addBookingClassData (QClassData);
00260     lBookingClassDataSet.addBookingClassData (MClassData);
00261     lBookingClassDataSet.addBookingClassData (BClassData);
00262     lBookingClassDataSet.addBookingClassData (YClassData);
00263
00264     // DEBUG
00265     STDAIR_LOG_DEBUG (lBookingClassDataSet.toString());
00266
00267     // Number of classes
00268     const unsigned int lNoOfClass = lBookingClassDataSet.getNumberOfClass();
00269
00270     // DEBUG
00271     STDAIR_LOG_DEBUG ("Number of Classes: " << lNoOfClass);
00272
00273     // Minimum fare
00274     BOOST_CHECK_NO_THROW (lBookingClassDataSet.updateMinimumFare());
00275     const double lMinFare = lBookingClassDataSet.getMinimumFare();
00276
00277     // DEBUG
00278     STDAIR_LOG_DEBUG ("Minimum fare: " << lMinFare);

```

```

00285
00286 // Censorship flag
00287 BOOST_CHECK_NO_THROW (lBookingClassDataSet.updateCensorshipFlag());
00288 const bool lCensorshipFlag = lBookingClassDataSet.getCensorshipFlag();
00289
00290 // DEBUG
00291 STDAIR_LOG_DEBUG ("Censorship Flag: " << lCensorshipFlag);
00292
00293 // Close the log output file
00294 logOutputFile.close();
00295 }
00296
00297 // End the test suite
00298 BOOST_AUTO_TEST_SUITE_END()
00299
00300

```

43.148 test/rmol/ForecasterTestSuite.cpp File Reference

43.149 ForecasterTestSuite.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 #include <vector>
00013 #include <cmath>
00014 // Boost Unit Test Framework (UTF)
00015 #define BOOST_TEST_DYN_LINK
00016 #define BOOST_TEST_MAIN
00017 #define BOOST_TEST_MODULE ForecasterTestSuite
00018 #include <boost/test/unit_test.hpp>
00019 // StdAir
00020 #include <stdair/basic/BasLogParams.hpp>
00021 #include <stdair/basic/BasDBParams.hpp>
00022 #include <stdair/basic/BasFileMgr.hpp>
00023 #include <stdair/service/Logger.hpp>
00024 // RMOL
00025 #include <rmol/RMOL_Service.hpp>
00026
00027 namespace boost_utf = boost::unit_test;
00028
00029 // (Boost) Unit Test XML Report
00030 std::ofstream utfReportStream ("ForecasterTestSuite_utfresults.xml");
00031
00035 struct UnitTestConfig {
00037     UnitTestConfig() {
00038         boost_utf::unit_test_log.set_stream (utfReportStream);
00039         boost_utf::unit_test_log.set_format (boost_utf::XML);
00040         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00041         //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tes
00042         ts);
00042     }
00043
00044     ~UnitTestConfig() {
00046     }

```

```

00047 };
00048
00049
00050 // //////////// Main: Unit Test Suite ////////////
00051
00052 // Set the UTF configuration (re-direct the output to a specific file)
00053 BOOST_GLOBAL_FIXTURE (UnitTestFixture);
00054
00059 BOOST_AUTO_TEST_SUITE (master_test_suite)
00060
00061
00064 BOOST_AUTO_TEST_CASE (rmol_forecaster_q_forecasting) {
00065     const bool lTestFlag = true; //testForecasterHelper(0);
00066     BOOST_CHECK_EQUAL (lTestFlag, true);
00067     BOOST_CHECK_MESSAGE (lTestFlag == true,
00068         "The test has failed. Please see the log file for "
00069         "<< \"more details\");
00070 }
00071
00072 // End the test suite
00073 BOOST_AUTO_TEST_SUITE_END()
00074
00075

```

43.150 test/rmol/ForecasterTestSuite.hpp File Reference

```

#include <sstream>

#include <cppunit/extensions/HelperMacros.h>

```

Classes

- class [ForecasterTestSuite](#)

Functions

- [CPPUNIT_TEST_SUITE_REGISTRATION \(ForecasterTestSuite\)](#)

43.150.1 Function Documentation

43.150.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (ForecasterTestSuite)

43.151 ForecasterTestSuite.hpp

```

00001 // STL
00002 #include <sstream>
00003 // CPPUNIT
00004 #include <cppunit/extensions/HelperMacros.h>
00005
00006 class ForecasterTestSuite : public CppUnit::TestFixture {
00007     CPPUNIT_TEST_SUITE (ForecasterTestSuite);
00008     CPPUNIT_TEST (testQForecaster);
00009     CPPUNIT_TEST_SUITE_END ();
00010 public:

```

```

00011
00013     void testQForecaster();
00014
00016     ForecasterTestSuite ();
00017
00018 protected:
00019     std::stringstream _describeKey;
00020 };
00021
00022 CPPUNIT_TEST_SUITE_REGISTRATION (ForecasterTestSuite);

```

43.152 test/rmol/OptimiseTestSuite.cpp File Reference

43.153 OptimiseTestSuite.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 // Boost Unit Test Framework (UTF)
00013 #define BOOST_TEST_DYN_LINK
00014 #define BOOST_TEST_MAIN
00015 #define BOOST_TEST_MODULE OptimiseTestSuite
00016 #include <boost/test/unit_test.hpp>
00017 // StdAir
00018 #include <stdair/basic/BasLogParams.hpp>
00019 #include <stdair/basic/BasDBParams.hpp>
00020 #include <stdair/basic/BasFileMgr.hpp>
00021 #include <stdair/service/Logger.hpp>
00022 // RMOL
00023 #include <rmol/RMOL_Service.hpp>
00024 #include <rmol/config/rmol-paths.hpp>
00025
00026 namespace boost_utf = boost::unit_test;
00027
00028 // (Boost) Unit Test XML Report
00029 std::ofstream utfReportStream ("OptimiseTestSuite_utfresults.xml");
00030
00034 struct UnitTestConfig {
00036     UnitTestConfig() {
00037         boost_utf::unit_test_log.set_stream (utfReportStream);
00038         boost_utf::unit_test_log.set_format (boost_utf::XML);
00039         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00040         //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tes
00041         ts);
00042     }
00044     ~UnitTestConfig() {
00045     }
00046 };
00047
00048
00049 // //////////////////////////////////////
00050 int testOptimiseHelper (const unsigned short optimisationMethodFlag,
00051                        const bool isBuiltin) {
00052

```

