A short course on occasionally binding constraints and other non-linearities in DSGE models

Tom Holden

Motivation for the course

- Since the financial crisis, many central banks around the world have set their nominal interest rates near zero.
- Additionally, during the crisis many households, firms and banks were pushed up against their borrowing constraints.

- The zero lower bound on nominal interest rates and borrowing constraints are prominent examples of occasionally binding constraints (OBCs).

- OBCs generate significant non-linearies, which means that (log)-linear simulation and estimation methods are likely to perform poorly.

Course aims

- This course is designed to give a hands on introduction to the simulation and estimation of models with occasionally binding constraints and other non-linearities.

- We will chiefly be using the “DynareOBC” toolkit, which is a Dynare add-on.

- The course will also cover tools for assessing the properties (e.g. existence and uniqueness) of models with occasionally binding constraints.

- We will also discuss some of the relevant theoretical results as we go along.

Please…

- Play around with code while I’m talking.
- Try things out on your own models, as well as just those I provide.
- Interrupt me whenever things aren’t clear.

Course outline: Day 1

- Session 1 (1.5 hours)
  - DynareOBC set-up.
  - Basic utilisation of DynareOBC for perfect foresight simulation of models with occasionally binding constraints.
  - DynareOBC's core simulation algorithm.
  - Simulation of non-linear models via the extended path in DynareOBC.
- Session 2 (1.5 hours)
  - The stochastic extended path algorithm in the context of DynareOBC.
  - Cubature rules.
  - Using DynareOBC for approximate rational expectations solutions.
  - Some accuracy results.
  - Other DynareOBC options.
- Session 3 (1 hour)
  - Average IRFs.
  - The Dynare pre-processor.
  - Use of MLVs.
  - Use of the Dynare Transformation Engine.
  - Simulation of MLVs.

Course outline: Day 2

- Session 1 (1.5 hours)
  - Theoretical results on existence, uniqueness and efficient computability.
  - Testing model properties using DynareOBC.
  - Equilibrium selection.
  - Application to New Keynesian models.
- Session 2 (1.5 hours)
  - The cubature Kalman filter.
  - Estimating using the Cubature Kalman filter in DynareOBC.
  - Set-up of the Holden Unstable Dynare version.
  - Estimating using the Cubature Kalman filter in Dynare.
- Session 3 (1 hour)
  - The Kollman (2013) non-linear estimation approach.
  - Estimating using the Kollman approach in the Holden Unstable Dynare version.
  - The Meyer-Gohde (2014) non-linear estimation approach.
  - Estimating using the Meyer-Gohde approach in the Holden Unstable Dynare version.

Key readings on occasionally binding constraints

- Fair and Taylor (1983): *"Solution and Maximum Likelihood Estimation of Dynamic Nonlinear Rational Expectations Models"*
- Guerrieri and Iacoviello (2015): *"OccBin: A toolkit for solving dynamic models with occasionally binding constraints easily"*
- Adjemian and Juillard (2013, 2016): *"The stochastic extended path approach"*

- Holden (2016a): "*Existence and uniqueness of solutions to dynamic models with occasionally binding constraints"*
- Holden (2016b): *"Computation of solutions to dynamic models with occasionally binding constraints"*
- Holden (2016c): *"Estimation of dynamic models with occasionally binding constraints"*

Key readings on other aspects of non-linear simulation and estimation

- Lan and Meyer-Gohde (2013): *"Pruning in Perturbation DSGE Models - Guidance from Nonlinear Moving Average Approximations"*
- Meyer-Gohde (2014): *"Risky Linear Approximations"*

- Arasaratnam and Haykin (2009): *"Cubature Kalman Filters"*
- Kollmann (2013): *"Tractable Latent State Filtering for Non-Linear DSGE Models Using a Second-Order Approximation"*

Session outline: Day 1, Session 1 (1.5 hours)

- DynareOBC set-up.
- Basic utilisation of DynareOBC for perfect foresight simulation of models with occasionally binding constraints.
- DynareOBC's core simulation algorithm.
- Simulation of non-linear models via the extended path in DynareOBC.

Obtaining DynareOBC (1/2)

- DynareOBC is always available from:

  https://github.com/tholden/dynareOBC

- The recommended way to get it on your computer is:
  1. Download and install GitHub Desktop, from https://desktop.github.com/. (Note that this does not require admin rights!).
  2. Click on the “clone or download” button on the DynareOBC website.
  3. Click “open in desktop”.
- Downloading via GitHub Desktop means you can view the change history, and also contribute fixes.
- If you cannot install GitHub desktop due to machine restrictions, you may still be able to install PotableGit from here:

  https://github.com/git-for-windows/git/releases

  - Then, using the git command prompt, type:

```
git clone https://github.com/tholden/dynareOBC --recurse-submodules
```

Obtaining DynareOBC (2/2)

- Alternatively, you may just download a DynareOBC release from this page:

  https://github.com/tholden/dynareOBC/releases

  - DynareOBC should auto-update itself the first time it runs.

- Or, you can click “clone or download” then “download ZIP” on the DynareOBC homepage.
  - This does not give all dependencies, but DynareOBC should automatically download these dependencies the first time it runs.

DynareOBC installation

- To install, just add the root DynareOBC folder (i.e. the one containing “dynareOBC.m”) to your MATLAB path.

- Once you have done this, test DynareOBC by running (in MATLAB):

```
dynareOBC testsolvers
```

- This should produce output roughly as shown on the next slide.

- After running TestSolvers, it is a good idea to clean up the path using:

```
dynareOBC rmpath
```

Sample DynareOBC TestSolvers Output

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|                   Test|   Solution|                    Solver message|
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|   Core functionalities|        N/A|       Successfully solved (YALMIP)|
|                     LP|    Correct|    Successfully solved (CPLEX-IBM)|
|                     LP|    Correct|    Successfully solved (CPLEX-IBM)|
|                     QP|    Correct|    Successfully solved (CPLEX-IBM)|
|                     QP|    Correct|    Successfully solved (CPLEX-IBM)|
|                   SOCP|    Correct|    Successfully solved (CPLEX-IBM)|
|                   SOCP|    Correct|    Successfully solved (CPLEX-IBM)|
|                   SOCP|    Correct|    Successfully solved (CPLEX-IBM)|
|                    SDP|    Correct|   Successfully solved (SeDuMi-1.3)|
|                    SDP|    Correct|   Successfully solved (SeDuMi-1.3)|
|                    SDP|    Correct|   Successfully solved (SeDuMi-1.3)|
|                    SDP|    Correct|   Successfully solved (SeDuMi-1.3)|
|                 MAXDET|    Correct|   Successfully solved (SeDuMi-1.3)|
|                 MAXDET|    Correct|   Successfully solved (SeDuMi-1.3)|
|          Infeasible LP|        N/A|     Infeasible problem (CPLEX-IBM)|
|          Infeasible QP|        N/A|     Infeasible problem (CPLEX-IBM)|
|         Infeasible SDP|        N/A|    Infeasible problem (SeDuMi-1.3)|
|      Moment relaxation|    Correct|   Successfully solved (SeDuMi-1.3)|
|         Sum-of-squares|    Correct|   Successfully solved (SeDuMi-1.3)|
|           Bilinear SDP|        N/A|                 No suitable solver|
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Checking OPTI Toolbox Installation:
Checking Paths...                Ok
Checking    LP Solver Results... Ok
Checking  MILP Solver Results... Warning:  Non-integral bounds for integer variables rounded.
Ok
Checking    QP Solver Results... Ok
Checking  MIQP Solver Results... Ok
Checking   SDP Solver Results... Ok
Checking   NLS Solver Results... Ok
Checking   NLP Solver Results... Ok
Checking MINLP Solver Results... Ok

Toolbox Checked Out Ok! - Enjoy
```

Further tests of a DynareOBC install

- Navigate to the DynareOBC sub-directory:

  Tests\ComparisonOfPerfectForesightSolutionsForLinearModels

- Then run `RunAllAndCompare.`
- You should (eventually) see one or two graphs:
  - The first shows the difference between a DynareOBC simulation and an ExtendedPath one.
    - If you are using an unstable with a pre-27/05/2016 build date, there may be big gaps here due to a Dynare bug.
  - The second shows the difference between a DynareOBC simulation and an OccBin one.
    - This will only be present if you are using a fairly recent Dynare Unstable.
  - Note: without Dynare unstable there may be major differences between DynareOBC and the ExtendedPath, or it may fail to solve the ExtendedPath entirely, due to a Dynare bug.
- Do not conclude from this that DynareOBC can't do anything that EP and OccBin can't do!
  - This test uses a special set of options to produce a valid comparison.

Generating perfect foresight impulse responses with Dynare

- Open a MOD file for an NK model, e.g.:

  Examples\BraunKorberWaki2012\bkw2012.mod

- If your model does not already include a ZLB on interest rates, add it by modifying the interest rule to be of the form:

```
r = max( 0, re + phi_pi*(pi-pi_STEADY) + … );
```

- Make sure your `stoch_simul` command contains `periods=0` and `irf=40` (or some other positive integer).
- Then, from the folder containing the MOD file, run:

```
dynareOBC mod_file_name.mod nocubature
```

- E.g.:

```
dynareOBC bkw2012.mod nocubature
```

DynareOBC's behaviour (1/2)

- Watching the output, you will see three separate Dynare calls.

- You will also notice some printed diagnostics about existence and uniqueness.
  - These will be discussed fully tomorrow.

- You will then see some IRFs.
- In these IRFs:
  - The solid line is the impulse response imposing the constraint(s).
  - The dotted line is the path in the absence of constraints.
- IRFs are stored in `oo_.irfs`, as in standard Dynare.

DynareOBC's behaviour (2/2)

- If you were using the BKW model, you might find that the ZLB has not been hit in any of the IRFs.

- To fix this, we need to increase the scale of the impulse used to generate the IRFs. This may be done by running e.g.:

