An update on Dynare

New features and future plans

Sébastien Villemot

Dynare

February 15, 2023

Timeline of major releases

2008 Version 4.0

2009 Version 4.1

2011 Version 4.2

2012 Version 4.3

2013 Version 4.4

2017 Version 4.5

2020 Version 4.6

2022 Version 5

2024 Version 6 (still under development)

Features only available in the *unstable* version (to become version 6) henceforth marked with this symbol: ❻

Outline

Outline

Higher-order solution and simulation

- Solution under perturbation now available at arbitrary Taylor approximation order

Example: 4th order approximation

```
stoch_simul(order=4);
```

- Of course, subject to computational limits. Internally uses Dynare++ engine (in C++); soon a Fortran rewrite with performance improvements ❻
- Optional "pruning" to avoid explosive simulation trajectories
  - Up to 3rd order
  - Theoretical moments available
  - Soon at arbitrary approximation order ❻

Example: 3rd order approximation with pruning

```
stoch_simul(order=3, pruning);
```

Nonlinear Bayesian estimation

- Possibility to estimate models approximated at an arbitrary order
- Necessitates a particle or nonlinear filter. Available filters:
  - Sequential importance sampling (default)
  - Auxiliary particle filter
  - Gaussian filter
  - Gaussian mixture filter
  - Conditional particle filter
  - Nonlinear Kalman filter

Example: particle filtering at 2nd order (expliciting some default option values)

```
estimation(datafile='mydata.xlsx', order=2, filter_algorithm=sis,
           number_of_particles=5000, resampling=systematic);
```

Heteroskedastic filter

- Filter where standard error of shocks may *unexpectedly* change in every period
- Standard errors may be set/modified in each observed period by either a scale factor or a provided value

Example

```
shocks;
  var e1; stderr 0.014;
  var e2; stderr 0.005;
end;
...
heteroskedastic_shocks;
  var e1;
  periods 86:87, 88, 89:97;
  scales 0.5, 0.1, 0;

  var e2;
  periods 86:87 88:97;
  values 0.04 0.01;
end;

estimation(order=1, datafile='mydata.xlsx', heteroskedastic_filter);
```

Method of moments (1/2)

Generalized method of moments (GMM)

Example: GMM at 2nd order (with pruning)

```
matched_moments;
  c;
  y;
  c*c;
  c*y;
  y^2;
  c*c(3);
end;

method_of_moments(mom_method=GMM, datafile='mydata.xlsx', order=2);
```

Available up to 3rd approximation order, only with pruning. Can only match 1st and 2nd moments.

Method of moments (2/2)

Simulated method of moments (SMM)

Example: SMM at 4th order (without pruning)

```
matched_moments;
  y;
  c*y;
  y^2;
  c*c(3);
  y(1)^2*c(-4)^3;
  c(-5)^3*y(0)^2;
end;

method_of_moments(mom_method=SMM, datafile='mydata.xlsx', order=4,
                  burnin=300);
```

Available at any approximation order, with or without pruning. Can match any moment.

Identification

- Identification analysis has been available since v4.3, based on moments (Iskrev, 2010)
- New identification check based on spectral density (Qu and Tkachenko, 2012)
- New identification check based on minimal system (Komunjer and Ng, 2011)
- Identification now also available for approximation orders 2 and 3, with either analytical or numerical parameter derivatives
- New options for disabling individual tests

Example

```
identification(order=2, advanced=1, no_identification_strength);
```

Outline

Syntax change

Old syntax

```
simul(periods=200, stack_solve_algo=1);
```

New syntax

```
perfect_foresight_setup(periods=200);
perfect_foresight_solver(stack_solve_algo=1);
```

- More meaningful names
- Facilitates customization of problem constraints or guess values

Dynamic homotopy

- If `perfect_foresight_solver` fails to find a solution, it automatically switches to a homotopy technique
- Idea: achieve convergence on smaller shock size, then use the result as initial guess for bigger shock size (divide-and-conquer strategy)
- Works with both temporary and permanent shocks (*i.e.* `shocks` and `endval`)
- Can be combined with any deterministic solver
- Can be disabled with option `no_homotopy`

Perfect foresight with expectation errors (1/2) ❻

With a perfect foresight solver:

- shocks are unexpected in period 1
- but in subsequent periods they are fully anticipated

How to simulate an unexpected shock at a period $t > 1$?

