

# Contents

<b>1 DynareOBC: A toolkit for handling occasionally binding constraints with Dynare, by Tom Holden.</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Installation . . . . .	1
1.3 Requirements . . . . .	1
1.4 Additional requirements for people using Windows, without administrative rights . . . . .	2
1.5 Troubleshooting . . . . .	2
1.6 Basic Usage . . . . .	3
1.7 Supported options inside the .mod file . . . . .	8
1.8 Advanced Usage . . . . .	8
1.9 Acknowledgements and copyright information . . . . .	8

## 1 DynareOBC: A toolkit for handling occasionally binding constraints with Dynare, by Tom Holden.

### 1.1 Background

Please read the theory paper from here: <https://github.com/tholden/dynareOBC/raw/master/TheoryPaper.pdf>  
(Or here for citations: <https://ideas.repec.org/p/zbw/esprep/130142.html> )

And the computational paper from here: <https://github.com/tholden/dynareOBC/raw/master/ComputationalPaper.pdf>  
(Or here for citations: <https://ideas.repec.org/p/zbw/esprep/130143.html> )

A paper documenting the estimation procedure is available from here: <https://github.com/tholden/dynareOBC/raw/master/EstimationPaper.pdf>  
(Or here for citations: <http://dx.doi.org/10.5281/zenodo.50127> )

The code itself may be cited via the following DOI: <http://dx.doi.org/10.5281/zenodo.50132>

Alternatively, read the presentation slides from here: <https://github.com/tholden/dynareOBC/raw/master/slides.pdf>

Or the course slides from here: <https://github.com/tholden/dynareOBC/raw/master/slidesCourse.pdf>

### 1.2 Installation

1. Download the latest release from: <https://github.com/tholden/dynareOBC/releases>
2. Extract the release to a directory on a local drive.
3. Make sure your MATLAB path does not contain any Dynare folders other than the `matlab` sub-folder of your Dynare install. You should never click “add with subfolders” when adding Dynare to your MATLAB path, else Dynare will perform poorly, and DynareOBC will not work at all.
4. Add just the folder containing `dynareOBC.m` to your MATLAB path. Do not click with add-with subfolders!
5. Relax, as DynareOBC updates itself and its dependencies each time it is run.

### 1.3 Requirements

Requirements (to be installed and added to your Matlab path):

- Matlab version R2013a or later, or a fully compatible clone. Note that while DynareOBC should work on all platforms, it has been most heavily tested on 64-bit Windows, so if possible we suggest you use this platform.
- Dynare, version 4.4 or later, from: <http://www.dynare.org/download/dynare-stable>

- If on Windows, either administrative rights on the computer on which you are installing DynareOBC, so that DynareOBC can automatically install its other dependencies, or the manual installation of the items listed in the *Additional requirements for people using Windows, without administrative rights* section below.

Recommended additional installations:

- MATLAB R2016a or later.
- The MATLAB Parallel Toolbox, or a fully compatible clone, which speeds up assorted routines.
- The MATLAB Optimization Toolbox, or an alternative non-linear least squares routine, which is required for the experimental `global` option. (To use an alternative routine, you must set `dynareOBC.FSolveFunctor`.) This toolbox is also required for some options of estimation, as detailed below.
- The MATLAB Symbolic Toolbox, which is required for the UseVPA option.
- A working compiler for MEX which is supported by MATLAB Coder, ideally supporting OpenMP. On Windows, a free compiler meeting these requirements is available from: <https://www.visualstudio.com/en-us/news/vs2013-community-vs.aspx>. Alternatively, on Windows, with MATLAB r2015b, another free compiler meeting these requirements (which uses much less disk space) is available by clicking on “Add-Ons” in the MATLAB toolbar, then searching for MinGW. Be sure to untick the “check for updated files” in the installer that opens.
- MATLAB Coder, or a fully compatible clone (only used with MATLAB R2015a or later).
- A competitive mixed integer linear programming solver, such as one of the below (listed in rough order of performance), each of which is available for free to academics:
  - GUROBI Optimizer, from <http://www.gurobi.com/academia/for-universities>
  - IBM CPLEX, by following the instructions here: [https://www.ibm.com/developerworks/community/blogs/jfp/entry/cplex\\_studio\\_in\\_ibm\\_academic\\_initiative?lang=en](https://www.ibm.com/developerworks/community/blogs/jfp/entry/cplex_studio_in_ibm_academic_initiative?lang=en)
  - FICO Xpress, from <https://community.fico.com/download.jspa>
  - MOSEK, from <https://mosek.com/resources/academic-license> and <https://mosek.com/resources/downloads>