```
dynareOBC bkw2012.mod nocubature shockscale=3
```

- The number 3 after shockscale specifies that we want the IRF to magnitude 3 shocks.
  - Shockscale does not change the standard deviation of the model's shocks, it only scales the initial impulse.
  - This difference will be important when we look at rational expectations rather than perfect foresight solutions.

Useful references for other MOD files for experimentation

- DynareOBC contains a number of examples within its Examples sub-folder.
- As does Dynare, again within its examples sub-folder.
- Johannes Pfeifer maintains a large collection of MOD files for popular models here:

  https://github.com/JohannesPfeifer/DSGE_mod

- The Wieland et al. model database contains further MOD files for standard models (though they are all somewhat unwieldy), here:

  http://www.macromodelbase.com/download/

- You no doubt have many models of your own floating around your harddrive.

DynareOBC options

- "`nocubature`" and "`shockscale=3`" are some of DynareOBC's many options.
- All options are case insensitive, so you can as well enter "`NoCubature`".
- When an option takes a parameter (such as `shockscale`), there must not be spaces between the option name, the equals sign and the value.

- To see all of DynareOBC's options, just enter "`dynareOBC`" without arguments in MATLAB.
- Alternatively, view the DynareOBC home-page:

https://github.com/tholden/dynareOBC

- Or look at README.md or ReadMe.pdf in the DynareOBC root directory.

DynareOBC's algorithm

- The base of the algorithm is a very efficient perfect foresight solver for models with occasionally binding constraints.
  - Unique in being guaranteed to find a solution in finite time if one exists.
  - If no solution exists, will also report this in finite time too.
  - Also applies to high order pruned perturbation solutions to the model.

- DynareOBC restores consistency with rational expectations by integrating over future uncertainty.
  - Exploits the convenient properties of pruned perturbation solutions to very efficiently integrate over a large number of periods of future uncertainty.
  - This will be discussed later today.

The set-up without bounds (1/3)

- Suppose for $t \in \mathbb{N}^+$:

$$\left(\hat{A} + \hat{B} + \hat{C}\right)\hat{\mu} = \hat{A}\hat{x}_{t-1} + B\hat{x}_t + \hat{C}\mathbb{E}_t\hat{x}_{t+1} + \hat{D}\varepsilon_t,$$

  - where $\mathbb{E}_{t-1}\varepsilon_t = 0$ for all $t \in \mathbb{N}^+$,
  - $\varepsilon_t = 0$ for $t > 1$, (impulse response/perfect foresight simulation).

- $\hat{x}_0$ is given as an initial condition.
- Terminal condition: $\hat{x}_t \to \hat{\mu}$ as $t \to \infty$.

The set-up without bounds (2/3)

- For $t \in \mathbb{N}^+$, define:

$$x_t \coloneqq \begin{bmatrix} \hat{x}_t \\ \varepsilon_{t+1} \end{bmatrix}, \qquad \mu \coloneqq \begin{bmatrix} \hat{\mu} \\ 0 \end{bmatrix}, \qquad A \coloneqq \begin{bmatrix} \hat{A} & \widehat{D} \\ 0 & 0 \end{bmatrix}, \qquad B \coloneqq \begin{bmatrix} \hat{B} & 0 \\ 0 & I \end{bmatrix}, \qquad C \coloneqq \begin{bmatrix} \hat{C} & 0 \\ 0 & 0 \end{bmatrix}$$

- then, for $t \in \mathbb{N}^+$:

$$(A + B + C)\mu = Ax_{t-1} + Bx_t + Cx_{t+1},$$

- and $x_0 = \begin{bmatrix} \hat{x}_0 \\ \varepsilon_1 \end{bmatrix}$, $x_t \to \mu$ as $t \to \infty$.
- Take this as the form of our problem without bounds in the following.

The set-up without bounds (3/3)

- **Problem 1**
- Suppose that $x_0 \in \mathbb{R}^n$ is given. Find $x_t \in \mathbb{R}^n$ for $t \in \mathbb{N}^+$ such that $x_t \to \mu$ as $t \to \infty$, and such that for all $t \in \mathbb{N}^+$:

$$(A + B + C)\mu = Ax_{t-1} + Bx_t + Cx_{t+1}.$$

- **Assumption:** For any given $x_0 \in \mathbb{R}^n$, Problem 1 has a unique solution, of the form $x_t = (I - F)\mu + Fx_{t-1}$, for $t \in \mathbb{N}^+$, where $F = -(B + CF)^{-1}A$, and all of the eigenvalues of F are weakly inside the unit circle.
- **Assumption:** $\det(A + B + C) \neq 0$.

The set-up with bounds

- **Problem 2**
- Suppose that $x_0 \in \mathbb{R}^n$ is given. Find $T \in \mathbb{N}$ and $x_t \in \mathbb{R}^n$ for $t \in \mathbb{N}^+$ such that $x_t \to \mu$ as $t \to \infty$, and such that for all $t \in \mathbb{N}^+$:

$$x_{1,t} = \max\{0, I_{1,\cdot}\mu + A_{1,\cdot}(x_{t-1} - \mu) + (B_{1,\cdot} + I_{1,\cdot})(x_t - \mu) + C_{1,\cdot}(x_{t+1} - \mu)\},$$

$$(A_{-1,\cdot} + B_{-1,\cdot} + C_{-1,\cdot})\mu = A_{-1,\cdot}x_{t-1} + B_{-1,\cdot}x_t + C_{-1,\cdot}x_{t+1},$$

- and such that $x_{1,t} > 0$ for $t > T$.
- Ruling out solutions that get stuck at another steady-state by assumption.

A note on the terminal condition used by DynareOBC

- Our terminal condition implies that agents believe that if no shocks arrived in the future, the economy would eventually return to the original steady-state.

- Accords with a belief in the credibility of the long-run inflation target.
  - Christiano and Eichenbaum (2012) argue a central bank may rule out the deflationary equilibria in practice by switching to a money growth rule following severe deflation, along the lines of Christiano and Rostagno (2001).
  - Belief in credibility of the long-run target is in line with the evidence of Gürkaynak, Levin, and Swanson (2010).

- In line with the approach of Brendon, Paustian, and Yates (2015).
- Contrary to the approach of, e.g. Benhabib, Schmitt-Grohe, and Uribe (2001a,b), Schmitt-Grohe and Uribe (2012), Mertens and Ravn (2014), Aruoba, Cuba-Borda and Schorfheide (2013).

The news shock set-up

- **Problem 3**
- Suppose that $T \in \mathbb{N}$, $x_0 \in \mathbb{R}^n$ and $y_0 \in \mathbb{R}^T$ is given. Find $x_t \in \mathbb{R}^n, y_t \in \mathbb{R}^T$ for $t \in \mathbb{N}^+$ such that $x_t \to \mu$, $y_t \to 0$, as $t \to \infty$, and such that for all $t \in \mathbb{N}^+$:

$$(A + B + C)\mu = Ax_{t-1} + Bx_t + Cx_{t+1} + I_{\cdot,1}y_{1,t-1},$$
$$\forall i \in \{1, \dots, T-1\}, \qquad y_{i,t} = y_{i+1,t-1},$$
$$y_{T,t} = 0.$$

- A version of Problem 1 with news shocks up to horizon T added to the first equation.
  - The value of $y_{t,0}$ gives the news shock that hits in period t.
  - I.e. $y_{1,t-1} = y_{t,0}$ for $t \leq T$, and $y_{1,t-1} = 0$ for $t > T$.

A representation of solutions to Problem 3

- **Lemma**: There is a unique solution to Problem 3 that is linear in x_0 and y_0.

- Let $x_t^{(3,k)}$ be the solution to Problem 3 when $x_0 = \mu$, $y_0 = I_{\cdot,k}$.
- Let $M \in \mathbb{R}^{T \times T}$ satisfy:

$$M_{t,k} = x_{1,t}^{(3,k)} - \mu_1, \qquad \forall t, k \in \{1, .., T\},$$

  - i.e. M horizontally stacks the (column-vector) relative impulse responses to the news shocks.
  - In DynareOBC, M is stored in `dynareOBC_.MMatrix`.
- Let $x_t^{(1)}$ be the solution to Problem 1 for some given x_0.
- Then the solution to Problem 3 for given x_0, y_0 satisfies:

$$\left(x_{1,1\ldots T}\right)' = q + My_0,$$

  - where $q \coloneqq \left(x_{1,1\ldots T}^{(1)}\right)'$.

The links between the solutions to Problem 2 and the solution to Problem 3 (1/2)

- Let $x_t^{(2)}$ be a solution to Problem 2 given an arbitrary x_0.
- Define:

$$e_t := \begin{cases} -\left[I_{1,\cdot}\mu + A_{1,\cdot}\left(x_{t-1}^{(2)} - \mu\right) + \left(B_{1,\cdot} + I_{1,\cdot}\right)\left(x_t^{(2)} - \mu\right) + C_{1,\cdot}\left(x_{t+1}^{(2)} - \mu\right)\right] & \text{if } x_{1,t}^{(2)} = 0 \\ 0 & \text{if } x_{1,t}^{(2)} > 0 \end{cases}$$

- **Lemma:** The following statements hold:
  - $e_{1\ldots T} \geq 0$, $x_{1,1\ldots T}^{(2)} \geq 0$ and $x_{1,1\ldots T}^{(2)} \circ e_{1\ldots T} = 0$,
  - $x_t^{(2)}$ is the unique solution to Problem 3 when started with $x_0 = x_0^{(2)}$ and with $y_0 = e'_{1\ldots T}$.
  - If $x_t^{(2)}$ solves Problem 3 when started with $x_0 = x_0^{(2)}$ and with some y_0, then $y_0 = e'_{1\ldots T}$.

The links between the solutions to Problem 2 and the solution to Problem 3 (2/2)

- **Proposition**: The following statements hold:

  - Let $x_t^{(3)}$ be the unique solution to Problem 3 when initialized with some x_0, y_0. Then $x_t^{(3)}$ is a solution to Problem 2 when initialized with x_0 if and only if $y_0 \geq 0$, $y_0 \circ (q + My_0) = 0$, $q + My_0 \geq 0$ and $x_{1,t}^{(3)} \geq 0$ for all $t \in \mathbb{N}$ with $t > T$.

  - Let $x_t^{(2)}$ be any solution to Problem 2 when initialized with x_0. Then there exists a $y_0 \in \mathbb{R}^T$ such that $y_0 \geq 0$, $y_0 \circ (q + My_0) = 0$, $q + My_0 \geq 0$, such that $x_t^{(2)}$ is the unique solution to Problem 3 when initialized with x_0, y_0.

Linear complementarity problems (LCPs)

- The previous proposition establishes that solving the model with occasionally binding constraints is equivalent to solving the following "linear complementarity problem".

- **Problem 4**
- Suppose $q \in \mathbb{R}^T$ and $M \in \mathbb{R}^{T \times T}$ are given. Find $y \in \mathbb{R}^T$ such that: $y \geq 0$, $y \circ (q + My) = 0$ and $q + My \geq 0$.
- We call this the linear complementarity problem (LCP) (q, M).

- DynareOBC solves these LCPs using a mixed integer linear programming representation which will be discussed tomorrow.
- By default DynareOBC first looks for a solution with $T = 0$, then a solution with $T = 1$, then one with $T = 2$, and so on.

Generalisations

- For multiple bounds:
  - DynareOBC stacks the impulse responses of the bounded variables ignoring bounds into q.
  - DynareOBC stacks the vectors of news shocks to each variable into y.
  - M is a block matrix of each bounded variable's responses to each bounded variable's news shocks.
  - Then the stacked solution for the paths of the bounded variables is $q + My$, and we again have an LCP, so results go through as before.
- For bounds not at zero:
  - If $z_{1,t} = \max\{z_{2,t}, z_{3,t}\}$, then $z_{1,t} - z_{2,t} = \max\{0, z_{3,t} - z_{2,t}\}$.
- For minimums:
  - If $z_{1,t} = \min\{z_{2,t}, z_{3,t}\}$, then $-z_{1,t} = \max\{-z_{2,t}, -z_{3,t}\}$.

Application to models with uncertainty

- To convert the perfect foresight solver into a solver for stochastic models, DynareOBC uses a variant of the extended path algorithm of Fair and Taylor (1983).
  - Each period DynareOBC draws a shock, and then solves for the expected future path of the model, ignoring the impact of the OBC on expectations (for now).
  - From this expected path, DynareOBC solves for the news shocks necessary to impose the bound.
  - DynareOBC then adds those news shocks to today's variables, and steps the model forward using the model's transition matrix.

Simulation of models with occasionally binding constraints with DynareOBC

1. Take a MOD file containing a max, min or abs function.
2. Ensure that the `stoch_simul` command contains `periods=1100` (or some other positive integer).
3. Run, e.g.:

```
dynareOBC bkw2012.mod nocubature
```

   - The nocubature option ensures that DynareOBC treats the bound in a perfect foresight manner.

- You may notice that DynareOBC now spends some time obtaining parameteric solutions.
  - These speed up solving bound problems in which the bound only binds for a few periods.

DynareOBC simulation problems

- If you attempt to simulate the BKW model, you will receive the message: “Impossible problem encountered. Try increasing TimeToEscapeBounds, or reducing the magnitude of shocks.”

- This means that there is definitely no solution to the particular LCP that the algorithm was attempting to solve.
  - This contrasts with policy function iteration, and the Extended Path and OccBin algorithms, in which a failure to converge to a solution may just reflect a failure of the algorithm.

- However, in this case the problem is fixable by increasing `TimeToEscapeBounds`, which is the maximum T the algorithm considers. Its default value is 32, so we might try running, e.g.:

```
dynareOBC bkw2012.mod nocubature timetoescapebounds=64
```

DynareOBC simulation output

- Like standard Dynare, DynareOBC will print assorted moments of the simulated variables.
  - Note that the presence of occasionally binding constraints often leads to substantial skewness.
  - As in standard Dynare, DynareOBC discards the initial simulation periods. The precise number of periods discarded is controlled by the `drop` option to `stoch_simul`.
  - Note that DynareOBC starts simulations from a draw from the stationary distribution in the absence of OBCs, so large values for `drop` should not be needed.

- As in standard Dynare, the simulated variables (with the bounds imposed) are stored in `oo_.endo_variables`, which is ordered in declaration order.
- `dynareOBC_.SimulationsWithoutBounds` contains the simulation without imposing the bound, for comparison.

The non-linear problem solved by DynareOBC (1/2)

- **Problem 6**
- Suppose that $x_0 \in \mathbb{R}^n$ is given and that $f: \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^c \times \mathbb{R}^m \to \mathbb{R}^n$, $g, h: \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^c \times \mathbb{R}^m \to \mathbb{R}^c$ are given continuously $d \in \mathbb{N}^+$ times differentiable functions.

- Find $x_t \in \mathbb{R}^n$ and $v_t \in \mathbb{R}^c$ for $t \in \mathbb{N}^+$ such that for all $t \in \mathbb{N}^+$:

$$0 = \mathbb{E}_t f(x_{t-1}, x_t, x_{t+1}, v_t, \varepsilon_t),$$
$$v_t = \mathbb{E}_t \max\{h(x_{t-1}, x_t, x_{t+1}, v_t, \varepsilon_t), g(x_{t-1}, x_t, x_{t+1}, v_t, \varepsilon_t)\}$$

- where $\varepsilon_t \sim \text{NIID}(0, \Sigma)$, where the max operator acts elementwise on vectors, and where the information set is such that for all $t \in \mathbb{N}^+$, $\mathbb{E}_{t-1}\varepsilon_t = 0$ and $\mathbb{E}_t\varepsilon_t = \varepsilon_t$.

The non-linear problem solved by DynareOBC (2/2)

- **Assumption:** There exists $\mu_x \in \mathbb{R}^n$ and $\mu_v \in \mathbb{R}^c$ such that:

$$0 = f(\mu_x, \mu_x, \mu_x, \mu_v, 0),$$

$$\mu_v = \max\{h(\mu_x, \mu_x, \mu_x, \mu_v, 0), g(\mu_x, \mu_x, \mu_x, \mu_v, 0)\},$$

- and such that for all $a \in \{1, \ldots, c\}$:

$$\big(h(\mu_x, \mu_x, \mu_x, \mu_v, 0)\big)_a \neq \big(g(\mu_x, \mu_x, \mu_x, \mu_v, 0)\big)_a.$$

DynareOBC at order=1 (1/2)

- Without loss of generality, suppose our model is:

$$0 = \mathbb{E}_t f(x_{t-1}, x_t, x_{t+1}, v_t, \varepsilon_t),$$
$$v_t = \mathbb{E}_t \max\{0, g(x_{t-1}, x_t, x_{t+1}, v_t, \varepsilon_t)\},$$

  - where $g(\mu_x, \mu_x, \mu_x, \mu_v, 0) \gg 0$.

- Linearizing around the steady-state gives:

$$v_t = \mu_v + g_1(x_{t-1} - \mu_x) + g_2(x_t - \mu_x) + g_3\mathbb{E}_t(x_{t+1} - \mu_x) + g_4(v_t - \mu_v) + g_5\varepsilon_t.$$

- DynareOBC replaces this with the more accurate:

$$v_t = \max\{0, \mu_v + g_1(x_{t-1} - \mu_x) + g_2(x_t - \mu_x) + g_3\mathbb{E}_t(x_{t+1} - \mu_x) + g_4(v_t - \mu_v) + g_5\varepsilon_t\}.$$

DynareOBC at order=1 (2/2)

- DynareOBC replaces this in turn with:

$$v_{a,t} = \mathbb{E}_t\left(g(x_{t-1}, x_t, x_{t+1}, v_t, \varepsilon_t)\right)_a + I_{1,\cdot} y_t^{(a)},$$

- for all $a \in \{1, \ldots, c\}$, where, for all $a \in \{1, \ldots, c\}$:

$$\forall i \in \{1, \ldots, T-1\}, \qquad y_{i,t}^{(a)} = y_{i+1,t-1}^{(a)} + \eta_{i,t}^{(a)}$$
$$y_{T,t}^{(a)} = \eta_{T,t}^{(a)}.$$

DynareOBC at higher orders (1/2)

- DynareOBC first takes a pruned perturbation approximation to the source non-linear model, using the Lan and Meyer-Gohde (2013) algorithm.

- A convenient property of pruned perturbation solutions of order d is that they are linear in additive shocks of the form η_t^d.
  - So using shocks of this form preserves the tractable linearity.
  - In fact the M matrix we get at second or higher order is equal to the M matrix at first order, (at least in the limit as the variance of the news shocks goes to zero).

- While we still treat the bound in a perfect-foresight manner (for now), by taking a higher order approximation we at least capture other risk channels.
  - More precisely, with `nocubature`, DynareOBC models agents as approximating expectations of the bounded variable by forming expectations in the model without bounds, and then adding on the "news" needed to impose the bound on these expectations.

DynareOBC at higher orders (2/2)

- To simulate at higher order, just set `order=2` or `order=3` in the call to `stoch_simul`.
- You might find that at order 3 simulation is a little slow, to speed it up, try the "`CompileSimulationCode`" and/or the "`Sparse`" option by running e.g.:

```
dynareOBC bkw2012.mod nocubature compilesimulationcode sparse
```

  - `CompileSimulationCode` compiles the code used for simulating without the OBCs. It requires a supported compiler. For details of how to get a free one, see the DynareOBC ReadMe.
  - `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.

Session outline: Day 1, Session 2 (1.5 hours)

- The stochastic extended path algorithm in the context of DynareOBC.
- Cubature rules.
- Using DynareOBC for approximate rational expectations solutions.
- Some accuracy results.
- Other DynareOBC options.

Integrating over future uncertainty (1/4)

- Adjemian and Juillard (2013) showed how a perfect foresight solver could account for future uncertainty by integrating over future shocks.
  - I.e. draw shocks for a certain number of future periods, $t+1, \ldots, t+S$.
  - Solve for the perfect foresight path assuming they were known at t.
  - Repeat many times to get expectations.

- In their very general non-linear set-up, doing this integration requires p^{mS} solutions of the perfect foresight problem,
  - for some $p > 1$, m is the number of shocks, S is the integration horizon.

- Solving their general perfect foresight problem is also orders of magnitude slower than solving our LCP.

Integrating over future uncertainty (2/4)

- Due to a somewhat subtle Jensen's inequality result, the Adjemian and Juillard (2013) algorithm is not even "consistent" for general non-linear models.
  - I.e. its approximation error does not go to zero as the integration horizon goes to infinity, even when exact integration is used.

- Adjemian and Juillard (2016) proposes an alternative algorithm which is consistent in the general non-linear case, and which is both considerably slower, and less amenable to parallelisation.

- In our special context, due to linearity in the news, the original Adjemian and Juillard (2013) algorithm is consistent, so we do not need to use the slower (2016) version.

Integrating over future uncertainty (3/4)

- Let $w_{t,s}$ be the value the bounded variables would take at s if the constraints did not apply from period t onwards.

- By the properties of pruned perturbation solutions, we can evaluate $\mathrm{cov}_t\left(w_{t,t+i}, w_{t,t+j}\right)$, for $t, i, j \in \mathbb{N}$ in closed form.
  - So we can take a Gaussian approximation to the joint distribution of $w_{t,t}, w_{t,t+1}$, …, and efficiently integrate over these variables via Gaussian cubature techniques.
  - Rather than exponential in both m and S evaluations, we just need polynomial in S evaluations.

- For each draw of $w_{t,t}, w_{t,t+1}$, …, we solve the bounds problem to get the cumulated news shocks (i.e. y).

Integrating over future uncertainty (4/4)

- Unlike Adjemian and Juillard (2013) we do not just consider full variance shocks up to some horizon, and then nothing beyond.

- Instead, we apply a windowing function to the shock variances, to ensure that the covariance is a smooth function of time.
  - This reduces artefacts caused by the sudden change at horizon S.

- In particular, we scale the shock variance at horizon k by:

$$\frac{1}{2}\left(1+\cos\left(\pi\frac{k-1}{S}\right)\right).$$

- The cosine form has some desirable frequency domain properties.

Three alternative Gaussian cubature methods

- With $\hat{S} \leq S$ the integration dimension, these are:
  - A degree 3 monomial rule with $2\hat{S}+1$ nodes and positive weights.
    - Positive weights give robustness. Evaluates far from steady-state though.
  - The Genz and Keister (1996) Gaussian cubature rules with $\mathrm{O}\left(\hat{S}^K\right)$ nodes.
    - $2K+1$ is the degree of monomial integrated exactly.
    - Since the rules are nested, adaptive degree is possible.
    - In DynareOBC, this is implemented in `fwtpts.m`.
  - Quasi-Monte Carlo.
    - Much less efficient than the others on well behaved functions, but is much better behaved on non-differentiable ones.
    - In DynareOBC, this is implemented in `SobolSequence.m`.

Using cubature in DynareOBC (1/3)

- To turn on cubature, you just need to invoke DynareOBC ***without*** the `nocubature` option.

- The integration horizon "S" is determined by setting `PeriodsOfUncertainty=16`, or some other value.
  - 16 is actually the default.
  - Due to the cosine window, the effective number of periods of uncertainty is roughly half this number.

- By default, DynareOBC uses the Genz and Keister (1996) rules.

Using cubature in DynareOBC (2/3)

- To specify the polynomial degree that should be integrated exactly, invoke DynareOBC with `MaxCubatureDegree=7`, or some other value.
  - 7 is actually the default.

- Since the Genz and Keister (1996) rules sometimes oscillate from too high to too low as the degree increases, DynareOBC also gives the option to smooth the results of adjacent degrees by setting `CubatureSmoothing=0.1` or to some other float in the unit interval.
  - This reduces the effective degree of the integration rule by two.

- To speed up integration, DynareOBC exploits the fact that the Genz and Keister (1996) rules are nested, and does not integrate to the full degree if rules of adjacent lower degrees give sufficiently close answers.
  - Setting e.g. `CubatureTolerance=1e-6` determines the meaning of “sufficiently close”.

Using cubature in DynareOBC (3/3)

- Invoking DynareOBC with the FastCubature option replaces the default cubature method with the degree three monomial rule.
  - This is much faster, so is advisable in larger models.

- Invoking DynareOBC with `QuasiMonteCarloLevel=7`, or some other value, replaces the default integration method with Q.M.C. using $2^{1+7} - 1$ points (in this case).
  - This is better grounded in theory than the other methods, since it handles kinks, but is unlikely to dominate them without very large run times.

Options to further speed up cubature in DynareOBC

- `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.

Accuracy results

- To help understand the merits of the various rules, we give some accuracy results.

- We examine three simple models, to ensure accuracy tests are reliable.

- These are:
  - A very simple model with an analytic solution.
  - A model for which log-linearization gives the exact answer in the absence of bounds.
  - An otherwise linear open-economy model.

A simple model

- Closed economy, no capital, inelastic unit labour supply.
- Households maximise:

$$\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \frac{C_t^{1-\gamma} - 1}{1-\gamma}$$

- Subject to the budget constraint:

$$A_t + R_{t-1}B_{t-1} = C_t + B_t$$

  - A_t is productivity. R_t is the real interest rate.
  - B_t is the household's holdings of zero net supply bonds.

- Define $g_t := \log A_t - \log A_{t-1}$. Evolves according to:

$$g_t = \max\{0, (1-\rho)\bar{g} + \rho g_{t-1} + \sigma\varepsilon_t\},$$

- $\varepsilon_t \sim \text{NIID}(0,1)$, $\beta := 0.99$, $\gamma := 5$, $\bar{g} := 0.05$, $\rho := 0.95$, $\sigma := 0.07$.