```

00053 // Return value
00054 int oExpectedBookingLimit = 0;
00055
00056 // Output log File
00057 std::ostringstream oStr;
00058 oStr << "OptimiseTestSuite_" << optimisationMethodFlag << ".log";
00059 const stdair::Filename_T lLogFilename (oStr.str());
00060
00061 // Number of random draws to be generated (best if greater than 100)
00062 const int K = 100000;
00063
00064 // Methods of optimisation (0 = Monte-Carlo, 1 = Dynamic Programming,
00065 // 2 = EMSR, 3 = EMSR-a, 4 = EMSR-b, 5 = EMSR-a with sellup prob.)
00066 const unsigned short METHOD_FLAG = optimisationMethodFlag;
00067
00068 // Cabin Capacity (it must be greater then 100 here)
00069 const double cabinCapacity = 100.0;
00070
00071 // Set the log parameters
00072 std::ofstream logOutputFile;
00073 // Open and clean the log outputfile
00074 logOutputFile.open (lLogFilename.c_str());
00075 logOutputFile.clear();
00076
00077 // Initialise the RMOL service
00078 const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00079 RMOL::RMOL_Service rmolService (lLogParams);
00080
00081 // Check wether or not a (CSV) input file should be read
00082 if (isBuiltin == true) {
00083
00084     // Build the default sample BOM tree and build a dummy BOM tree.
00085     rmolService.buildSampleBom();
00086
00087 } else {
00088
00089     // Parse the optimisation data and build a dummy BOM tree
00090     const stdair::Filename_T lRMInputFileName (STDAIR_SAMPLE_DIR "/rm02.csv");
00091     rmolService.parseAndLoad (cabinCapacity, lRMInputFileName);
00092 }
00093
00094 switch (METHOD_FLAG) {
00095 case 0: {
00096     // DEBUG
00097     STDAIR_LOG_DEBUG ("Optimisation by Monte-Carlo (MC)");
00098
00099     // Calculate the optimal protections by the Monte Carlo
00100     // Integration approach
00101     rmolService.optimalOptimisationByMCIntegration (K);
00102     break;
00103 }
00104
00105 case 1: {
00106     // DEBUG
00107     STDAIR_LOG_DEBUG ("Optimisation by Dynamic Programming (DP)");
00108
00109     // Calculate the optimal protections by DP.
00110     rmolService.optimalOptimisationByDP ();
00111     break;
00112 }
00113
00114 case 2: {

```

```

00115     // DEBUG
00116     STDAIR_LOG_DEBUG ("Calculate the Bid-Price Vectors (BPV) by EMSR");
00117
00118     // Calculate the Bid-Price Vector by EMSR
00119     rmolService.heuristicOptimisationByEmsr ();
00120     break;
00121 }
00122
00123 case 3: {
00124     // DEBUG
00125     STDAIR_LOG_DEBUG ("Calculate the Authorisation Levels (AUs) by EMSRa");
00126
00127     // Calculate the protections by EMSR-a
00128     // Test the EMSR-a algorithm implementation
00129     rmolService.heuristicOptimisationByEmsrA ();
00130
00131     // Return a cumulated booking limit value to test
00132     // oExpectedBookingLimit = static_cast<int> (lBookingLimitVector.at(2));
00133     break;
00134 }
00135
00136 case 4: {
00137     // DEBUG
00138     STDAIR_LOG_DEBUG ("Calculate the Authorisation Levels (AUs) by EMSRb");
00139
00140     // Calculate the protections by EMSR-b
00141     rmolService.heuristicOptimisationByEmsrB ();
00142     break;
00143 }
00144
00145 default: rmolService.optimalOptimisationByMCIntegration (K);
00146 }
00147
00148 // Close the log file
00149 logOutputFile.close();
00150
00151 return oExpectedBookingLimit;
00152 }
00153
00154
00155 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////
00156
00157 // Set the UTF configuration (re-direct the output to a specific file)
00158 BOOST_GLOBAL_FIXTURE (UnitTestFixture);
00159
00160 // ////////////////////////////////////////
00161 // Tests are based on the following input values
00162 // price; mean; standard deviation;
00163 // 1050; 17.3; 5.8;
00164 // 567; 45.1; 15.0;
00165 // 534; 39.6; 13.2;
00166 // 520; 34.0; 11.3;
00167 // ////////////////////////////////////////
00168
00173 BOOST_AUTO_TEST_SUITE (master_test_suite)
00174
00175
00178 BOOST_AUTO_TEST_CASE (rmol_optimisation_monte_carlo) {
00179
00180     // State whether the BOM tree should be built-in or parsed from an input file
00181     const bool isBuiltin = false;
00182

```