- Do a perfect foresight simulation from periods 0 to T *without the shock*
- Do another perfect foresight simulation from periods t to T
  - applying the shock in t,
  - and using the results of the first simulation as initial condition
- Combine the two simulations:
  - use the first one for periods 1 to $t - 1$,
  - and the second one for t to T

Perfect foresight with expectation errors (2/2) ❻

Example

```
// Declare pre-announced shocks
shocks(learnt_in=1);
  var epsilon;
  periods 5, 15;
  values -0.1, -0.1;
end;

// Declare shocks learnt in period 10
shocks(learnt_in=10);
  var epsilon;
  periods 10;
  values 0.1;
end;

perfect_foresight_with_expectation_errors_setup(periods=300);
perfect_foresight_with_expectation_errors_solver;
```

- For terminal conditions, use: `endval(learnt_in=...);`
- Alternatively, `datafile` option to provide all the information sets in CSV file

Outline

OccBin (1/3)

- Piecewise linear approach of Guerrieri and Iacoviello (JME, 2015)
- Under certainty equivalence; but quite fast, works on large models

Example

```
model;
  [name='Notional rate Taylor rule']
  i_not=rho*i_not(-1)+rho*(phi_pi*pie+phi_y*y)+zeps_i;
  [name='Observed interest rate', relax='zlb']
  i = i_not;
  [name='Observed interest rate', bind='zlb']
  i = i_elb;
...
end;
```

OccBin (2/3)

Example (cont'd)

```
occbin_constraints;
  name 'zlb'; bind i_not <= i_elb;
end;

shocks(surprise);
  var zeps_i;
  periods 1 2;
  values -0.01 -0.02;
end;

occbin_setup;
occbin_solver(simul_periods=20, simul_check_ahead_periods=50);
occbin_graph y i i_not pie;
```

OccBin (3/3)

Estimation possible with either:

- Piecewise Kalman Filter from Giovannini, Pfeiffer and Ratto (2022)

PKF example

```
occbin_setup(likelihood_piecewise_kalman_filter,
             smoother_piecewise_kalman_filter);
estimation(datafile='mydata.xlsx', mh_replic=0, smoother);
```

- Inversion Filter from Guerrieri and Iacoviello (JME, 2017)
  Caveat: requires exactly as many shocks as observables

IF example

```
occbin_setup(likelihood_inversion_filter, smoother_inversion_filter);
estimation(datafile='mydata.xlsx', mh_replic=0, smoother);
```

Mixed-complementarity problems (1/2)

Euler equation of neoclassical growth model with irreversible investment ($i_t \geq 0$):

$$c_t^{-\tau} - \mu_t = \beta\,\mathbb{E}_t\left[c_{t+1}^{-\tau}\left(\alpha A_{t+1}k_t^{\alpha-1} + 1 - \delta\right) - \mu_{t+1}(1-\delta)\right]$$

Slackness condition:

$$\mu_t = 0 \;\; \text{and} \;\; i_t \geq 0$$

or

$$\mu_t > 0 \;\; \text{and} \;\; i_t = 0$$

where $\mu_t \geq 0$ is the Lagrange multiplier associated to the non-negativity constraint for investment

Mixed-complementarity problems (2/2)

Example: MCP solution under perfect foresight

```
model;
  c^(-tau) - mu = beta*(c(+1)^(-tau)
      *(alpha*a(+1)*k^(alpha-1)+1-delta)-mu(+1)*(1-delta));
...
  [ mcp = 'i > 0' ]
  mu = 0;
end;
...
perfect_foresight_setup(periods=400);
perfect_foresight_solver(lmmcp, maxit=200);
```