## 1.4 Additional requirements for people using Windows, without administrative rights

If you are using Windows, and you do not have administrative rights on your own computer, you need to ask your system administrator to download and install the following standard redistributable packages, which provide system libraries used by code compiled with either Microsoft Visual C++, the Intel C Compiler, or the Intel Fortran Compiler.

If you are using a 64-bit version of MATLAB, please ask for the following to be installed:

- [https://download.microsoft.com/download/4/2/F/42FF78CE-8DE0-4C88-AD7A-5F8DFFB49F74/vc\\_redist.x64.exe](https://download.microsoft.com/download/4/2/F/42FF78CE-8DE0-4C88-AD7A-5F8DFFB49F74/vc_redist.x64.exe)
- `ww_ifort_redist_intel64_2016.3.207.msi` from inside: [https://software.intel.com/sites/default/files/managed/46/54/ww\\_ifort\\_redist\\_msi\\_2016.3.207.zip](https://software.intel.com/sites/default/files/managed/46/54/ww_ifort_redist_msi_2016.3.207.zip)

If you are using a 32-bit version of MATLAB, please ask for the following to be installed:

- [http://download.microsoft.com/download/2/E/6/2E61CFA4-993B-4DD4-91DA-3737CD5CD6E3/vcredist\\_x86.exe](http://download.microsoft.com/download/2/E/6/2E61CFA4-993B-4DD4-91DA-3737CD5CD6E3/vcredist_x86.exe)
- [https://download.microsoft.com/download/9/3/F/93FCF1E7-E6A4-478B-96E7-D4B285925B00/vc\\_redist\\_x86.exe](https://download.microsoft.com/download/9/3/F/93FCF1E7-E6A4-478B-96E7-D4B285925B00/vc_redist_x86.exe)
- `w_fcompxe_redist_ia32_2015.2.179.msi` from inside: [https://software.intel.com/sites/default/files/managed/6a/21/w\\_fcompxe\\_redist\\_msi\\_2015.2.179.zip](https://software.intel.com/sites/default/files/managed/6a/21/w_fcompxe_redist_msi_2015.2.179.zip)

## 1.5 Troubleshooting

If you have any strange errors, first try these steps:

1. Delete all `.mat` files from the DynareOBC directory.

2. Delete all `.mex???` files from the `Core` sub-directory of your DynareOBC install (but not from its sub-directories).
3. Delete your DynareOBC directory, and then go through the installation steps above again.

If after doing this, you suspect a bug, report it here: <https://github.com/tholden/dynareOBC/issues>

## 1.6 Basic Usage

**Usage:** `dynareOBC FILENAME[.mod, .dyn] [OPTIONS]`

DynareOBC executes instruction included in the conventional dynare mod file, `FILENAME.mod`.

Unlike dynare, DynareOBC can handle simulation of models containing non-differentiable functions.