```
00183 BOOST_CHECK_NO_THROW (testOptimiseHelper(0, isBuiltin));
00184 }
00185
00189 BOOST_AUTO_TEST_CASE (rmol_optimisation_dynamic_programming) {
00190
00191     // State whether the BOM tree should be built-in or parsed from an input file
00192     const bool isBuiltin = false;
00193
00194     BOOST_CHECK_NO_THROW (testOptimiseHelper(1, isBuiltin));
00195 }
00196
00201 BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_bpv) {
00202
00203     // State whether the BOM tree should be built-in or parsed from an input file
00204     const bool isBuiltin = false;
00205
00206     BOOST_CHECK_NO_THROW (testOptimiseHelper(2, isBuiltin));
00207 }
00208
00213 BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_a) {
00214
00215     // State whether the BOM tree should be built-in or parsed from an input file
00216     const bool isBuiltin = false;
00217
00218     BOOST_CHECK_NO_THROW (testOptimiseHelper(3, isBuiltin));
00219     // const int lBookingLimit = testOptimiseHelper(3);
00220     // const int lExpectedBookingLimit = 61;
00221     // BOOST_CHECK_EQUAL (lBookingLimit, lExpectedBookingLimit);
00222     // BOOST_CHECK_MESSAGE (lBookingLimit == lExpectedBookingLimit,
00223         // "The booking limit is " << lBookingLimit
00224         // << ", but it is expected to be "
00225         // << lExpectedBookingLimit);
00226 }
00227
00232 BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_b) {
00233
00234     // State whether the BOM tree should be built-in or parsed from an input file
00235     const bool isBuiltin = false;
00236
00237     BOOST_CHECK_NO_THROW (testOptimiseHelper(4, isBuiltin));
00238 }
00239
00243 BOOST_AUTO_TEST_CASE (rmol_optimisation_monte_carlo_built_in) {
00244
00245     // State whether the BOM tree should be built-in or parsed from an input file
00246     const bool isBuiltin = true;
00247
00248     BOOST_CHECK_NO_THROW (testOptimiseHelper(5, isBuiltin));
00249 }
00250
00254 BOOST_AUTO_TEST_CASE (rmol_optimisation_dynamic_programming_built_in) {
00255
00256     // State whether the BOM tree should be built-in or parsed from an input file
00257     const bool isBuiltin = true;
00258
00259     BOOST_CHECK_NO_THROW (testOptimiseHelper(6, isBuiltin));
00260 }
00261
00266 BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_bpv_built_in) {
00267
00268     // State whether the BOM tree should be built-in or parsed from an input file
00269     const bool isBuiltin = true;
```

```

00270
00271     BOOST_CHECK_NO_THROW (testOptimiseHelper(7, isBuiltin));
00272 }
00273
00278 BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_a_built_in) {
00279
00280     // State whether the BOM tree should be built-in or parsed from an input file
00281     const bool isBuiltin = true;
00282
00283     BOOST_CHECK_NO_THROW (testOptimiseHelper(8, isBuiltin));
00284 }
00285
00290 BOOST_AUTO_TEST_CASE (rmol_optimisation_emsr_b_built_in) {
00291
00292     // State whether the BOM tree should be built-in or parsed from an input file
00293     const bool isBuiltin = true;
00294
00295     BOOST_CHECK_NO_THROW (testOptimiseHelper(9, isBuiltin));
00296 }
00297
00298 // End the test suite
00299 BOOST_AUTO_TEST_SUITE_END()
00300
00301

```

43.154 test/rmol/OptimiseTestSuite.hpp File Reference

```

#include <sstream>
#include <cppunit/extensions/HelperMacros.h>

```

Classes

- class [OptimiseTestSuite](#)

Functions

- [CPPUNIT_TEST_SUITE_REGISTRATION \(OptimiseTestSuite\)](#)

43.154.1 Function Documentation

43.154.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (OptimiseTestSuite)

43.155 OptimiseTestSuite.hpp

```

00001 // STL
00002 #include <sstream>
00003 // CPPUNIT
00004 #include <cppunit/extensions/HelperMacros.h>
00005
00006 class OptimiseTestSuite : public CppUnit::TestFixture {
00007     CPPUNIT_TEST_SUITE (OptimiseTestSuite);
00008     CPPUNIT_TEST (testOptimiseMC);
00009     CPPUNIT_TEST (testOptimiseDP);

```

```

00010 CPPUNIT_TEST (testOptimiseEMSR);
00011 CPPUNIT_TEST (testOptimiseEMSRa);
00012 CPPUNIT_TEST (testOptimiseEMSRb);
00013 CPPUNIT_TEST (testOptimiseEMSRaWithSU);
00014 // CPPUNIT_TEST (errorCase);
00015 CPPUNIT_TEST_SUITE_END ();
00016 public:
00017
00019 void testOptimiseMC ();
00020
00022 void testOptimiseDP ();
00023
00026 void testOptimiseEMSR ();
00027
00030 void testOptimiseEMSRa ();
00031
00034 void testOptimiseEMSRb ();
00035
00037 // void errorCase ();
00038
00040 OptimiseTestSuite ();
00041
00042 protected:
00043 std::stringstream _describeKey;
00044 };
00045
00046 CPPUNIT_TEST_SUITE_REGISTRATION (OptimiseTestSuite);

```

43.156 test/rmol/UnconstrainerTestSuite.cpp File Reference

43.157 UnconstrainerTestSuite.cpp

```

00001
00005 // //////////////////////////////////////
00006 // Import section
00007 // //////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 // Boost Unit Test Framework (UTF)
00013 #define BOOST_TEST_DYN_LINK
00014 #define BOOST_TEST_MAIN
00015 #define BOOST_TEST_MODULE UnconstrainerTestSuite
00016 #include <boost/test/unit_test.hpp>
00017 // StdAir
00018 #include <stdair/basic/BasLogParams.hpp>
00019 #include <stdair/basic/BasDBParams.hpp>
00020 #include <stdair/basic/BasFileMgr.hpp>
00021 #include <stdair/service/Logger.hpp>
00022 // RMOL
00023 #include <rmol/RMOL_Service.hpp>
00024
00025 namespace boost_utf = boost::unit_test;
00026
00027 // (Boost) Unit Test XML Report
00028 std::ofstream utfReportStream ("UnconstrainerTestSuite_utfresults.xml");
00029
00033 struct UnitTestConfig {
00035     UnitTestConfig() {

```

```

00036     boost_utf::unit_test_log.set_stream (utfReportStream);
00037     boost_utf::unit_test_log.set_format (boost_utf::XML);
00038     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00039     //boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_tes
ts);
00040 }
00041
00043 ~UnitTestConfig() {
00044 }
00045 };
00046
00047
00048 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////
00049
00050 // Set the UTF configuration (re-direct the output to a specific file)
00051 BOOST_GLOBAL_FIXTURE (UnitTestConfig);
00052
00057 BOOST_AUTO_TEST_SUITE (master_test_suite)
00058
00059
00062 BOOST_AUTO_TEST_CASE (rmol_unconstraining_em) {
00063     const bool lTestFlag = true; // testUnconstrainerHelper(0);
00064     BOOST_CHECK_EQUAL (lTestFlag, true);
00065     BOOST_CHECK_MESSAGE (lTestFlag == true,
00066         "The test has failed. Please see the log file for "
00067         "<< "more details");
00068 }
00069
00070 // End the test suite
00071 BOOST_AUTO_TEST_SUITE_END()
00072
00073

```

43.158 test/rmol/UnconstrainerTestSuite.hpp File Reference