Outline

Syntax change for optimal policy with commitment

Old syntax

```
ramsey_policy(planner_discount = beta, instruments = (i), order = 2);
```

New syntax

```
ramsey_model(planner_discount = beta, instruments = (i));
stoch_simul(order=2);
evaluate_planner_objective;
```

Estimation now possible

Example: estimation under optimal policy with commitment

```
ramsey_model(planner_discount = beta, instruments = (i));
estimation(datafile='mydata.xlsx');
```

Example: estimation under discretionary optimal policy

```
discretionary_policy(planner_discount = beta, instruments = (i));
estimation(datafile='mydata.xlsx');
```

Caveat: it's not (yet) possible to estimate the discount factor of the social planner

Welfare computation

The `evaluate_planner_objective` command:

- Returns unconditional (*i.e.* long-run) welfare, in addition to conditional welfare (*i.e.* specific to initial conditions)
- Available for any approximation order under perturbation
- Also available in perfect foresight context

Outline

VAR expectations

Example: expectation based on linear combination of a VAR(2)

```
var_model(model_name = var3eqs, eqtags = [ 'X' 'Y' 'Z' ]);

var_expectation_model(model_name = varexp, expression = 0.2*x + 0.3*y,
                      auxiliary_model_name = var3eqs, horizon = 2, discount = beta);
model;
  [ name = 'X' ]
  x = a*x(-1) + b*x(-2) + c*z(-2) + e_x;
  [ name = 'Y' ]
  y = d*y(-2) + e*z(-1) + e_y;
  [ name = 'Z' ]
  z = f*z(-1) + e_z;
...
  foo = .5*foo(-1) + var_expectation(varexp);
end;

var_expectation.initialize('varexp');
var_expectation.update('varexp');
...
perfect_foresight_setup(periods=100);
perfect_foresight_solver(solve_algo=14);
```

Polynomial adjustment costs (PAC) equation (1/3)

- Equation of the form:

$$\Delta y_t = a_0(y^*_{t-1} - y_{t-1}) + \sum_{i=1}^{m-1} a_i \Delta y_{t-i} + \mathbb{E}_t \sum_{i=0}^{\infty} d_i \Delta y^*_{t+i} + \varepsilon_t$$

  where y^*_t is the long-run target
- Can be derived from the minimization of a quadratic cost function penalising expected deviations from the target and non-smoothness of y
- Expectation term may be either VAR-based or model-consistent
- Used extensively in FRB/US and ECB/BASE
- Can be extended with growth neutrality correction term and exogenous terms

PAC equation (2/3)

Example with VAR-based expectations

```
trend_component_model(model_name=vecm, eqtags=['eq:x1', 'eq:x2', 'eq:x1bar', 'eq:x2bar'],
                      targets=['eq:x1bar', 'eq:x2bar']);

pac_model(auxiliary_model_name=vecm, discount=beta, growth=diff(x1(-1)), model_name=pacmod);

model;
  [name='eq:x1']
  diff(x1) = a10*(x1(-1)-x1bar(-1)) + a11*diff(x1(-1)) + a12*diff(x2(-1)) + ex1;
  [name='eq:x2']
  diff(x2) = a20*(x2(-1)-x2bar(-1)) + a21*diff(x1(-1)) + a22*diff(x1(-2)) + ex2;
  [name='eq:x1bar']
  x1bar = x1bar(-1) + ex1bar;
  [name='eq:x2bar']
  x2bar = x2bar(-1) + ex2bar;

  diff(z) = e_c_m*(x1(-1)-z(-1)) + c_z*diff(z(-1)) + pac_expectation(pacmod) + ez;
end;

pac.initialize('pacman');
pac.update.expectation('pacman');
```

PAC equation (3/3)

- Model-consistent solution obtained by removing auxiliary model (and option `auxiliary_model_name` of `pac_model`)
- Estimation of PAC equation possible with:
  - Nonlinear least squares
  - Iterative ordinary least squares