Note:

- DynareOBC approximates the non-differentiable function in levels. Thus if `r` and `pi` are the endogenous variables of interest: `r = max( 0, 0.005 + 1.5 * pi );` will be more accurate than: `exp(r) = max( 1, exp( 0.005 + 1.5 * pi ) );`.
- DynareOBC may produce strange results on models with an indeterminate steady-state, so caution should be taken when using the `STEADY_STATE` command. The `initval` or `steady_state_model` blocks should not be used to attempt to pin down a steady-state, since these will be ignored by DynareOBC in later steps of its solution procedure.
- DynareOBC defines the preprocessor constant `dynareOBC` during execution.

**OPTIONS (NOT CASE SENSITIVE!)** include:

- **For controlling the inner solution procedure**
  - `TimeToSolveParametrically=INTEGER` (default: 4)  
If the simulation is at the bound for at most this number of periods, then a pre-computed solution will be used, increasing the speed of long simulations, or those involving integration.
  - `TimeToEscapeBounds=INTEGER` (default: 32)  
The number of periods following a shock after which the model is expected to be away from any occasionally binding constraints. Note that when a global solution is requested, this value is ignored, and the maximum of the requested number of IRF periods, and the value of `TimeToReturnToSteadyState`, is used instead.
  - `TimeToReturnToSteadyState=INTEGER` (default: 64)  
The number of periods in which to verify that the constraints are not being violated. If this is lower than `TimeToEscapeBounds`, or the requested number of IRF periods, or `PeriodsOfUncertainty + 1`, then that value will be used instead.
  - `Omega=FLOAT` (default: 1000)  
The tightness of the constraint on the news shocks. If this is large, solutions with news shocks close to zero will be returned when there are multiple solutions.
  - `MILPSolver=STRING`  
(default: `gurobi, cplex, xpress, mosek, scip, cbc, intlinprog, lpsolve, glpk, *`)  
DynareOBC uses YALMIP internally for solving a mixed integer linear programming problem. This option sets YALMIP's preferred solvers. To find out what solvers are available to you, run `dynareOBC TestSolvers`, and examine the list displayed by YALMIP.
  - `ReverseSearch`  
By default, DynareOBC finds a solution in which the last period at the bound is as soon as possible. This option makes DynareOBC find a solution in which the last period at the bound is as remote as possible, subject to being less than the longest horizon (i.e. `TimeToEscapeBounds`).
  - `FullHorizon`  
By default, DynareOBC finds a solution in which the last period at the bound is as soon as possible. This option makes DynareOBC just solve the bounds problem at the longest horizon (i.e. `TimeToEscapeBounds`).