```

#include <sstream>

#include <cppunit/extensions/HelperMacros.h>

```

Classes

- class [UnconstrainerTestSuite](#)

Functions

- [CPPUNIT_TEST_SUITE_REGISTRATION \(UnconstrainerTestSuite\)](#)

43.158.1 Function Documentation

43.158.1.1 CPPUNIT_TEST_SUITE_REGISTRATION (UnconstrainerTestSuite)

43.159 UnconstrainerTestSuite.hpp

```

00001 // STL

```

```
00002 #include <sstream>
00003 // CPPUNIT
00004 #include <cppunit/extensions/HelperMacros.h>
00005
00006 class UnconstrainerTestSuite : public CppUnit::TestFixture {
00007     CPPUNIT_TEST_SUITE (UnconstrainerTestSuite);
00008     CPPUNIT_TEST (testUnconstrainingByEM);
00009     CPPUNIT_TEST_SUITE_END ();
00010 public:
00011
00012     void testUnconstrainingByEM();
00013
00014     UnconstrainerTestSuite ();
00015
00016 protected:
00017     std::stringstream _describeKey;
00018 };
00019
00020 CPPUNIT_TEST_SUITE_REGISTRATION (UnconstrainerTestSuite);
```

Index

- ~DemandGeneratorList
 - RMOL::DemandGeneratorList, [103](#)
- ~FacRmolServiceContext
 - RMOL::FacRmolServiceContext, [110](#)
- ~HistoricalBooking
 - RMOL::HistoricalBooking, [116](#)
- ~HistoricalBookingHolder
 - RMOL::HistoricalBookingHolder, [119](#)
- ~RMOL_Service
 - RMOL::RMOL_Service, [132](#)
- _describeKey
 - ForecasterTestSuite, [113](#)
 - OptimiseTestSuite, [128](#)
 - UnconstrainerTestSuite, [140](#)
- addHistoricalBooking
 - RMOL::HistoricalBookingHolder, [121](#)
- BINDIR
 - rmol-paths.hpp, [259](#)
- BookingClassUnconstrainedDemandMap_
 - T
 - RMOL, [99](#)
- BookingClassUnconstrainedDemandVectorMap_
 - T
 - RMOL, [99](#)
- BookingVector_T
 - RMOL, [99](#)
- bpsk.cpp
 - main, [144](#)
- BucketHolderList_T
 - RMOL, [98](#)
- buildRemainingDCPList
 - RMOL::Utilities, [142](#)
- buildRemainingDCPList2
 - RMOL::Utilities, [142](#)
- buildSampleBom
 - RMOL::RMOL_Service, [132](#)
- buildVirtualClassListForLegBasedOptimisation
 - RMOL::Optimiser, [126](#)
- calculateExpectedDemand
 - RMOL::HistoricalBookingHolder, [121](#)
- cdfGaussianQ
 - RMOL::DPOptimiser, [106](#)
- computeAggregatedVirtualClass
 - RMOL::EmsrUtils, [108](#)
- computeDistributionParameters
 - RMOL::Utilities, [142](#)
- computeEmsrValue
 - RMOL::EmsrUtils, [109](#)
- computeProtectionLevel
 - RMOL::EmsrUtils, [109](#)
- convcode.cpp
 - main, [145](#)
- CppUnit::TestFixture, [139](#)
- CPPUNIT_TEST_SUITE_REGISTRATION
 - ForecasterTestSuite.hpp, [311](#)
 - OptimiseTestSuite.hpp, [316](#)
 - UnconstrainerTestSuite.hpp, [318](#)
- create
 - RMOL::FacRmolServiceContext, [110](#)
- createCumulativeFRAT5Curve
 - RMOL::DefaultMap, [102](#)
- csvDisplay
 - RMOL::RMOL_Service, [137](#)
- DATADIR
 - rmol-paths.hpp, [260](#)
- DATAROOTDIR
 - rmol-paths.hpp, [260](#)
- DEFAULT_CUMULATIVE_FRAT5_CURVE
 - RMOL, [100](#)
- DEFAULT_DCP_LIST
 - RMOL, [100](#)
- DEFAULT_EPSILON
 - RMOL, [100](#)
- DEFAULT_INITIALIZER_DOUBLE_NEGATIVE
 - RMOL, [100](#)
- DEFAULT_NUMBER_OF_DRAWS_FOR_MC_SIMULATION
 - RMOL, [100](#)
- DEFAULT_PRECISION
 - RMOL, [100](#)
- DEFAULT_RMOL_SERVICE_AIRLINE_CODE
 - RMOL, [99](#)
- DEFAULT_RMOL_SERVICE_CAPACITY
 - RMOL, [99](#)
- DEFAULT_STOPPING_CRITERION
 - RMOL, [100](#)
- DemandGeneratorList
 - RMOL::DemandGeneratorList, [103](#)
- DemandGeneratorList_T
 - RMOL::DemandGeneratorList, [103](#)

- describe
 - RMOL::HistoricalBooking, 117
 - RMOL::HistoricalBookingHolder, 122
- display
 - RMOL::HistoricalBooking, 117
 - RMOL::HistoricalBookingHolder, 122
- DistributionParameterList_T
 - RMOL, 98
- doc/ Directory Reference, 95
- doc/local/authors.doc, 142
- doc/local/codingrules.doc, 142
- doc/local/copyright.doc, 142
- doc/local/documentation.doc, 143
- doc/local/features.doc, 143
- doc/local/help_wanted.doc, 143
- doc/local/howto_release.doc, 143
- doc/local/index.doc, 143
- doc/local/installation.doc, 143
- doc/local/linking.doc, 143
- doc/local/test.doc, 143
- doc/local/users_guide.doc, 143
- doc/local/verification.doc, 143
- doc/tutorial/ Directory Reference, 96
- doc/tutorial/bpsk.doc, 143
- doc/tutorial/convcode.doc, 143