Accuracy in the simple model, along simulated paths of length 1000 (after 100 periods dropped)

Bound in Model	Order	Cubature	Seconds	Mean Abs Error	Root Mean Squared Error	Max Abs Error	Mean Abs Error at Bound
No	1	N/A	66	6.13E-04	6.13E-04	6.13E-04	
No	2	N/A	62	1.52E-17	2.36E-17	1.67E-16	
No	3	N/A	53	1.99E-17	2.72E-17	1.11E-16	
Yes	1	No	141	3.67E-03	6.05E-03	1.31E-02	1.31E-02
Yes	2	No	139	3.76E-03	6.39E-03	1.37E-02	1.37E-02
Yes	3	No	140	3.76E-03	6.39E-03	1.37E-02	1.37E-02
Yes	1	Monomial, Degree 3	274	7.32E-04	8.45E-04	1.88E-03	7.40E-04
Yes	2	Monomial, Degree 3	1537	4.18E-04	6.73E-04	1.97E-03	1.28E-04
Yes	3	Monomial, Degree 3	1397	4.18E-04	6.73E-04	1.97E-03	1.28E-04
Yes	2	Sparse, Degree 3	1794	9.65E-04	1.67E-03	3.85E-03	3.85E-03
Yes	2	Sparse, Degree 5	1840	9.65E-04	1.67E-03	3.85E-03	3.85E-03
Yes	2	Sparse, Degree 7	2009	5.25E-04	9.30E-04	2.17E-03	2.17E-03
Yes	2	QMC, 15 Points	1965	9.12E-04	1.27E-03	2.17E-03	2.17E-03
Yes	2	QMC, 31 Points	2214	5.98E-04	8.17E-04	1.39E-03	1.39E-03
Yes	2	QMC, 63 Points	3184	4.04E-04	5.49E-04	9.55E-04	9.55E-04
Yes	2	QMC, 1023 Points	5197	1.57E-04	2.30E-04	4.45E-04	4.45E-04

[1] All timings are "wall" time, and include time spent starting the parallel pool, time spent compiling code (although written in MATLAB, DynareOBC generates and compiles C code for key routines), and time spent calculating accuracy. Code was run on one of the following (very similar) twenty core machines: 2x E5-2670 v2 2.5GHz, 64GB RAM; 2x E5-2660 v3 2.6GHz, 128GB RAM. Use of machines with network attached storage means that there may be some additional variance in these timings.

[2] Errors conditional on the bounded variable being less than *0.0001*. The numbers for this column would be identical had we used root mean squared errors or maximum absolute errors, conditional on being at the bound.

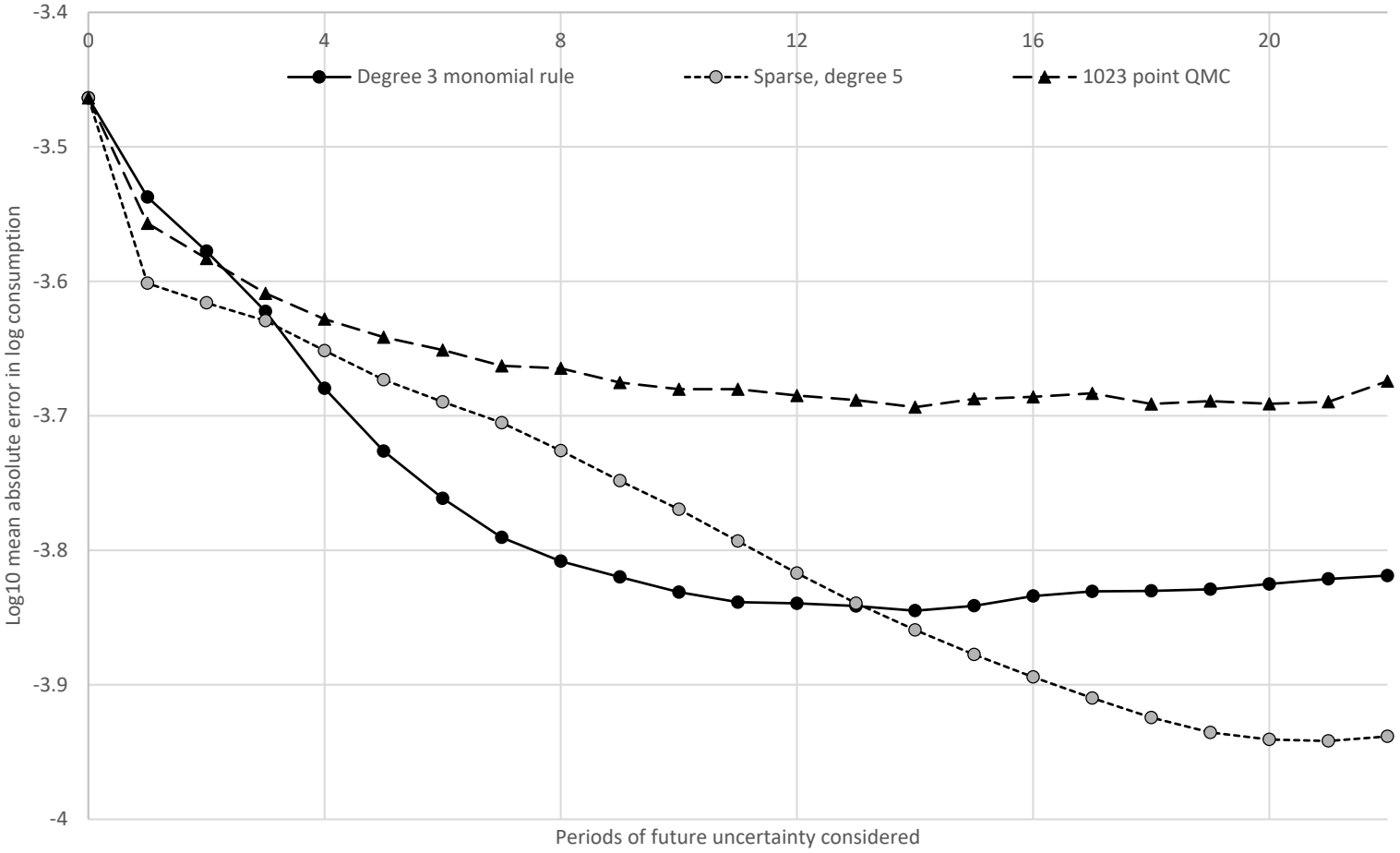
A model for which log-linearization is exact without bounds

- The social planner chooses consumption, C_t, L_t, and K_t, to maximise:

$$\mathbb{E}_t \sum_{k=0}^{\infty} \beta^k \left[\log C_{t+k} - \frac{L_{t+k}^{1+\nu}}{1+\nu} \right],$$

- subject to the capital constraint: $K_t \geq \theta K_{t-1}$,
- and the budget constraint $C_t + K_t = Y_t = A_t K_{t-1}^{\alpha} L_t^{1-\alpha}$.
- Productivity, A_t, evolves according to $A_t = A_{t-1}^{\rho} \exp \varepsilon_t$, where $\varepsilon_t \sim \mathrm{N}(0, \sigma^2)$.
- In the following, we set $\alpha = 0.3$, $\beta = 0.99$, $\nu = 2$, $\theta = 0.99$, $\rho = 0.95$ and $\sigma = 0.01$.
- Compare to a full global solution.

Effect of increasing periods of uncertainty on accuracy, along simulated paths (as before)



An otherwise linear open-economy model

- The social planner chooses C_t, D_t and B_t to maximise:

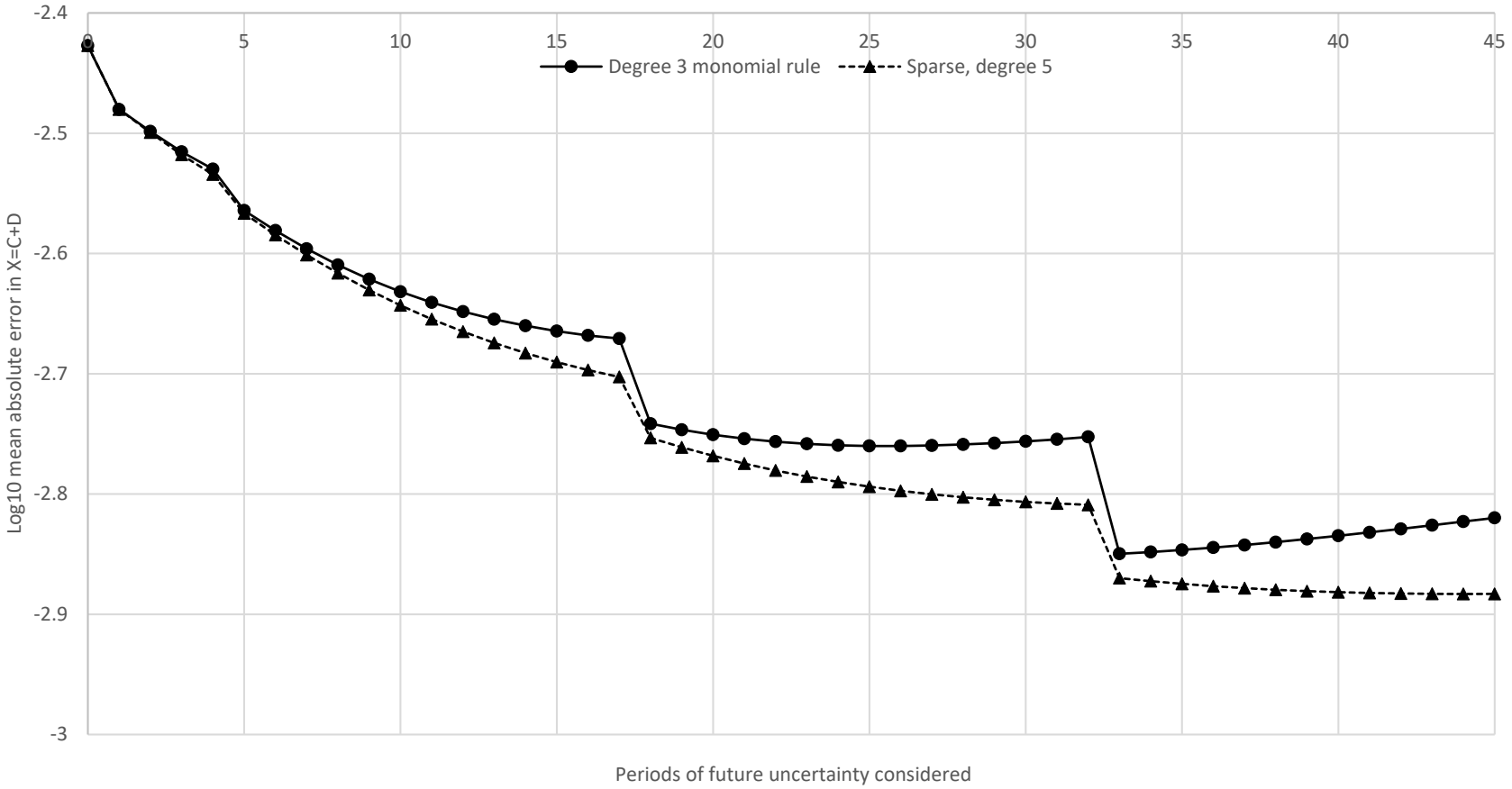
$$\mathbb{E}_t \sum_{k=0}^{\infty} \beta^k \left[-\frac{1}{2}(1 - C_t)^2 - \frac{\phi}{2} B_t^2 \right],$$

- subject to the budget constraint: $C_t + D_t + B_t - RB_{t-1} = Y_t = \max\{\underline{Y}, A_t\}$,
- the positivity constraints: $0 \leq C_t,\ 0 \leq D_t$,
- and the certain repayment of interest constraint:

$$\forall k \in \mathbb{N}^+, \qquad \Pr_t\big((R-1)B_t \leq Y_{t+k}\big) = 1.$$

- Productivity evolves according to $A_t = (1-\rho)\mu + \rho A_{t-1} + \sigma\varepsilon_t$, where $\varepsilon_t \sim \text{NIID}(0,1)$.
- We set $\beta = 0.99$, $\mu = 0.5$, $\rho = 0.95$, $\sigma = 0.05$, $\underline{Y} = 0.25$, $R = \beta^{-1}$ and $\phi = R - 1$.

Effect of increasing periods of uncertainty on accuracy, along simulated paths (as before)



Other key DynareOBC options: 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.
- `TimeToReturnToSteadyState=INTEGER` (default: `64`)
  - The number of periods in which to verify that the constraints are not being violated.
- `FullHorizon`
  - By default, DynareOBC finds a solution for 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`).
- `IgnoreBoundFailures`
  - Makes DynareOBC atttempt to continue even after it has failed to solve the bounds problem due to e.g. infeasability. This will severely compromise accuracy.

Other key DynareOBC options: For controlling accuracy (and speed)

- `FirstOrderAroundRSS`
  - Takes a linear approximation around the risky steady state of the non-linear model, using the Meyer-Gohde (2014) approach. 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 , using the Meyer-Gohde (2014) approach. If specifying this option, you should set `order=2` or `order=3` in your mod file.

- `FirstOrderConditionalCovariance`
  - When `order` is greater than one (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.

Other key DynareOBC options: For controlling IRFs

- `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.

Session outline: Day 1, Session 3 (1 hour)

- Average IRFs.
- The Dynare pre-processor.
- Use of MLVs.
- Use of the Dynare Transformation Engine.
- Simulation of MLVs.

Interpreting `nocubature` IRFs

- With `order=1`, the interpretation of a `nocubature` IRF is clear—it is just a perfect foresight impulse response.

- At higher orders, in a model without OBCs, DynareOBCs impulse responses give the expected response to the shock, without conditioning on the current state or the future shocks.
  - These are the same as the average IRFs produced by Dynare.
  - The difference is that whereas Dynare's IRFs are produced by Monte Carlo, these are calculated without repeated simulation, thanks to the convenient properties of pruned perturbation solutions.

- In the presence of OBCs, with `nocubature`, DynareOBC calculates the base IRF path as it would if there were no OBC, and then imposes the OBC upon that.
  - This is also how agent expectations are approximated with `nocubature`.
  - This is a useful indicative approximation, but it is not appropriate for "production" use.

Interpreting IRFs with cubature

- Even with cubature, IRFs are still a fudge for models with OBCs.
- With cubature, DynareOBC integrates out the effects of future shocks on whether the model is at the bound in future, but it does not integrate out the effect of current uncertainty on the bound.
- Thus, while the base path takes into account current uncertainty, current uncertainty is ignored when calculating the expected news shocks.
- Again this is a useful indicative approximation, but it is not appropriate for production use.

Average IRFs

- An improvement would be to calculate the average response in a world in which agents act as they do under the respective simulation algorithm.
  - This will be an improvement even if the `nocubature` option is specified, since in that case, while agent behaviour is still not fully rational, at least this way our IRFs are the true average IRF given the behavioural approximation.
  - If accurate cubature is used for calculating agent expectations, then these average IRFs will approach the average IRFs of the "true" model.

- DynareOBC calculates such IRFs by repeated simulation when the `slowirfs` option is provided, with e.g.:

```
dynareOBC bkw2012.mod nocubature timetoescapebounds=64 slowirfs
```

  - For final "production" IRFs, I always recommend the `slowirfs` option is chosen, ideally combined with a reasonable cubature method.

- As in base Dynare, the number of Monte Carlo repetitions is controlled by the `replic` option of `stoch_simul`.
  - Due to the high degree of non-linearity in models with OBCs, a high value of `replic` may be required.

Increasing simulation accuracy in any model (with or without OBCs) (1/4)

- In non-linear models, perturbation/DynareOBC simulation accuracy is crucially determined by the transformation applied to the source variables (e.g. log-linearization versus linearization).
  - Judd (2002), “Perturbation Methods with Nonlinear Changes of Variables” is an early reference for this.

- While perturbation is a local method, and so its local properties are unaffected by changes of variables, its global properties are affected.

Increasing simulation accuracy in any model (with or without OBCs) (2/4)

- For a trivial example, consider the "model":

$$Z_t = \exp \varepsilon_t \,,$$

  - where $\varepsilon_t \sim \mathrm{NIID}(0,1)$.
- Then:
  - A first order approximation in levels gives $Z_t \approx 1 + \varepsilon_t$, which has zero skewness and which violates $Z_t \geq 0$.
  - A second order approximation in levels gives $Z_t \approx 1 + \varepsilon_t + \frac{1}{2}\varepsilon_t^2$, which violates the monotonicity of Z_t in ε_t (for $\varepsilon_t \leq -1$).
  - A first order approximation in logs gives $z_t = \varepsilon_t$, where $z_t = \log Z_t$.

- For a more practical example, simulate an RBC model in levels versus in logs, and ramp up the shock variance.
  - E.g. compare "example1.mod" and "example2.mod" from the Dynare examples directory.
  - With sufficiently large variance, in levels the mean of the capital stock will be negative!

Increasing simulation accuracy in any model (with or without OBCs) (3/4)

- One useful guide is that the model's variables and equations should be transformed so that the model is as near as possible to being linear.
  - Be careful to respect Jensen's inequality though.
  - $\frac{1}{C_t} = \beta R_t \mathbb{E}_t \frac{1}{C_{t+1}}$ does not imply $C_t = \frac{1}{\beta R_t} \mathbb{E}_t C_{t+1}$.

- Linearity in expectations operators is particularly helpful (and this is particularly true when using DynareOBC).
  - I.e. replace equations of the form $f(x_t) = \mathbb{E}_t y_{t+1}$ with $z_t = \mathbb{E}_t y_{t+1}$ where $z_t = f(x_t)$.

- For DynareOBC, it is also desirable to have linearity in $\max$, $\min$ and abs operators.
  - So equations of the form $f(x_t) = \max\{0, y_t\}$ should be replaced with $z_t = \max\{0, y_t\}$, where $z_t = f(x_t)$.

Increasing simulation accuracy in any model (with or without OBCs) (4/4)

- One way to achieve "near" linearity is to transform all variables such that in the absence of OBCs, the transformed variables can take values on the whole real line.
  - Thus we should always work with e.g. logs of capital, output, etc.

- It may also help to transform all equations without expectations so that both sides can take values on the whole real line.
  - I.e. enter $\exp z_t = \exp \varepsilon_t$ as $z_t = \varepsilon_t$.

- A final suggestion for improving accuracy is to work with as few endogenous variables as possible. (This also improves simulation/estimation speed.)
  - If $Z_t = f(x_t, x_{t-1}, \varepsilon_t)$, then Z_t can be substituted out of the model, and simulations of Z_t can be recovered from simulations of x_t and ε_t.
  - Do not approximate variables that do not need to be approximated!
  - We will see that model local variables and DynareOBC can help with this.
  - All endogenous variables in a MOD file should enter with a +1 or a -1 somewhere.

Introduction to the Dynare Transformation Engine

- The Dynare Transformation Engine is a set of files which are designed to speed up the creation of MOD files involving transformed variables.
  - It also results in much cleaner, easier to read, MOD files that ones in which the transformation is applied manually.

- It is available from:

  https://github.com/tholden/DynareTransformationEngine

  - Just click the “clone or download” button, and then either “Open in Desktop” if you have GitHub Desktop installed, or “Download ZIP” otherwise.

- Internally, it makes heavy use of Dynare’s pre-processor, and model local variables, so we will begin with a quick review of these Dynare features.

The Dynare pre-processor language: Basics

- Before the .mod file is passed to the main Dynare processing engine, it first goes through a pre-processor.
- This parses the original .mod file and generates a new .mod file from it.

- Commands may be passed to the pre-processor using lines beginning with the symbols "`@#`".
  - `@#define VariableName = VariableValue` sets the value of a pre-processor variable.
    - This variable is only known to the pre-processor, not to Dynare or MATLAB.
  - `@#for Index in Array … @#endfor` starts a pre-processor for loop. The pre-processor will iterate over the elements of `Array`.
  - `@#if … @#else … @#endif` gives a pre-processor conditional.

- To see what the pre-processor is doing, we can ask it to save the intermediate .mod file using "`dynare ModFileName.mod savemacro nolinemacro`".
  - These options also work in DynareOBC.

The Dynare pre-processor language: Variables

- The Dynare pre-processor supports integer variables, strings and arrays.
  - It does not support floating point numbers.

- The usual mathematical operations may be applied to integers.
- The Dynare pre-processor uses C-notation for comparisons and logical operators.
  - So `==` tests comparison, `!=` tests difference and `!` is the "not" operator.

- Pre-processor strings are defined by text in double quotes, e.g.: `"a string"`.
- Strings may be concatentated with `+`.
- Substrings may be extracted with e.g. `MyString[3:5]`.

- The text representation of a pre-processor variable may be inserted in to the generated .mod file using `@{VariableName}`.

The Dynare pre-processor language: Arrays

- Array variables may be defined with either MATLAB ":" syntax (e.g. `1:10`), or by specifying elements (e.g. `["home", "foreign"]`).
- `+` and – on arrays performs concatenation and set-difference, respectively.
- Array membership may be tested with the `in` operator.
- Pre-processor arrays may be dereferenced with e.g. `MyArray[3]` or `MyArray[4:8]`.
- The `length` operator returns the length of an array with e.g. `length(MyArray)`.

The pre-processor language: Other commands

- The pre-processor also supports the following commands:
  - `@#include "ModFileName.mod"` which inserts the contents of the .mod file ModFileName.mod.
  - `@#echo` which displays the following text to the user of the .mod file.
  - `@#error` which stops further pre-processing, and quits Dynare.

The difference between pre-processor variables and model local variables

- Model local variables (MLVs) are defined by lines starting with a "#" and ending with a semi-colon.
  - E.g. `#PI = exp(pi);`.
- Model local variables are interpreted by the main Dynare processor, and they can only occur within the model block.
- Dynare effectively copies and pastes the contents of the model local variable into all of the equations in which it is used, surrounding the entire variable with brackets.

- Similar results may be accomplished with both MLVs and pre-processor variables.
- The chief difference between the two is that while the pre-processor variable is effectively copied and pasted in as is, the MLV is effectively copied and pasted in surrounded by brackets.

More on model local variables

- A further difference between MLVs and pre-processor variables is that whereas the pre-processor cannot write another definition for a pre-processor variable, the pre-processor can be used to create definitions of new MLVs.
  - This feature is used heavily by the Dynare Transformation Engine.

- One limitation of MLVs is that some of Dynare's solution algorithms do not currently support them (e.g. those using the block decomposition, or those using byte-code).
  - To work around this, I have written an alternative pre-processor which generates a new .mod file by replacing each use of a model local variable by its definition.
  - The source and binary releases are available from:

    https://github.com/tholden/DynareRemoveLocalVariables

Details on the Dynare Transformation Engine

- The Dynare Transformation Engine (DTE) is a set of .mod files which, when `@#include`d into another .mod file, automatically transform variables so that they take values on the entire real line.
  - If the variable is only bounded on one side, this is done by first taking a linear transformation so that the new variable is defined on $[0, \infty)$, then taking logs.
  - If the variable is bounded both above and below, this is done by first taking a linear transformation so that the new variable is defined on $(0,1)$, then taking logits, where where $\text{logit}\, p = \log\left(\frac{p}{1-p}\right)$.

- When using the DTE, rather than declaring variables with `var`, we will define a pre-processor array of the form `EndoVariables=["VariableName1","Minimum1","Maximum1", "VariableName2","Minimum2","Maximum2", ...]`, where we allow "`Minimum*`" to take the special value "`-Inf`" and "`Maximum*`" to take the special value "`Inf`".
  - The DTE loops through the elements of this array, declaring the appropriately transformed variables.
  - It then generates model local variables defining the untransformed variables in terms of the transformed ones.
  - Finally, it defines the steady-state for the transformed variables in terms of the provided steady-state for the untransformed ones.

A surprising example of a variable bounded both above and below

- Consider a basic New Keynesian model, such as that of Fernández-Villaverde et al. (2012).
- In this model:
  - The price-level is given by: $P_t = \left[\int_0^1 P_{i,t}^{1-\varepsilon}\, di\right]^{\frac{1}{1-\varepsilon}}$.
  - A fraction $1-\theta$ of all firms update their price to P_t^* in period t.
- Hence: $P_t^{1-\varepsilon} = \theta P_{t-1}^{1-\varepsilon} + (1-\theta){P_t^*}^{1-\varepsilon}$.
- If we define gross inflation by $\Pi_t \coloneqq \frac{P_t}{P_{t-1}}$ and the relative reset price by $\Pi_t^* \coloneqq \frac{P_t^*}{P_t}$, then the evolution of inflation is governed by: $1 = \theta\Pi_t^{\varepsilon-1} + (1-\theta){\Pi_t^*}^{1-\varepsilon}$.
- Now, $\varepsilon > 1$ else the firm's optimal price decision is unbounded, so Π_t is maximised when $(1-\theta){\Pi_t^*}^{1-\varepsilon}$ is as small as possible (i.e. zero).