- SkipFirstSolutions=INTEGER (default: 0)  
If this is greater than 0, then DynareOBC ignores the first INTEGER solutions it finds, unless no other solutions are found, in which case it takes the last found one. Thus, without ReverseSearch, this tends to find solutions at the bound for longer. With ReverseSearch, this tends to find solutions at the bound for less time.
- IgnoreBoundFailures  
Makes DynareOBC attempt to continue even after it has failed to solve the bounds problem due to e.g. infeasibility. This will severely compromise accuracy.
- **For controlling cubature**
  - PeriodsOfUncertainty=INTEGER (default: 16)  
Controls the number of periods of uncertainty over which we integrate. Since a cosine windowing function is used, the effective number of periods of uncertainty is roughly half this number.
  - FastCubature  
Causes DynareOBC to ignore the value specified in MaxCubatureDegree and QuasiMonteCarlo, and to instead use a degree 3 monomial cubature rule without negative weights, but involving evaluations further from the origin.
  - QuasiMonteCarloLevel=INTEGER (default: 0)  
If this is non-zero, then Gaussian cubature is not used (so the MaxCubatureDegree option is ignored). Instead, quasi-Monte Carlo integration with at most  $2^{(1+INTEGER)} - 1$  samples is used.
  - MaxCubatureDegree=INTEGER (default: 7)  
Specifies the degree of polynomial which will be integrated exactly in the highest degree, cubature performed. Values above 51 are treated as equal to 51. Note that setting CubatureSmoothing>0 will reduce the effective maximum accuracy by 2 degrees. Setting CubatureTolerance>0 may also mean that the result does not integrate the stated degree polynomials exactly as well.
    - \* CubatureSmoothing=FLOAT\_IN\_UNIT\_INTERVAL (default: 0)  
When this is larger than 0, and less than 1, DynareOBC takes a weighted combination of the results of cubature rules of adjacent degrees. Large numbers imply larger weights on lower degree rules. A good setting is often 0.01 - 0.3.
  - CubaturePruningCutOff=FLOAT (default: 0.01)  
Eigenvalues of the covariance matrix of the distribution from which we integrate that are below FLOAT times the maximum eigenvalue are “pruned” to zero, in order to increase integration speed.
  - MaxCubatureDimension=INTEGER (default: 128)  
The maximum dimension over which to integrate. If the algorithm needs to integrate over a larger space, it will “prune” all but the INTEGER largest eigenvalues of the covariance matrix to zero.
  - CubatureTolerance=FLOAT (default: 1e-6)  
Specifies that the maximum acceptable change in the integrals is the given value, for quasi Monte Carlo or default cubature. Setting this to zero disables adaptive cubature.
  - NoCubature  
Speeds up DynareOBC by assuming that agents are “surprised” by the existence of the bound. At order=1, this is equivalent to a perfect foresight solution to the model.
- **For controlling accuracy**
  - FirstOrderAroundRSS  
Takes a linear approximation around the risky steady state of the non-linear model. If specifying this option, you should set order=2 or order=3 in your mod file.
  - FirstOrderAroundMean  
Takes a linear approximation around the ergodic mean of the non-linear model. If specifying this option, you should set order=2 or order=3 in your mod file.
  - FirstOrderConditionalCovariance  
When order>1 (possibly with FirstOrderAroundRSS or FirstOrderAroundMean), by default, DynareOBC uses a second order approximation of the conditional covariance to determine the space of paths to integrate over. This option specifies that a first order approximation should be used instead.
  - MLVSimulationMode=0|1|2|3 (default: 0)  
If MLVSimulationMode=0, DynareOBC does not attempt to simulate the path of model local variables.

If `MLVSimulationMode>0`, DynareOBC generates simulated paths and average impulse responses for each model local variable (MLV) which is used in the model, non-constant, non-forward looking, and not purely backwards looking. Note that to generate impulse responses, you must enable the `SlowIRFs` option.

If `MLVSimulationMode>1`, DynareOBC additionally generates simulated paths and average impulse responses for each non-constant MLV, used in the model, containing forward looking terms.

If `MLVSimulationMode=2`, then DynareOBC takes the expectation of each forward looking MLV using sparse cubature.

If `MLVSimulationMode=3`, then DynareOBC takes the expectation of each forward looking MLV using quasi-Monte Carlo integration.

- \* `MLVSimulationAccuracy=INTEGER` (default: 9)

When `MLVSimulationMode=2`, this specifies the degree of polynomial which should be integrated exactly. In this case, values above 51 are treated as equal to 51. When `MLVSimulationMode=3`,  $2^{(1+INTEGER)} - 1$  is the number of points used for quasi-Monte Carlo integration.

- \* `MLVSimulationSubSample=INTEGER` (default: 1)

Causes DynareOBC to only calculate the value of MLVs every `INTEGER` samples. Setting this greater than 1 is useful when calculating MLVs is expensive, and you want to reduce the standard error of simulated moments.

- `Sparse`

Causes DynareOBC to replace all of the elements of the decision rules by sparse matrices, which may speed up DynareOBC, at the cost of some slight reduction in accuracy for certain models with small coefficients.

- **For controlling and performing model diagnostics**

- `FeasibilityTestGridSize=INTEGER` (default: 0)

Specifies the number of points in each of the two axes of the grid on which a test of a sufficient condition for feasibility is performed. Setting a larger number increases the chance of finding feasibility, but may be slow.