- doc/tutorial/interleaver.doc, 143
- doc/tutorial/itfile.doc, 143
- doc/tutorial/ldpc_bersim_awgn.doc, 143
- doc/tutorial/ldpc_gen_codes.doc, 143
- doc/tutorial/matlab_itpp.doc, 143
- doc/tutorial/mimoconv.doc, 143
- doc/tutorial/mog.doc, 143
- doc/tutorial/qpsk_simulation.doc, 143
- doc/tutorial/rayleigh.doc, 143
- doc/tutorial/reedsolomon.doc, 143
- doc/tutorial/spread.doc, 144
- doc/tutorial/src/ Directory Reference, 96
- doc/tutorial/src/bpsk.cpp, 144
- doc/tutorial/src/convcode.cpp, 145
- doc/tutorial/src/interleaver.cpp, 147
- doc/tutorial/src/ldpc_bersim_awgn.cpp, 148
- doc/tutorial/src/ldpc_gen_codes.cpp, 150
- doc/tutorial/src/mimoconv.cpp, 152
- doc/tutorial/src/mog.cpp, 157
- doc/tutorial/src/qpsk_simulation.cpp, 160
- doc/tutorial/src/rayleigh.cpp, 162
- doc/tutorial/src/read_it_file.cpp, 162, 163
- doc/tutorial/src/reedsolomon.cpp, 163, 164
- doc/tutorial/src/spread.cpp, 165
- doc/tutorial/src/timer.cpp, 167
- doc/tutorial/src/vector_and_matrix.cpp, 168
- doc/tutorial/src/write_it_file.cpp, 169
- doc/tutorial/timer.doc, 170
- doc/tutorial/tutorial.doc, 170
- doc/tutorial/vector_and_matrix.doc, 170
- DOCDIR
 - rmol-paths.hpp, 260
- EXEC_PREFIX
 - rmol-paths.hpp, 259
- FacRmolServiceContext
 - RMOL::FacRmolServiceContext, 110
 - RMOL::RMOL_ServiceContext, 138
- FlagVector_T
 - RMOL, 99
- ForecasterTestSuite, 112
 - _describeKey, 113
 - ForecasterTestSuite, 113
 - testQForecaster, 113
- ForecasterTestSuite.hpp
 - CPPUNIT_TEST_SUITE_REGISTRATION, 311
- ForecastException
 - RMOL::ForecastException, 114
- forecastOnD
 - RMOL::RMOL_Service, 134
- forecastUsingAdditivePickUp
 - RMOL::Forecaster, 112
- forecastUsingMultiplicativePickUp
 - RMOL::Forecaster, 112
- FRAT5Curve_T
 - RMOL, 99
- generateDemandVector
 - RMOL::MCOptimiser, 124
- generateVariateList
 - RMOL::DemandGeneratorList, 104
- getCensorshipFlag
 - RMOL::HistoricalBookingHolder, 121
- getDemandMean
 - RMOL::HistoricalBookingHolder, 120
- getFlag
 - RMOL::HistoricalBooking, 117
- getHistoricalBooking
 - RMOL::HistoricalBookingHolder, 120
- getListOfToBeUnconstrainedFlags
 - RMOL::HistoricalBookingHolder, 120
- getNbOfBookings
 - RMOL::HistoricalBooking, 116

- getNbOfDepartedSimilarSegments
 - RMOL::Utilities, 142
- getNbOfFlights
 - RMOL::HistoricalBookingHolder, 119
- getNbOfSegmentAlreadyPassedThisDTD
 - RMOL::GuillotineBlockHelper, 114
- getNbOfUncensoredBookings
 - RMOL::HistoricalBookingHolder, 119
- getNbOfUncensoredData
 - RMOL::HistoricalBookingHolder, 119
- getStandardDeviation
 - RMOL::HistoricalBookingHolder, 120
- getUncensoredStandardDeviation
 - RMOL::HistoricalBookingHolder, 119
- getUnconstrainedDemand
 - RMOL::HistoricalBooking, 116
 - RMOL::HistoricalBookingHolder, 120
- getUnconstrainedDemandOnFirstElement
 - RMOL::HistoricalBookingHolder, 121
- getYieldFeatures
 - RMOL::RMOL_Service, 134
- hasPassedThisDTD
 - RMOL::GuillotineBlockHelper, 114
- heuristicOptimisationByEmsr
 - RMOL::Emsr, 107
 - RMOL::Optimiser, 126
 - RMOL::RMOL_Service, 133
- heuristicOptimisationByEmsrA
 - RMOL::Emsr, 108
 - RMOL::Optimiser, 126
 - RMOL::RMOL_Service, 133
- heuristicOptimisationByEmsrB
 - RMOL::Emsr, 108
 - RMOL::Optimiser, 126
 - RMOL::RMOL_Service, 133
- HistoricalBooking
 - RMOL::HistoricalBooking, 116
- HistoricalBookingHolder
 - RMOL::HistoricalBookingHolder, 119
- HistoricalBookingVector_T
 - RMOL, 98
- HTMLDIR
 - rmol-paths.hpp, 260
- INCLUDEDIR
 - rmol-paths.hpp, 260
- INFODIR
 - rmol-paths.hpp, 260
- init
 - RMOL::DefaultDCPList, 102
- instance
 - RMOL::FacRmolServiceContext, 110
- interleaver.cpp
 - main, 147
- jsonExport
 - RMOL::RMOL_Service, 136
- K_RMOL_DEFAULT_BUILT_IN_INPUT
 - rmol.cpp, 175
- K_RMOL_DEFAULT_CAPACITY
 - rmol.cpp, 176
- K_RMOL_DEFAULT_INPUT_FILENAME
 - rmol.cpp, 175
- K_RMOL_DEFAULT_LOG_FILENAME
 - rmol.cpp, 175
- K_RMOL_DEFAULT_METHOD
 - rmol.cpp, 176
- K_RMOL_DEFAULT_RANDOM_DRAWS
 - rmol.cpp, 176
- K_RMOL_EARLY_RETURN_STATUS
 - rmol.cpp, 176
- ldpc_bersim_awgn.cpp
 - main, 148
- ldpc_gen_codes.cpp
 - main, 150