- Thus $\Pi_t < \theta^{\frac{1}{1-\varepsilon}}$!
  - $\theta = 0.75$, $\varepsilon = 6$ implies $\Pi_t < 1.06$.

Usage of the Dynare Transformation Engine

- The easiest way to learn how to use the Dynare Transformation Engine is to see an example.
- The DTE directory contains a slightly modified version of the NK model of Fernández-Villaverde et al. (2012).
  - The utility function is modified so that utility goes to $-\infty$ as $L_t \to 1$.
  - The laws of motion of β_t (the discount factor) and $S_{g,t}$ (the government spending share) are modified so that they are AR(1) in logits, rather than AR(1) in logs.

- The original .mod file is example_original.mod, and the version using the DTE is in example.mod.
  - To construct the DTE version, we (basically) just did a search and replace of `(+1)` for `_LEAD` and `(-1)` for `_LAG`.
  - Note that we had to slightly modify the `steady_state_model` block because Dynare does not allow us to introduce new variables in the `steady_state_model` block with the same name as MLVs.

- Compare the results of the transformed and untransformed files when σ_b is increased from 0.26 to 0.27.

Simulation of MLVs

- Having defined the variables of interest as model local variables, we then have to apply a non-linear transformation on the simulated data to recover them.

- Luckily DynareOBC will do this for us.
  - Independent of whether the MLVs came from the DTE or were manually added.
  - Independent of whether or not the model contains OBCs.

- Usage is straightforward:
  1. List the MLVs of interest after the `stoch_simul` command, just as you would normally list endogenous variables of interest.
  2. Invoke DynareOBC with the `MLVSimulationMode=1` option, and with `SlowIRFs` if IRFs are desired.

Calculation of expectations of MLVs using DynareOBC

- With `MLVSimulationMode=1` DynareOBC ignores any MLVs containing leads (i.e. +1 terms).
- With `MLVSimulationMode=2` DynareOBC calculates the expected value of any MLVs containing leads using the Genz and Keister (1996) sparse cubature rule.
  - The cubature is over the value of next period's shocks.
  - In this case, setting `MLVSimulationAccuracy=9` (for example) specifies that degree 9 polynomials should be integrated exactly.
- With `MLVSimulationMode=3` DynareOBC calculates the expected value of any MLVs containing leads using quasi-Monte Carlo.
  - In this case, setting `MLVSimulationAccuracy=9` (for example) specifies that $2^{1+9}-1$ points should be used.

Calculation of Euler equation errors using DynareOBC

- By defining MLVs for normalised equation errors and using these options, we can evaluate the accuracy of an approximation.
  - E.g. we define `#error1=(rhs1-lhs1)/lhs1;`, to assess the accuracy of the equation `lhs1=rhs1;`, where `lhs1` and `rhs1` are MLVs.
    - Note: for this to work correctly, all lead variables must be on the right hand side.
  - Have a go at doing this for example.mod versus example_original.mod.
    - Due to time constraints, you should probably remove the max operator from both.

- To speed up calculating equation errors, holding accuracy fixed, DynareOBC provides the option `MLVSimulationSubSample=4` (for example) which causes DynareOBC to only calculate MLVs every 4 samples.
  - This reduces the auto-correlation in successive MLV samples, giving lower sampling noise for the same number of samples.

Session outline: Day 2, Session 1 (1.5 hours)

- Theoretical results on existence, uniqueness and efficient computability.
- Testing model properties using DynareOBC.
- Equilibrium selection.
- Application to New Keynesian models.

Existence and uniqueness results: Holden (2016a) provides…

- Necessary and sufficient conditions for existence of a unique solution to models with OBCs (satisfying the terminal condition).
  - I will omit mentioning the terminal condition in the following.

- Necessary and sufficient conditions for existence of a unique solution when away from the bound.
  - E.g. suppose that the impulse response to some shock does not hit the bound. Must it be the unique solution?

- Some necessary conditions, and some sufficient conditions for existence of any solution.

- Sufficient conditions for the polynomial time computability of the solution.

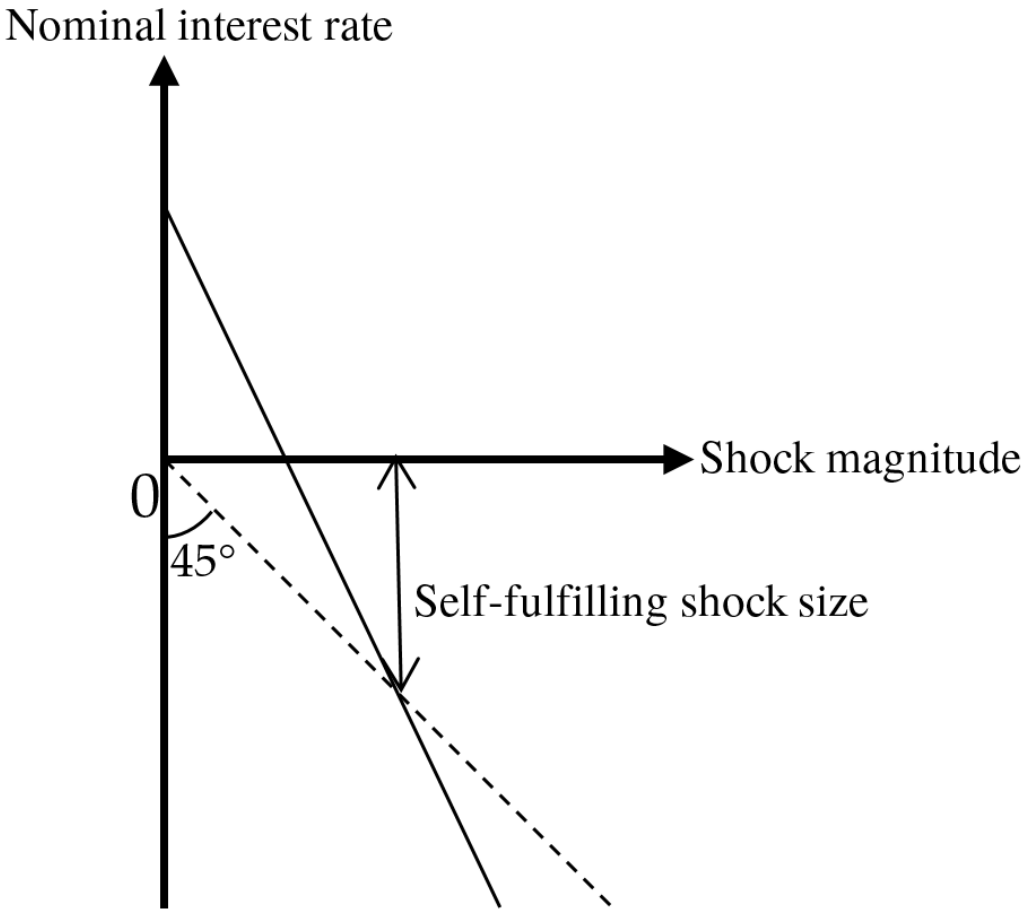
A preview of the application of our results to Smets and Wouters (2003; 2007)

- Augment the Smets and Wouters (2003) and (2007) models, with a zero lower bound, at their estimated modes.

- For both, there are combinations of predicted future shocks for which:
  - There are multiple solutions to the model, including combinations for which one solution features strictly positive nominal interest rates.
  - There are zero solutions to the model.
  - There is no known algorithm capable of finding a solution in time polynomial in the simulation horizon.

- Both models are determinate under price level targeting.

Intuition for multiplicity: Self-fulfilling news shocks



Is our M matrix special?

- The properties of solutions to LCPs are determined by the properties of the M matrix.
  - One might think that ours would have "nice" properties because of where it came from.

- Unfortunately:

- **Proposition:** For any matrix $\mathcal{M} \in \mathbb{R}^{T \times T}$, there exists a model in the form of Problem 2 with a number of state variables given by a quadratic in T, such that $M = \mathcal{M}$ for that model.

Introduction to the relevant matrix classes for the LCP

- Properties of solutions of LCPs have been extensively studied in the linear algebra and optimization literatures.

- As previously mentioned, all existence and uniqueness results are given in terms of the properties of the matrix M.

- Unfortunately, the required properties are rather harder to state (and check) than just looking at a few eigenvalues.

- In the next few slides, I give definitions of the key properties.
  - The chief references for the below are Cottle, Pang, and Stone (2009) and Xu (1993).

Principal sub matrices and principal minors

- For a matrix $M \in \mathbb{R}^{T \times T}$, the principal sub-matrices of M are the matrices:

$$\left\{ \left[M_{i,j} \right]_{i,j=k_1,\dots,k_S} \middle| S, k_1, \dots, k_S \in \{1, \dots, T\}, k_1 < k_2 < \cdots < k_S \right\},$$

  - i.e. the principal sub-matrices of M are formed by deleting the same rows and columns.

- The principal minors of M are the collection of values:

$$\left\{ \det\left(\left[M_{i,j} \right]_{i,j=k_1,\dots,k_S} \right) \middle| S, k_1, \dots, k_S \in \{1, \dots, T\}, k_1 < k_2 < \cdots < k_S \right\},$$

  - i.e. the principal minors of M are the determinants of the principal sub-matrices of M.

P($_0$)-matrices and General positive (semi-)definite matrices

- A matrix $M \in \mathbb{R}^{T \times T}$ is called a **P-matrix** if the principal minors of M are all strictly positive. M is called a **P_0-matrix** if the principal minors of M are all non-negative.
  - *Note: for symmetric M, M is a P($_0$)-matrix if and only if all of its eigenvalues are strictly (weakly) positive.*

- A matrix $M \in \mathbb{R}^{T \times T}$ is called **general positive definite** if $M + M'$ is a P-matrix. If $M + M'$is a P_0-matrix, then M is called **general positive semi-definite**.
  - *Note: that we do not require that M is symmetric in either case, but the definitions coincide with the standard ones for symmetric M.*

S($_0$)-matrices, (Strictly) Semi-monotone matrices and (Strictly) Copositive matrices

- A matrix $M \in \mathbb{R}^{T\times T}$ is called an **S-matrix** if there exists $y \in \mathbb{R}^T$ such that $y > 0$ and $My \gg 0$. M is called an **S_0-matrix** if there exists $y \in \mathbb{R}^T$ such that $y > 0$ and $My \geq 0$.

- A matrix $M \in \mathbb{R}^{T\times T}$ is called **strictly semi-monotone** if each of its principal sub-matrices is an **S-matrix**. M is called **semi-monotone** if each of its principal sub-matrices is an **S_0-matrix**.

- A matrix $M \in \mathbb{R}^{T\times T}$ is called **strictly copositive** if $M + M'$ is strictly semi-monotone. If $M + M'$ is semi-monotone then M is called **copositive**.

Sufficient matrices

- Let $M \in \mathbb{R}^{T \times T}$. M is called **column sufficient** if M is a P_0-matrix, and the principal sub-matrices of M with zero determinant satisfy a further technical condition.

- M is called **row sufficient** if M' is column sufficient.

- M is called **sufficient** if it is column sufficient and row sufficient.

Relationships between the matrix classes (Cottle, Pang and Stone 2009)

- All general p.s.d. matrices are copositive and sufficient.
- P_0 includes skew-symmetric matrices, general p.s.d. matrices, sufficient matrices and P-matrices.
- All P_0-matrices, and all copositive matrices are semi-monotone.
- All P-matrices, and all strictly copositive matrices are strictly semi-monotone (and hence S-matrices).

- All general p.s.d., semi-monotone, sufficient, P_0 and copositive matrices have non-negative diagonals.
- All general p.d., strictly semi-monotone, P and strictly copositive matrices have strictly positive diagonals.

Uniqueness results (1/3)

- We would ideally like a unique solution to exist for all possible q, since there are predicted shocks which can bring about any such q.

- **Proposition:** The LCP (q, M) has a unique solution for all $q \in \mathbb{R}^T$, if and only if M is a P-matrix.
- If M is not a P-matrix, then the LCP (q, M) has multiple solutions for some q.
- (Samelson, Thrall, and Wesler 1958; Cottle, Pang, and Stone 2009)

- This is the analogue of the Blanchard-Kahn conditions for models with occasionally binding constraints.
  - It tends to be satisfied in efficient models, but is rarely satisfied in New Keynesian models with zero lower bounds.

Uniqueness results (2/3)

- While for many models, the previous condition does not hold, we would hope that at least for $q \geq 0$ there ought to still be a unique solution.

- **Proposition:** The LCP (q, M) has a unique solution for all $q \in \mathbb{R}^T$ with $q \gg 0$ if and only if M is semi-monotone. (Cottle, Pang, and Stone 2009)

- **Proposition:** The LCP (q, M) has a unique solution for all $q \in \mathbb{R}^T$ with $q \geq 0$ if and only if M is strictly semi-monotone. (Cottle, Pang, and Stone 2009)

Uniqueness results (3/3)

- Thus, if M is not semi-monotone, there are some $q \gg 0$ such that the LCP (q, M) has multiple solutions.

- I.e., if agents today got appropriate signals about future shocks, then the economy could jump to the bound, even though the bound would not have been violated had it not been there at all.

- This remains true even if shocks are arbitrarily small, and even if the steady-state is arbitrarily far away from the bound.