If `FeasibilityTestGridSize=0` then the test is disabled.

- `PTest=INTEGER` (default: 0)

Runs a fast as possible test to see if the top `INTEGERxINTEGER` submatrix of `M` is a `P` matrix. Set this to 0 to disable these tests.

- `AltPTest=INTEGER` (default: 0)

Uses a slower, more verbose procedure to test if the top `INTEGERxINTEGER` submatrix of `M` is a `P` matrix. Set this to 0 to disable these tests.

- `FullTest=INTEGER` (default: 0)

Runs very slow tests to see if the top `INTEGERxINTEGER` submatrix of `M` is a `P(0)` and/or (strictly) semi-monotone matrix.

- `UseVPA`

Enables more accurate evaluation of determinants using the symbolic toolbox.

- `LPSolver=STRING`

(default: `xpress, gurobi, cplex, mosek, clp, scip, linprog, glpk, lpsolve, cdd, qsopt, *`)  
Specifies the preferred solver to use for the linear programming problem that is solved when checking whether matrices are `S/S_0`. To find out what solvers are available to you, run `dynareOBC TestSolvers`, and examine the list displayed by `YALMIP`.

- **For controlling IRFs**

- `SlowIRFs`

Calculates a more accurate approximation to expected IRFs using Monte-Carlo simulation. Without this option, DynareOBC calculates expected IRFs via cubature (unless this is also disabled).

- `IRFsAroundZero`

By default, IRFs are centred around the risky steady state without the `SlowIRFs` option, or around the approximate mean with it. This option instead centres IRFs around 0.

- `ShockScale=FLOAT` (default: 1)

Scale of shocks for IRFs. This allows the calculation of IRFs to shocks larger or smaller than one standard deviation.

- **Settings for controlling estimation or smoothing**

- Estimation

Enables estimation of the model's parameters

- \* `DataFile=STRING` (default: `MOD-FILE-NAME.xlsx`)  
Specifies the spreadsheet containing the data to estimate. This spreadsheet should contain two worksheets. The first sheet should have a title row containing the names of the MLVs being observed, followed by one row per observation. There should not be a column with dates. The second sheet should contain a title row with the names of the parameters being estimated, followed by one row for their minima (with empty cells being interpreted as minus infinity), then by one row for their maxima (with empty cells being interpreted as plus infinity).
- \* `Prior=STRING` (default: `FlatPrior`)  
Specifies the function containing the prior to be used in maximum a posteriori estimation. The default prior results in maximum likelihood estimates being returned. The function should accept a single argument giving the vector of parameters to be estimated, in the order they appear in the datafile, including the measurement error variances in the final elements of the vector. The function should return the log prior density at that point (up to a constant).
- \* `StationaryDistMaxIterations=INTEGER` (default: 1000)  
The maximum number of iterations used to evaluate the stationary distribution of the non-linear filter.
- \* `SkipStandardErrors`  
Makes DynareOBC skip calculation of standard errors for the estimated parameters.
- \* `PredictSparseCubatureDegree=INTEGER` (default: 0)  
If this is greater than zero, then DynareOBC uses an alternative sparse cubature rule including additional points for integrating over the states and shocks of the model in the predict step. While this requires solving the model less far from the steady-state when the state dimension is large, it also requires negative weights, which may cause numerical issues e.g. with the positive definiteness of the state covariance matrix. The cubature method exactly integrates a polynomial of degree `INTEGER`. Thus, in a model without bounds, there is no need to have `INTEGER` larger than twice the order of approximation. Values above 51 are treated as equal to 51.
- \* `UpdateSparseCubatureDegree=INTEGER` (default: 0)  
If this is greater than zero, then DynareOBC uses an alternative sparse cubature rule including additional points for integrating over the state of the model in the update step. While this requires evaluating the measurement equations less far from the steady-state when the state dimension is large, it also requires negative weights, which may cause numerical issues e.g. with the positive definiteness of the state covariance matrix. The cubature method exactly integrates a polynomial of degree `INTEGER`. Thus, in a model with measurement equations that are a polynomial of degree `d`, there is no need to have `INTEGER` larger than two times `d`. Values above 51 are treated as equal to 51.
- \* `StdDevThreshold=FLOAT` (default: `1e-6`)  
Specifies the threshold below which the standard deviation of the state is set to zero, for dimension reduction.
- \* `MinimisationFunctions=STRING`  
(default: `CMAESWrapper, FMinConWrapper`)  
`A` , ; or # delimited list of minimisation function names, which will be invoked in order. DynareOBC includes the following: `CMAESWrapper` (an evolutionary global search algorithm), `FMinConWrapper` (MATLAB's local search, which requires a license for the MATLAB Optimisation Toolbox), `FMinBndWrapper` (performs repeated one dimensional search, only viable for very low dimensional problems).
- \* `FixedParameters=STRING` (default: `' '`)  
`A` , ; or # delimited list of parameters names. Any parameters in this list will not be estimated, even if they occur in the second sheet of the data file.
- \* `TimeOutLikelihoodEvaluation=INTEGER` (default: 60)  
Any likelihood evaluations that take longer than this number of seconds will be terminated prematurely. Beware that this may bias parameter estimates.