  Caveat: estimation of the whole model not available

Outline

On-the-fly variable declarations

With equation tags (only for endogenous)

```
varexo e;
parameters rho ybar;

model;
  [endogenous='y']
  y = rho*y(-1) + (1-rho)*ybar + e;
...
end;
```

With suffixes (à la TROLL)

```
model;
  y|e = rho|p*y(-1) + (1-rho)*ybar|p + e|x;
...
end;
```

Macro-processor extensions

- New object types: real (supersedes integers), boolean (distinct from integers), tuple
- New operators: set operations on arrays (union, intersection, difference, cartesian product and power), various mathematical functions
- Support for comprehensions, *e.g.*: `[ i^2 for i in 1:5 when mod(i,2) == 0 ]`
- User-defined functions can be defined, *e.g.*: `@#define f(x) = 2*x^2+3*x+5`
- Iterate over several variables at the same time, *e.g.*: `@#for (i,j) in X*Y` where `X` and `Y` are arrays
- Exclude some elements when iterating, *e.g.*: `@#for i in 1:5 when mod(i,2) == 0`
- `@#elseif` clauses supported in conditional statements

Automatic logarithmic variable transformation ❻

- If an endogenous is declared with: `var(log) y;`
  - Creates two endogenous `y` and `LOG_y`
  - Every occurrence of `y` is replaced by `exp(LOG_y)`
  - Adds an equation: `y = exp(LOG_y);`
- Useful for performing loglinear approximation of selected variable(s)
- Also useful to enforce positivity of `y`
  $\Rightarrow$ can help the nonlinear solver if `y` is used with `log` or `sqrt`

Model editing (1/2) ❻

Example: add/remove/replace equations

```
model_options(block, bytecode);

model;
  [name = 'resource']
  c + k = aa*x*k(-1)^alph + (1-delt)*k(-1);
end;
...
model;
  [name = 'euler']
  c^(-gam) = (1+bet)^(-1)*(aa*alph*x(+1)*k^(alph-1) + 1 - delt)*c(+1)^(-gam);
end;
...
model_remove('resource');

model_replace('euler');
  1/c = (1+bet)^(-1)*(aa*alph*x(+1)*k^(alph-1) + 1 - delt)/c(+1);
end
```

Model editing (2/2) ❻

Example: add/remove variables/parameters

```
var x y;
varexo u;
parameters alpha;
...
var z;
parameters beta;
...
var_remove y alpha;
```

Example: add/remove estimated parameters

```
estimated_params;
  alpha, normal_pdf, 1, 0.05;
end;
...
estimated_params;
  stderr y, uniform_pdf,,,0,1;
end;
...
estimated_params_remove;
  alpha;
  stderr y;
end;
```

Outline

Possible new features and improvements

- Performance improvements on (very) large models
- Heterogenous Agent New Keynesian (HANK) models
- Availability via MATLAB Online
- Graphical user interfaces
- More interactive model building (à la TROLL)
- . . . (your wishes here)

Reimplement on another platform?

- MATLAB
  - robust, good (and improving) performance, full-featured Dynare implementation
  - but expensive and slightly out of fashion
- Octave
  - free, full-featured Dynare implementation
  - but often slower than MATLAB (though mitigated by low-level routines, a.k.a. MEX files)
- Julia
  - free, modern and fast
  - but not widely adopted
  - early-stage experimental Dynare rewrite
  - remains to be seen whether benefits of rewrite outweigh costs
- Python
  - free, very popular, close to being the *lingua franca* of scientific computing
  - could be made fast if coupled with low-level code (Fortran or C++), as in MATLAB/Octave
  - nothing done yet on Dynare side

Advanced Dynare Users group/workshop

- For presenting, discussing, and sharing experiences regarding advanced features and expert use of Dynare
- Dedicated permanent online chat room (contact me for getting access)
- Annual 3-days workshop in Ispra (Italy)
  In 2023, early September or late November (TBC)

Thanks for participating!

My office: HQ1-10-151 (until Feb 24)

My email: `sebastien@dynare.org`