Finite T existence results (1/4)

- Suppose $q \in \mathbb{R}^T$ and $M \in \mathbb{R}^{T \times T}$ are given. The LCP corresponding to M and q is called **feasible** if there exists $y \in \mathbb{R}^T$ such that $y \geq 0$ and $q + My \geq 0$.

- **Proposition:** The LCP (q, M) is feasible for all $q \in \mathbb{R}^T$ if and only if M is an S-matrix. (Cottle, Pang, and Stone 2009)

- If M is not an S-matrix, there are positive measure of q for which no solution exists.

Finite T existence results (2/4)

- **Proposition:** The LCP (q, M) is solvable if it is feasible and, either:
  - M is row-sufficient, or,
  - M is copositive and all the non-singular principal submatrices of M satisfy a further technical condition.
- (Cottle, Pang, and Stone 2009; Väliaho 1986)

- This gives sufficient conditions for existence for feasible q.
  - Checking feasibility just requires solving a linear programming problem, which is possible in time polynomial in T.

Finite T existence results (3/4)

- **Proposition**: The LCP (q, M) is solvable for all $q \in \mathbb{R}^T$, if at least one of the following conditions holds:
  - M is an S-matrix, and either of the conditions of the previous proposition are satisfied.
  - M is copositive with no zero principal minors.
  - M is a P-matrix, a strictly copositive matrix or a strictly semi-monotone matrix.
- (Cottle, Pang, and Stone 2009)

- This gives sufficient conditions for existence for all q.

Finite T existence results (4/4)

- In the special case in which M has nonnegative entries, we have both necessary and sufficient conditions:

- **Proposition:** If M is a matrix with nonnegative entries, then the LCP (q, M) is solvable for all $q \in \mathbb{R}^T$, if and only if M has a strictly positive diagonal. (Cottle, Pang, and Stone 2009)

Large T existence results

- Define:

$$\varsigma \coloneqq \sup_{\substack{y \in [0,1]^{\mathbb{N}^+} \\ \exists T \in \mathbb{N} \text{ s.t. } \forall t > T, y_t = 0}} \inf_{t \in \mathbb{N}^+} M_{t,1:\infty} y \,,$$

- Then M is an S-matrix for sufficiently large T if and only if $\varsigma > 0$.
- We show that there exists $\underline{\varsigma}_T, \overline{\varsigma}_T \geq 0$, both computable in time polynomial in T, such that $\underline{\varsigma}_T \leq \varsigma \leq \overline{\varsigma}_T$ and $\left|\underline{\varsigma}_T - \overline{\varsigma}_T\right| \to 0$ as $T \to \infty$.

- The proof relies on deriving constructive bounds on M.

Bounds on M

- **Lemma**
- The difference equation $A\hat{d}_{k+1} + B\hat{d}_k + C\hat{d}_{k-1} = 0$ for all $k \in \mathbb{N}^+$ has a unique solution satisfying the terminal condition $\hat{d}_k \to 0$ as $k \to \infty$, given by $\hat{d}_k = H\hat{d}_{k-1}$, for all $k \in \mathbb{N}^+$, for some H with eigenvalues in the unit circle.

- Define $d_0 \coloneqq -(AH + B + CF)^{-1} I_{\cdot,1}$, $d_k = Hd_{k-1}$, for all $k \in \mathbb{N}^+$, and $d_{-t} = Fd_{-(t-1)}$, for all $t \in \mathbb{N}^+$.
- The rows and columns of M are converging to 0 (with constructive bounds).
- The k^{th} diagonal of the M matrix is converging to the value $d_{1,k}$.
  - Diagonals are indexed such that the principal diagonal is index 0, and indices increase as one moves up and to the right in the M matrix.

Results from dynamic programming (1/2)

- Here we introduce a class of problems that our algorithm will solve arbitrarily accurately, and give alternative uniqueness results.

- **Problem 5**
- Solve a concave quadratic dynamic programming problem subject to linear inequality constraints. (Full definition in the paper.)

- **Proposition:** If either:
  - $\tilde{\Gamma}(x)$ (the choice set) is compact valued and $x \in \tilde{\Gamma}(x)$ for all $x \in \tilde{X}$, or,
  - $\tilde{X}$ (the state space) is compact,
- then for all $x_0 \in \tilde{X}$, there is a unique path $(x_t)_{t=0}^{\infty}$ which solves Problem 5.

Results from dynamic programming (2/2)

- **Proposition:**
- The KKT conditions of Problem 5 may be placed into the form of the multiple-bound generalisation of Problem 2.
- Let $\left(q_{x_0}, M\right)$ be the infinite LCP corresponding to this representation, given initial state $x_0 \in \tilde{X}$.
- If y is a solution to the LCP, $q_{x_0} + My$ gives the stacked paths of the bounded variables in a solution to Problem 5.
- If, further, either condition of the previous proposition holds, then:
  - Both Problem 5 and this LCP have a unique solution for all $x_0 \in \tilde{X}$.
  - For sufficiently large T, the finite LCP also has a unique solution.

Testing membership of the matrix classes in DynareOBC (1/2)

- You will have noticed that DynareOBC reports whether M is an S-matrix (with the finite T given by `TimeToEscapeBounds`).

- To test if M is an S-matrix for arbitrarily high T, invoke DynareOBC with e.g. `FeasibilityTestGridSize=10` and a large value for `TimeToEscapeBounds`.
  - Increasing `FeasibilityTestGridSize` makes it more likely that DynareOBC will manage to prove feasibility, but it may also be slow.

Testing membership of the matrix classes in DynareOBC (2/2)

- DynareOBC also automatically checks a few easy to verify necessary conditions to be a P-matrix.
  - If any of these fail, DynareOBC reports that M is not a P-matrix.
  - If all of these succeed, M may or may not be a P-matrix.

- To test if the top (e.g.) 5x5 sub-matrix of M is actually a P-matrix, run DynareOBC with the `PTest=5` option.
  - Testing large sub-matrices will be incredibly slow.

- To test if the top (e.g.) 5x5 sub-matrix of M is a P_0 and/or (strictly) semi-monotone matrix, run DynareOBC with the `FullTest=5` option.
  - This is even slower than the `PTest` option.

Results on the computational complexity of the problem (1/2)

- If M is unrestricted, or M is a P_0-matrix, then finding a single solution to the LCP (q, M) is "strongly NP complete".

- If we could do this efficiently (i.e. in polynomial time), we could also solve in polynomial time any problem whose solution could be efficiently verified.
  - This includes, for example, breaking all standard forms of cryptography.

- Since there is a model corresponding to any M matrix, with quadratic in T states, this means that if there were a solution algorithm for DSGE models with occasionally binding constraints that worked in time polynomial in the number of states, then it could also be used to defeat all known forms of cryptography.

Results on the computational complexity of the problem (2/2)

- Polynomial time algorithms exist for special cases, but checking whether the relevant ones apply is not possible in polynomial time.

- This means that there cannot be an algorithm for checking if a model e.g. has a unique solution, that runs in time polynomial in the number of states.

Our computational approach to the perfect foresight problem (1/2)

- There is no way of escaping solving an NP-complete problem if we wish to simulate DSGE models with OBCs.

- Any algorithm we invented for the problem is likely to be inefficient, and possibly even non-finite.

- A better approach is to map our problem into another to which smart computer scientists have devoted a lot of time.

- It turns out that the solution to an LCP can be represented as a mixed integer linear programming problem.
  - One of the best studied problems in computer science.
  - Extremely well optimised, fully global, solvers exist.

Our computational approach to the perfect foresight problem (2/2)

- **Problem 7**
- Suppose $\widetilde{\omega} > 0$, $q \in \mathbb{R}^T$ and $M \in \mathbb{R}^{T \times T}$ are given.
- Find $\alpha \in \mathbb{R}$, $\hat{y} \in \mathbb{R}^T$, $z \in \{0,1\}^T$ to maximise α subject to the following constraints: $\alpha \geq 0$, $0 \leq \hat{y} \leq z$, $0 \leq \alpha q + M\hat{y} \leq \widetilde{\omega}(1_{T \times 1} - z)$.

- **Proposition**: If α, $\hat{y}$, z solve Problem 7, then if $\alpha = 0$, the LCP (q, M) has no solution, and if $\alpha > 0$, then $y \coloneqq \frac{\hat{y}}{\alpha}$ solves it. (Partial converse in paper.)

- As $\widetilde{\omega} \to 0$, the solution to Problem 7 is the solution to the LCP which minimises $\|q + My\|_\infty$.
- As $\widetilde{\omega} \to \infty$, the solution to Problem 7 is the solution to the LCP which minimises $\|y\|_\infty$.

Equilibrium selection with DynareOBC

- DynareOBC sets $\widetilde{\omega} = \omega \|q\|_\infty$, where $\omega > 0$ is another constant.
- ω may be set in DynareOBC by invoking DynareOBC with the `omega=1000` (for example) option.
- Setting $\omega = 1000$ is the default, which places a fairly high penalty on large news shocks, and hence will tend to find solutions with less time at the bound.
- For the other extreme, try $\omega = 0.0001$.
- We will see some examples very shortly.

Examples from New Keynesian models

- First look at a three equation NK model with a response to output growth, following Brendon, Paustian, and Yates (BPY) (2015) .
- Then assorted variants of this model.
- Then the Fernandez-Villaverde et al. (2012) model with price dispersion.
- Then Smets Wouters (2003) and (2007).

Simple Brendon, Paustian, and Yates (BPY) (2015) model (1/3)

- Equations:

$$x_{i,t} = \max\{0, 1 - \beta + \alpha_{\Delta y}(x_{y,t} - x_{y,t-1}) + \alpha_\pi x_{\pi,t}\},$$
$$x_{y,t} = \mathbb{E}_t x_{y,t+1} - \frac{1}{\sigma}(x_{i,t} + \beta - 1 - \mathbb{E}_t x_{\pi,t+1}),$$
$$x_{\pi,t} = \beta \mathbb{E}_t x_{\pi,t+1} + \gamma x_{y,t},$$

- $\beta \in (0,1)$, $\gamma, \sigma, \alpha_{\Delta y} \in (0, \infty)$, $\alpha_\pi \in (1, \infty)$.
- Unique stationary solution.
- If $T = 1$, then:

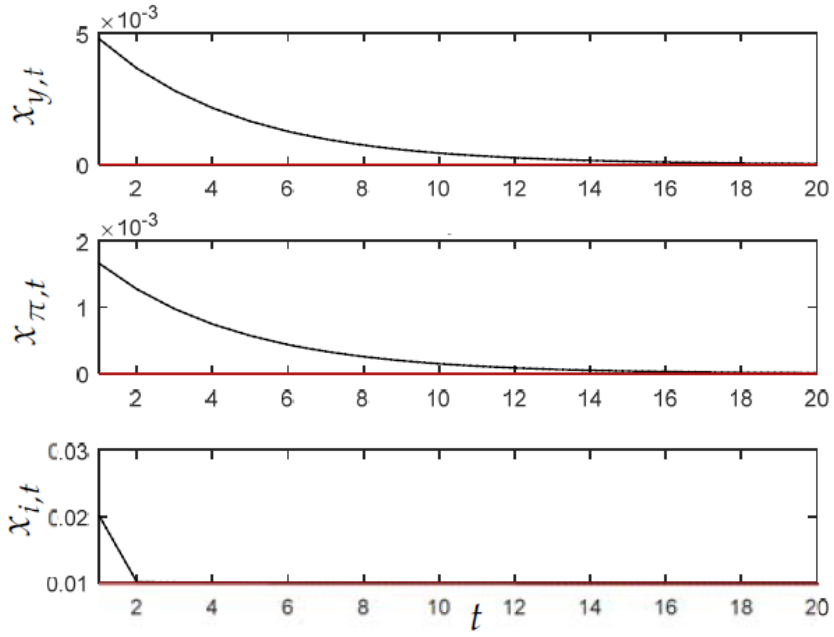
$$M = \frac{\beta\sigma f^2 - ((1+\beta)\sigma + \gamma)f + \sigma}{\beta\sigma f^2 - ((1+\beta)\sigma + \gamma + \beta\alpha_{\Delta y})f + \sigma + \alpha_{\Delta y} + \gamma\alpha_\pi},$$

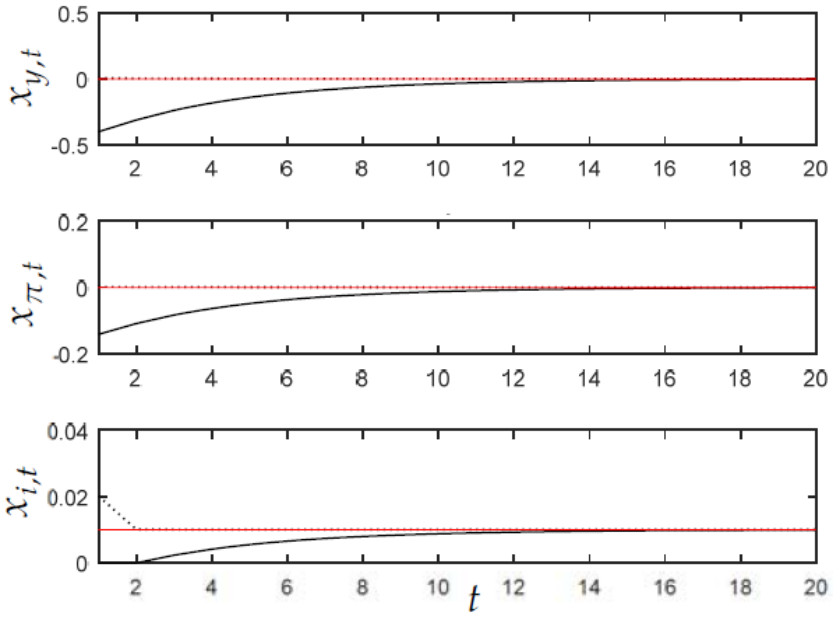
- M is negative if and only if $\alpha_{\Delta y} > \sigma\alpha_\pi$. M is zero if and only if $\alpha_{\Delta y} = \sigma\alpha_\pi$.

Simple BPY (2015) model (2/3)

- When $T = 1$:
  - If $\alpha_{\Delta y} < \sigma\alpha_{\pi}$ then the model has a unique solution for all q.
  - When $\alpha_{\Delta y} > \sigma\alpha_{\pi}$, for any positive q, there exists $y > 0$ such that $q + My = 0$, so the model has multiple solutions.
  - When $\alpha_{\Delta y} > \sigma\alpha_{\pi}$, for any negative q, there is no $y \geq 0$ such that $q + My \geq 0$, so the model has no solutions.
- When $T > 1$:
  - If $\alpha_{\Delta y} > \sigma\alpha_{\pi}$ then at least for some $q \gg 0$, the model has multiple solutions.

Simple BPY (2015) model (3/3)





Minimum $\|y\|_\infty$ solution[12] **Minimum $\|q + My\|_\infty$ solution[13]**