- Smoothing

Performs smoothing to estimate the model's state variables and shocks. Enabling this option disables estimation, so smoothing should be invoked in a separate DynareOBC run after estimation has completed.

\* `DataFile=STRING` (default: `MOD-FILE-NAME.xlsx`)

Specifies the spreadsheet containing the data to estimate. This spreadsheet should contain at least one worksheet. The first sheet should have a title row containing the names of the MLVs being observed, followed by one row per observation. There should not be a column with dates.

\* `StationaryDistMaxIterations=INTEGER` (default: 1000)

The maximum number of iterations used to evaluate the stationary distribution of the non-linear filter.

\* `PredictSparseCubatureDegree=INTEGER` (default: 0)

If this is greater than zero, then DynareOBC uses an alternative sparse cubature rule including additional points for integrating over the states and shocks of the model in the predict step. While this requires solving the model less far from the steady-state when the state dimension is large, it also requires negative weights, which may cause numerical issues e.g. with the positive definiteness of the state covariance matrix. The cubature method exactly integrates a polynomial of degree `INTEGER`. Thus, in a model without bounds, there is no need to have `INTEGER` larger than twice the order of approximation. Values above 51 are treated as equal to 51.

\* `UpdateSparseCubatureDegree=INTEGER` (default: 0)

If this is greater than zero, then DynareOBC uses an alternative sparse cubature rule including additional points for integrating over the state of the model in the update step. While this requires evaluating the measurement equations less far from the steady-state when the state dimension is large, it also requires negative weights, which may cause numerical issues e.g. with the positive definiteness of the state covariance matrix. The cubature method exactly integrates a polynomial of degree `INTEGER`. Thus, in a model with measurement equations that are a polynomial of degree `d`, there is no need to have `INTEGER` larger than two times `d`. Values above 51 are treated as equal to 51.

\* `StdDevThreshold=FLOAT` (default: `1e-6`)

Specifies the threshold below which the standard deviation of the state is set to zero, for dimension reduction.

- **EXPERIMENTAL settings for controlling accuracy**

- `Global`

Without this, DynareOBC assumes agents realise that shocks may arrive in the near future which push them towards the bound, but they do not take into account the risk of hitting the bound in the far future. With the global option, DynareOBC assumes agents take into account the risk of hitting the bound at all horizons. Note that under the global solution algorithm, dotted lines give the responses with the polynomial approximation to the bound. They are not the response ignoring the bound entirely. Requires the MATLAB Optimization toolbox, or an alternative non-linear least squares routine, see above for details.

- \* `Resume`

Resumes an interrupted solution iteration, when using global.