- LIBDIR
 - rmol-paths.hpp, 259
- LIBEXECDIR
 - rmol-paths.hpp, 259
- main
 - bpsk.cpp, 144
 - convcode.cpp, 145
 - interleaver.cpp, 147
 - ldpc_bersim_awgn.cpp, 148
 - ldpc_gen_codes.cpp, 150
 - mimoconv.cpp, 152
 - mog.cpp, 157
 - qpsk_simulation.cpp, 160
 - rayleigh.cpp, 162
 - read_it_file.cpp, 163
 - reedsolomon.cpp, 163
 - rmol.cpp, 175
 - spread.cpp, 165
 - timer.cpp, 167
 - vector_and_matrix.cpp, 168
 - write_it_file.cpp, 169

- MANDIR
 - rmol-paths.hpp, 260
- mimoconv.cpp
 - main, 152
 - ZF_demod, 152
- mog.cpp
 - main, 157
- operator<<
 - rmol.cpp, 175
- optimalOptimisationByDP
 - RMOL::DPOptimiser, 106
 - RMOL::Optimiser, 126
 - RMOL::RMOL_Service, 133
- optimalOptimisationByMCIntegration
 - RMOL::MCOptimiser, 124
 - RMOL::Optimiser, 126
 - RMOL::RMOL_Service, 133
- optimisationByMCIntegration
 - RMOL::MCOptimiser, 124
- OptimisationException
 - RMOL::OptimisationException, 125
- optimise
 - rmol.cpp, 175
 - RMOL::Optimiser, 126
 - RMOL::RMOL_Service, 133
- optimiseOnD
 - RMOL::RMOL_Service, 136
- optimiseOnDUsingAdvancedRMCooperation
 - RMOL::RMOL_Service, 136
- optimiseOnDUsingRMCooperation
 - RMOL::RMOL_Service, 136
- OptimiseTestSuite, 127
 - _describeKey, 128
 - OptimiseTestSuite, 128
 - testOptimiseDP, 128
 - testOptimiseEMSR, 128
 - testOptimiseEMSRa, 128
 - testOptimiseEMSRb, 128
 - testOptimiseMC, 128
- OptimiseTestSuite.hpp
 - CPPUNIT_TEST_SUITE_REGISTRATION, 316
- optimiseUsingOnDForecast
 - RMOL::Optimiser, 127
- OverbookingException
 - RMOL::OverbookingException, 129
- PACKAGE
 - rmol-paths.hpp, 259
- PACKAGE_NAME
 - rmol-paths.hpp, 259
- PACKAGE_VERSION
 - rmol-paths.hpp, 259
- parseAndLoad
 - RMOL::RMOL_Service, 132
- parseInputFileAndBuildBom
 - RMOL::InventoryParser, 123
- PDFDIR
 - rmol-paths.hpp, 260
- PREFIXDIR
 - rmol-paths.hpp, 259
- projectAggregatedDemandOnLegCabins
 - RMOL::RMOL_Service, 135
- projectOnDDemandOnLegCabinsUsingDA
 - RMOL::RMOL_Service, 135
- projectOnDDemandOnLegCabinsUsingDYP
 - RMOL::RMOL_Service, 135, 136
- projectOnDDemandOnLegCabinsUsingYP
 - RMOL::RMOL_Service, 135
- qpsk_simulation.cpp
 - main, 160
- rayleigh.cpp
 - main, 162
- read_it_file.cpp
 - main, 163
- readConfiguration
 - rmol.cpp, 175
- reedsolomon.cpp
 - main, 163
- resetDemandInformation
 - RMOL::RMOL_Service, 135
- retrieveUnconstrainedDemandForFirstDCP
 - RMOL::Detruncator, 105
- RMOL, 97
 - BookingClassUnconstrainedDemandMap_
T, 99
 - BookingClassUnconstrainedDemandVectorMap_
T, 99
 - BookingVector_T, 99
 - BucketHolderList_T, 98
 - DEFAULT_CUMULATIVE_FRAT5_CURVE,
100
 - DEFAULT_DCP_LIST, 100
 - DEFAULT_EPSILON, 100
 - DEFAULT_INITIALIZER_DOUBLE_NEGATIVE,
100

- DEFAULT_NUMBER_OF_DRAWNS_FOR_ - readConfiguration, 175
- MC_SIMULATION, 100
- DEFAULT_PRECISION, 100
- DEFAULT_RMOL_SERVICE_AIRLINE_CODE, 99
- DEFAULT_RMOL_SERVICE_CAPACITY, 99
- DEFAULT_STOPPING_CRITERION, 100
- DistributionParameterList_T, 98
- FlagVector_T, 99
- FRAT5Curve_T, 99
- HistoricalBookingVector_T, 98
- RMOL_ServicePtr_T, 98
- UnconstrainedDemandVector_T, 99
- rmol-paths.hpp
 - BINDIR, 259
 - DATADIR, 260
 - DATAROOTDIR, 260
 - DOCDIR, 260
 - EXEC_PREFIX, 259
 - HTMLDIR, 260
 - INCLUDEDIR, 260
 - INFODIR, 260
 - LIBDIR, 259
 - LIBEXECDIR, 259
 - MANDIR, 260
 - PACKAGE, 259
 - PACKAGE_NAME, 259
 - PACKAGE_VERSION, 259
 - PDFDIR, 260
 - PREFIXDIR, 259
 - SBINDIR, 260
 - STDAIR_SAMPLE_DIR, 260
 - SYSCONFDIR, 260
- rmol.cpp
 - K_RMOL_DEFAULT_BUILT_IN_INPUT, 175
 - K_RMOL_DEFAULT_CAPACITY, 176
 - K_RMOL_DEFAULT_INPUT_FILENAME, 175
 - K_RMOL_DEFAULT_LOG_FILENAME, 175
 - K_RMOL_DEFAULT_METHOD, 176
 - K_RMOL_DEFAULT_RANDOM_DRAWNS, 176
 - K_RMOL_EARLY_RETURN_STATUS, 176
 - main, 175
 - operator<<, 175
 - optimise, 175
 - rmol/ Directory Reference, 95
 - rmol/basic/ Directory Reference, 93
 - rmol/basic/BasConst.cpp, 170
 - rmol/basic/BasConst_Curves.hpp, 172
 - rmol/basic/BasConst_General.hpp, 172, 173
 - rmol/basic/BasConst_RMOL_Service.hpp, 173
 - rmol/batches/ Directory Reference, 93
 - rmol/batches/rmol.cpp, 174, 176
 - rmol/bom/ Directory Reference, 94
 - rmol/bom/BucketHolderTypes.hpp, 181
 - rmol/bom/DistributionParameterList.hpp, 182
 - rmol/bom/DPOptimiser.cpp, 182, 183
 - rmol/bom/DPOptimiser.hpp, 186
 - rmol/bom/EMDetruncator.cpp, 187
 - rmol/bom/EMDetruncator.hpp, 189
 - rmol/bom/Emsr.cpp, 190
 - rmol/bom/Emsr.hpp, 193
 - rmol/bom/EmsrUtils.cpp, 193, 194
 - rmol/bom/EmsrUtils.hpp, 195, 196
 - rmol/bom/GuillotineBlockHelper.cpp, 196, 197
 - rmol/bom/GuillotineBlockHelper.hpp, 198
 - rmol/bom/HistoricalBooking.cpp, 199
 - rmol/bom/HistoricalBooking.hpp, 200, 201
 - rmol/bom/HistoricalBookingHolder.cpp, 202
 - rmol/bom/HistoricalBookingHolder.hpp, 206, 207
 - rmol/bom/MCOptimiser.cpp, 208, 209
 - rmol/bom/MCOptimiser.hpp, 214
 - rmol/bom/old/ Directory Reference, 95
 - rmol/bom/old/DemandGeneratorList.cpp, 215
 - rmol/bom/old/DemandGeneratorList.hpp, 216
 - rmol/bom/Utilities.cpp, 217, 218
 - rmol/bom/Utilities.hpp, 220
 - rmol/command/ Directory Reference, 94
 - rmol/command/Detruncator.cpp, 220, 221
 - rmol/command/Detruncator.hpp, 231
 - rmol/command/Forecaster.cpp, 233, 234
 - rmol/command/Forecaster.hpp, 248
 - rmol/command/InventoryParser.cpp, 250
 - rmol/command/InventoryParser.hpp, 253
 - rmol/command/Optimiser.cpp, 254
 - rmol/command/Optimiser.hpp, 257, 258
 - rmol/config/ Directory Reference, 94
 - rmol/config/rmol-paths.hpp, 258, 260
 - rmol/factory/ Directory Reference, 95
 - rmol/factory/FacRmolServiceContext.cpp, 261
 - rmol/factory/FacRmolServiceContext.hpp, 262

- rmol/RMOL_Service.hpp, 263, 264
- rmol/RMOL_Types.hpp, 267, 268
- rmol/service/ Directory Reference, 96
- rmol/service/RMOL_Service.cpp, 269, 270
- rmol/service/RMOL_ServiceContext.cpp, 302
- rmol/service/RMOL_ServiceContext.hpp, 303, 304
- RMOL::DefaultDCPList, 101
 - init, 102
- RMOL::DefaultMap, 102
 - createCumulativeFRAT5Curve, 102
- RMOL::DemandGeneratorList, 102
 - ~DemandGeneratorList, 103
 - DemandGeneratorList, 103
 - DemandGeneratorList_T, 103
 - generateVariateList, 104
- RMOL::Detruncator, 104
 - retrieveUnconstrainedDemandForFirst-DCP, 105
 - unconstrainUsingAdditivePickUp, 104
 - unconstrainUsingMultiplicativePickUp, 105
- RMOL::DPOptimiser, 106
 - cdfGaussianQ, 106
 - optimalOptimisationByDP, 106
- RMOL::EMDetruncator, 106
 - unconstrainUsingEMMethod, 107
- RMOL::Emsr, 107
 - heuristicOptimisationByEmsr, 107
 - heuristicOptimisationByEmsrA, 108
 - heuristicOptimisationByEmsrB, 108
- RMOL::EmsrUtils, 108
 - computeAggregatedVirtualClass, 108
 - computeEmsrValue, 109
 - computeProtectionLevel, 109
- RMOL::FacRmolServiceContext, 109
 - ~FacRmolServiceContext, 110
 - create, 110
 - FacRmolServiceContext, 110
 - instance, 110
- RMOL::Forecaster, 111
 - forecastUsingAdditivePickUp, 112
 - forecastUsingMultiplicativePickUp, 112
- RMOL::ForecastException, 113
 - ForecastException, 114
- RMOL::GuillotineBlockHelper, 114
 - getNbOfSegmentAlreadyPassedThis-DTD, 114
 - hasPassedThisDTD, 114
- RMOL::HistoricalBooking, 115
 - ~HistoricalBooking, 116
 - describe, 117
 - display, 117
 - getFlag, 117
 - getNbOfBookings, 116
 - getUnconstrainedDemand, 116
 - HistoricalBooking, 116
 - setParameters, 117
 - setUnconstrainedDemand, 117
 - toStream, 117
- RMOL::HistoricalBookingHolder, 118
 - ~HistoricalBookingHolder, 119
 - addHistoricalBooking, 121
 - calculateExpectedDemand, 121
 - describe, 122
 - display, 122
 - getCensorshipFlag, 121
 - getDemandMean, 120
 - getHistoricalBooking, 120
 - getListOfToBeUnconstrainedFlags, 120
 - getNbOfFlights, 119
 - getNbOfUncensoredBookings, 119
 - getNbOfUncensoredData, 119
 - getStandardDeviation, 120
 - getUncensoredStandardDeviation, 119
 - getUnconstrainedDemand, 120
 - getUnconstrainedDemandOnFirstElement, 121
 - HistoricalBookingHolder, 119
 - setUnconstrainedDemand, 121
 - toStream, 121
- RMOL::InventoryParser, 122
 - parseInputFileAndBuildBom, 123
- RMOL::MCOptimiser, 123
 - generateDemandVector, 124
 - optimalOptimisationByMCIntegration, 124
 - optimisationByMCIntegration, 124
- RMOL::OptimisationException, 124
 - OptimisationException, 125