- **Advanced options**

- `Bypass`

Ignores all non-differentiabilities in the model. Useful for debugging.

- `DisplayBoundsSolutionProgress`

Displays progress solving the bounds problem. Only useful for very hard to solve models under perfect foresight.

- `CompileSimulationCode`

Compiles the code used for simulating the base model, without the bound. May speed up long simulations.

- `NoCleanup`

Prevents the deletion of DynareOBC's temporary files. Useful for debugging.

- `OrderOverride=1|2|3`

Overrides the order of approximation set within the call to `stoch_simul`.

- `ShockSequenceFile=FILENAME.mat`

Specifies a MAT file to open to load shocks from. The MAT file should contain a matrix called `Shock-`

- Sequence, which should have as many rows as there are shocks, and as many columns as there are simulation periods.
- `SimulateOnGridPoints`  
Rather than running an actual simulation, causes DynareOBC to draw QMC points from a Gaussian approximation to the stationary distribution of the model without bounds. The mean for the Gaussian approximation is accurate to the same order as the order of approximation, but the variance used is always only accurate to a first order approximation.

See the Dynare reference manual for other available options.

## 1.7 Supported options inside the .mod file

Note that DynareOBC only supports some of the options of `stoch_simul`, and no warning is generated if it is used with an unsupported option. Currently supported options for `stoch_simul` are:

- `irf=INTEGER`
- `periods=INTEGER`
- `drop=INTEGER`
- `order=1|2|3`
- `replic=INTEGER`
- `loglinear`
- `irf_shocks`
- `nograph`
- `nodisplay`
- `nomoments`
- `nocorr`

DynareOBC also supports a list of variables for simulation after the call to `stoch_simul`. When `MLVSimulationMode>0`, this list can include the names of model local variables. Any MLV included in this list will be simulated even if it does not meet the previous criteria.

## 1.8 Advanced Usage

Alternative usage: `dynareOBC [AddPath|RmPath|TestSolvers] [OPTIONS]`

- `AddPath`  
Adds all folders used by DynareOBC to the path, useful for debugging and testing.
- `RmPath`  
Removes all folders used by DynareOBC from the path, useful for cleanup following a crash.
- `TestSolvers`  
Tests the installed solvers.

## 1.9 Acknowledgements and copyright information

DynareOBC incorporates code:

- for calculating risky first order approximations that is copyright Meyer-Gohde, 2014.  
More information is contained in his paper describing the algorithm, available here: [http://enim.wiwi.hu-berlin.de/vwl/wtm2/mitarbeiter/meyer-gohde/stochss\\_main.pdf](http://enim.wiwi.hu-berlin.de/vwl/wtm2/mitarbeiter/meyer-gohde/stochss_main.pdf)
- from the nonlinear moving average toolkit that is copyright Lan and Alexander Meyer-Gohde, 2014,
- for nested Gaussian cubature that is copyright Genz and Keister, 1996,
- for displaying a progress bar that is copyright Cacho, "Stefan" and Scheff, 2014,
- for (mixed-integer) linear programming, from GLPKMEX, copyright Makhorin, Legat and others, 2015,
- for calculating pseudo-spectral radii, from EigTool, copyright Wright, Mengi, Overton and colleagues, 2014.



Additionally, DynareOBC automatically downloads:

- YALMIP, copyright Lofberg, 2015,
- the Opti Toolbox, copyright Currie, and others, 2015,
- and MPT, with its dependencies, copyright Herceg and others, 2015.

The original portions of DynareOBC are copyright (c) Tom Holden, 2016.

DynareOBC is released under the GNU GPL, version 3.0, available from <https://www.gnu.org/copyleft/gpl.html>

DynareOBC is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or(at your option) any later version.

DynareOBC 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 General Public License for more details.

You should have received a copy of the GNU General Public License along with dynareOBC. If not, see <http://www.gnu.org/licenses/>.