- RMOL::Optimiser, 125
 - buildVirtualClassListForLegBasedOptimisation, 126
 - heuristicOptimisationByEmsr, 126
 - heuristicOptimisationByEmsrA, 126
 - heuristicOptimisationByEmsrB, 126
 - optimalOptimisationByDP, 126
 - optimalOptimisationByMCIntegration, 126
 - optimise, 126

- optimiseUsingOnDForecast, [127](#)
- RMOL::OverbookingException, [128](#)
 - OverbookingException, [129](#)
- RMOL::RMOL_Service, [129](#)
 - ~RMOL_Service, [132](#)
 - buildSampleBom, [132](#)
 - csvDisplay, [137](#)
 - forecastOnD, [134](#)
 - getYieldFeatures, [134](#)
 - heuristicOptimisationByEmsr, [133](#)
 - heuristicOptimisationByEmsrA, [133](#)
 - heuristicOptimisationByEmsrB, [133](#)
 - jsonExport, [136](#)
 - optimalOptimisationByDP, [133](#)
 - optimalOptimisationByMCIntegration, [133](#)
 - optimise, [133](#)
 - optimiseOnD, [136](#)
 - optimiseOnDUsingAdvancedRMCooperation, [136](#)
 - optimiseOnDUsingRMCooperation, [136](#)
 - parseAndLoad, [132](#)
 - projectAggregatedDemandOnLegCabins, [135](#)
 - projectOnDDemandOnLegCabinsUsingDA, [135](#)
 - projectOnDDemandOnLegCabinsUsingDYP, [135](#), [136](#)
 - projectOnDDemandOnLegCabinsUsingYP, [135](#)
 - resetDemandInformation, [135](#)
 - RMOL_Service, [131](#)
 - setOnDForecast, [134](#)
 - setUpStudyStatManager, [132](#)
 - updateBidPrice, [136](#)
- RMOL::RMOL_ServiceContext, [137](#)
 - FacRmolServiceContext, [138](#)
 - RMOL_Service, [138](#)
- RMOL::UnconstrainingException, [140](#)
 - UnconstrainingException, [141](#)
- RMOL::Utilities, [141](#)
 - buildRemainingDCPList, [142](#)
 - buildRemainingDCPList2, [142](#)
 - computeDistributionParameters, [142](#)
 - getNbOfDepartedSimilarSegments, [142](#)
- RMOL_Service
 - RMOL::RMOL_Service, [131](#)
 - RMOL::RMOL_ServiceContext, [138](#)
- RMOL_ServicePtr_T
 - RMOL, [98](#)
- SBINDIR
 - rmol-paths.hpp, [260](#)
- setOnDForecast
 - RMOL::RMOL_Service, [134](#)
- setParameters
 - RMOL::HistoricalBooking, [117](#)
- setUnconstrainedDemand
 - RMOL::HistoricalBooking, [117](#)
 - RMOL::HistoricalBookingHolder, [121](#)
- setUpStudyStatManager
 - RMOL::RMOL_Service, [132](#)
- spread.cpp
 - main, [165](#)
- stdair, [101](#)
 - stdair::CmdAbstract, [101](#)
 - stdair::FacServiceAbstract, [111](#)
 - stdair::RootException, [138](#)
 - stdair::ServiceAbstract, [138](#)
 - stdair::StructAbstract, [139](#)
- STDAIR_SAMPLE_DIR
 - rmol-paths.hpp, [260](#)
- SYSCONFDIR
 - rmol-paths.hpp, [260](#)
- test/ Directory Reference, [96](#)
- test/rmol/ Directory Reference, [95](#)
- test/rmol/bomsforforecaster.cpp, [305](#)
- test/rmol/ForecasterTestSuite.cpp, [310](#)
- test/rmol/ForecasterTestSuite.hpp, [311](#)
- test/rmol/OptimiseTestSuite.cpp, [312](#)
- test/rmol/OptimiseTestSuite.hpp, [316](#)
- test/rmol/UnconstrainerTestSuite.cpp, [317](#)
- test/rmol/UnconstrainerTestSuite.hpp, [318](#)
- testOptimiseDP
 - OptimiseTestSuite, [128](#)
- testOptimiseEMSR
 - OptimiseTestSuite, [128](#)
- testOptimiseEMSRa
 - OptimiseTestSuite, [128](#)
- testOptimiseEMSRb
 - OptimiseTestSuite, [128](#)
- testOptimiseMC
 - OptimiseTestSuite, [128](#)
- testQForecaster
 - ForecasterTestSuite, [113](#)
- testUnconstrainingByEM
 - UnconstrainerTestSuite, [140](#)
- timer.cpp
 - main, [167](#)
- toStream

- RMOL::HistoricalBooking, [117](#)
- RMOL::HistoricalBookingHolder, [121](#)
- UnconstrainedDemandVector_T
 - RMOL, [99](#)
- UnconstrainerTestSuite, [139](#)
 - _describeKey, [140](#)
 - testUnconstrainingByEM, [140](#)
 - UnconstrainerTestSuite, [140](#)
- UnconstrainerTestSuite.hpp
 - CPPUNIT_TEST_SUITE_REGISTRATION,
[318](#)
- UnconstrainingException
 - RMOL::UnconstrainingException, [141](#)
- unconstrainUsingAdditivePickUp
 - RMOL::Detruncator, [104](#)
- unconstrainUsingEMMethod
 - RMOL::EMDetruncator, [107](#)
- unconstrainUsingMultiplicativePickUp
 - RMOL::Detruncator, [105](#)
- updateBidPrice
 - RMOL::RMOL_Service, [136](#)
- vector_and_matrix.cpp
 - main, [168](#)
- write_it_file.cpp
 - main, [169](#)
- ZF_demod
 - mimoconv.cpp, [152](#)