Figure 1: Alternative solutions following a magnitude 1 impulse to ε_t

BPY (2015) model with shadow interest rate persistence (1/3)

- Equations:

$$x_{i,t} = \max\{0, x_{d,t}\},$$
$$x_{d,t} = (1-\rho)\left(1-\beta+\alpha_{\Delta y}\left(x_{y,t}-x_{y,t-1}\right)+\alpha_{\pi}x_{\pi,t}\right)+\rho x_{d,t-1},$$
$$x_{y,t} = \mathbb{E}_t x_{y,t+1} - \frac{1}{\sigma}\left(x_{i,t}+\beta-1-\mathbb{E}_t x_{\pi,t+1}\right),$$
$$x_{\pi,t} = \beta\mathbb{E}_t x_{\pi,t+1} + \gamma x_{y,t}.$$

- Set $\sigma = 1, \beta = 0.99, \gamma = \frac{(1-0.85)\left(1-\beta(0.85)\right)}{0.85}(2+\sigma), \rho = 0.5.$

BPY (2015) model with shadow interest rate persistence (2/3)

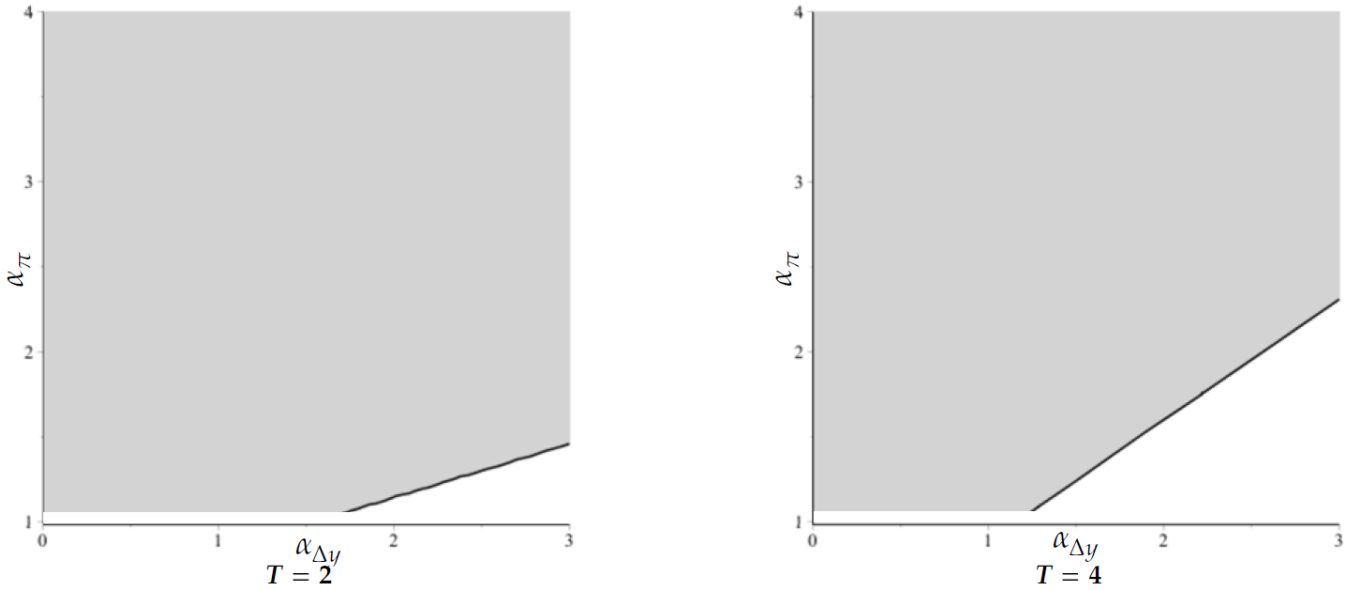


Figure 1: Regions in which M is a P-matrix (shaded grey) or a P_0-matrix (shaded grey, plus the black line), when $T = 2$ (left) or $T = 4$ (right).

BPY (2015) model with shadow interest rate persistence (3/3)

- With large T, it appears that the determinacy region agrees with that in the model without persistence in the shadow rate.

- Evidence for this is provided by the fact that with $\alpha_\pi = 1.5$:

  - If $\alpha_{\Delta y} = 1.51$, and $T = 200$ then 1) M is not an S-matrix, so it is not a P-matrix for higher T and 2) $\varsigma \leq 0 +$ num. err., providing numerical evidence that for sufficiently large T, the LCP (q, M) is not feasible for some q.

  - If $\alpha_{\Delta y} = 1.05$, then M is a P-matrix, and $\varsigma > 6.1319 \times 10^{-8}$, so M is an S-matrix for all sufficiently large T.

BPY (2015) model with price level targeting (1/2)

- Equations:

$$x_{i,t} = \max\{0, 1 - \beta + \alpha_{\Delta y} x_{y,t} + \alpha_{\pi} x_{p,t}\},$$

$$x_{y,t} = \mathbb{E}_t x_{y,t+1} - \frac{1}{\sigma}\left(x_{i,t} + \beta - 1 - \mathbb{E}_t x_{p,t+1} + x_{p,t}\right),$$

$$x_{p,t} - x_{p,t-1} = \beta \mathbb{E}_t x_{p,t+1} - \beta x_{p,t} + \gamma x_{y,t},$$

- Determinacy in the absence of the ZLB requires $\alpha_{\pi} \in (0, \infty)$, $\alpha_{\Delta y} \in [0, \infty)$.

BPY (2015) model with price level targeting (2/2)

- With $T = 1$, M is strictly positive for all $\alpha_{\Delta y}, \alpha_{\pi} \in (0, \infty)$, so price level targeting cures the multiplicity found by BPY.

- With parameters as before, and $\alpha_{\Delta y} = 1$, $\alpha_{\pi} = 1$:

  - From our lower bound on ς with $T = 20$, we find that $\varsigma \geq 0.042659$. Hence, this model is always feasible for any sufficiently large T.

  - Given that $d_0 > 0$ for this model, and that for $T = 20$, M is a P-matrix, this is strongly suggestive of the existence of a unique solution for any q and T.

Linearized Fernandez-Villaverde et al. (2012) model

- A basic NK model without investment, but with positive steady-state inflation, and hence price dispersion.

- With $T \leq 14$, M is a P-matrix, but with $T \geq 15$, M is not a P-matrix.
  - Thus this model always has multiple solutions.

- With $T = 1000$, from our upper bound on ς, we have that: $\varsigma \leq 0 +$ numerical error.
  - Provides strong evidence that M is not an S-matrix for large T either.

Linearized Fernandez-Villaverde et al. (2012) model with price level targetting

- With nominal GDP targeting (unit coefficients), with $T = 200$, our lower bound gives $\varsigma > 0.0048175$.
  - Hence, for all sufficiently large T, M is an S-matrix, so there is always a feasible solution.

- For all T tested, M is a P-matrix.

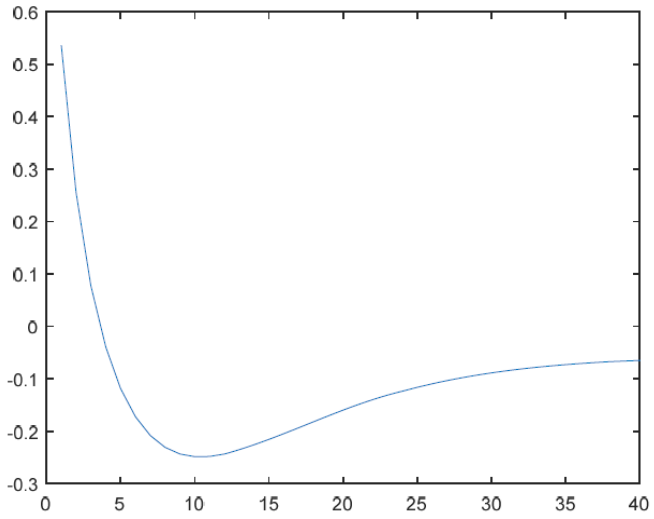
Smets & Wouters (2003; 2007) (1/3)

- Both models have:
  - assorted shocks, habits, price and wage indexation, capital (with adjustment costs), (costly) variable utilisation, general monetary policy reaction functions
- We augment both models with nominal interest rate rules of the form:

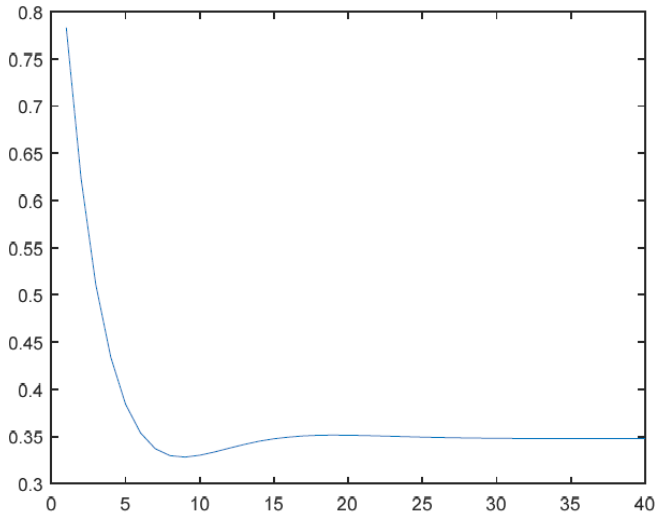
$$r_t = \max\{0, (1-\rho)(\cdots) + \rho r_{t-1} + \cdots\}$$

- Recall that the 2003 model is estimated on Euro area data, and the 2007 one is estimated on US data.
  - We use the posterior modes.
- Fairly similar models, except that the 2007 one:
  - Contains trend growth (permitting its estimation on non-detrended data),
  - Has a slightly more general aggregator across industries.

Smets & Wouters (2003; 2007) (2/3)



The Smets and Wouters (2003) model



The Smets and Wouters (2007) model

Figure 2: The diagonals of the M matrices for the Smets and Wouters (2003) and Smets and Wouters (2007) models

Smets & Wouters (2003; 2007) (3/3)

- Perhaps surprising that these graphs are so different.
- Negative diagonal for the Euro area model implies that the model does not always have a unique solution, even when $q \gg 0$.
- In fact, providing $T \geq 9$, the US model also does not always have a unique solution, even when $q \gg 0$.
- Furthermore, for both countries, there are some q for which the model has no solution.
  - Suggests that only solutions converging to the "bad" steady-state exist in those cases.
- Increasing the coefficient on inflation results in a positive diagonal for M even for the Euro area model, but does not result in a P-matrix.
- However, replacing inflation by the price-level minus a linear trend in the Taylor rule produces a P-matrix.

Session outline: Day 2, Session 2 (1.5 hours)

- The cubature Kalman filter.
- Estimating using the Cubature Kalman filter in DynareOBC.
- Set-up of the Holden Unstable Dynare version.
- Estimating using the Cubature Kalman filter in Dynare.

Introduction to the estimation section of the course

- In the remainder of the course, we will cover a variety of nonlinear, full-information, structural estimation methods that are designed to be computationally tractable even on medium scale models.

- Although the particle filter is a full-information, structural estimation method, it is not really computationally tractable on medium scale models.
  - So we won't waste time discussing it.

- In the Surrey CIMS Advanced macro course I discuss the particle filter as well as assorted partial information estimation methods.

- I start by discussing the Cubature Kalman Filter which is the only method supported by DynareOBC.
- After the break I will discuss a few other options in my custom version of Unstable Dynare.

Set-up of the Cubature Kalman Filter (CKF) (Arasaratnam and Haykin 2009; Holden 2016c)

- The solution to a general non-linear DSGE model takes the following form for all $t \in \mathbb{N}^+$:

$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = f(x_{t-1}, \varepsilon_t),$$

- where:
  - x_t are the n_x states,
  - y_t are the n_y controls appearing in the measurement equation,
  - z_t are the n_z other controls,
  - ε_t are the n_ε structural shocks, with $\varepsilon_t \sim \mathrm{NIID}(0, I)$ without loss of generality.
- We suppose we observe:

$$m_t = g\left(\begin{bmatrix} x_t \\ y_t \end{bmatrix}\right) + \nu_t,$$

- where $\nu_t \sim \mathrm{NIID}(0, \Lambda)$.
  - An additive Gaussian shock independent of ε_t is without loss of generality as other shocks may be incorporated into y_t.

Key trick for estimating with the CKF

- As ever in estimation, we wish to know the likelihood as a function of the structural parameters θ.

- By the usual predictive error decomposition, the likelihood is given by:

$$p(m_T, \dots, m_1|\theta) = p(m_T|m_{T-1}, \dots, m_1, \theta)p(m_{T-1}, \dots, m_1|\theta) = \cdots$$
$$= \prod_{t=1}^{T} p(m_t|\mathcal{F}_{t-1}, \theta)\,,$$

- where $\mathcal{F}_t \coloneqq (m_t, \dots, m_1)$ is the period t information set.

- The key trick behind the CKF is approximating the distribution of $x_t|\mathcal{F}_t$ and $\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix}|\mathcal{F}_t$ by Gaussians.

CKF predict step (1/2)

- Suppose $x_{t-1}|\mathcal{F}_{t-1} \sim \mathrm{N}\left(\hat{x}_{t-1|t-1}, S_{t-1|t-1}S'_{t-1|t-1}\right)$.
  - Also define $k_{t-1|t-1} \coloneqq \operatorname{cols} S_{t-1|t-1}$.
- Note that if $\xi \in \mathbb{R}^{k_{t-1|t-1}}$ is a draw from $\mathrm{N}(0, I)$, then $\hat{x}_{t-1|t-1} + S_{t-1|t-1}\xi$ has the same distribution as $x_{t-1}|\mathcal{F}_{t-1}$.
- Hence:

$$\mathbb{E}\left[\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \middle| \mathcal{F}_{t-1}\right] = \int_{\mathbb{R}^{n_\varepsilon}} \int_{\mathbb{R}^{k_{t-1|t-1}}} f\left(\hat{x}_{t-1|t-1} + S_{t-1|t-1}\xi, \varepsilon\right) \phi_{k_{t-1|t-1}+n_\varepsilon}\left(\begin{bmatrix} \xi \\ \varepsilon \end{bmatrix}\right) d\xi \, d\varepsilon \,,$$

$$\operatorname{var}\left[\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \middle| \mathcal{F}_{t-1}\right] = \int_{\mathbb{R}^{n_\varepsilon}} \int_{\mathbb{R}^{k_{t-1|t-1}}} a_{t|t-1}(\xi, \varepsilon) a_{t|t-1}(\xi, \varepsilon)' \phi_{k_{t-1|t-1}+n_\varepsilon}\left(\begin{bmatrix} \xi \\ \varepsilon \end{bmatrix}\right) d\xi \, d\varepsilon \,,$$

- where $a_{t|t-1}(\xi, \varepsilon) \coloneqq f\left(\hat{x}_{t-1|t-1} + S_{t-1|t-1}\xi, \varepsilon\right) - \mathbb{E}\left[\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \middle| \mathcal{F}_{t-1}\right]$,
- and $\phi_k \colon \mathbb{R}^k \to \mathbb{R}^+$ is the p.d.f. of a standard k-dimensional normal variable.

CKF predict step (2/2)

- We have two Gaussian integrals to evaluate.
- We do this using either a degree three monomial cubature rule, or using the Genz and Keister (1996) rule.
- Both integrals may be evaluated without duplicating the simulation step (the evaluation of f) by first evaluating f at the sample points, then calculating the first integral, then substituting into $a_{t|t-1}(\xi, \varepsilon)$ to calculate the second integral.
- We may then define:

$$\hat{x}_{t|t-1} :\approx \mathbb{E}[x_t|\mathcal{F}_{t-1}],$$
$$\hat{y}_{t|t-1} :\approx \mathbb{E}[y_t|\mathcal{F}_{t-1}],$$
$$P_{t|t-1} :\approx \operatorname{var}\left[\begin{bmatrix} x_t \\ y_t \end{bmatrix} \middle| \mathcal{F}_{t-1}\right],$$

- where "$:\approx$" is a shorthand for "is defined to be an approximation to".
- Finally, we approximate the distribution of $\begin{bmatrix} x_t \\ y_t \end{bmatrix} | \mathcal{F}_{t-1}$ by $\mathrm{N}\left(\begin{bmatrix} \hat{x}_{t|t-1} \\ \hat{y}_{t|t-1} \end{bmatrix}, P_{t|t-1}\right)$.

CKF update step (1/3)

- Let $U_{t|t-1} D_{t|t-1} U'_{t|t-1}$ be the Schur decomposition of $P_{t|t-1}$.
  - $U_{t|t-1}$ is orthogonal, and $D_{t|t-1}$ is diagonal and weakly positive.
- Let $d_{t|t-1} \coloneqq \operatorname{diag} D_{t|t-1}$.
- WLOG, suppose that only the $k_{t|t-1}$ first elements of $d_{t|t-1}$ are bigger than some threshold κ, (e.g. $\kappa \coloneqq 10^{-6}$).
- Let $U_{t|t-1,\cdot 1}$ be the first $k_{t|t-1}$ columns of $U_{t|t-1}$, and $d_{t|t-1,1}$ be the first $k_{t|t-1}$ rows of $d_{t|t-1}$.
- Then: $P_{t|t-1} \approx U_{t|t-1,\cdot 1} \operatorname{diag} d_{t|t-1,1} \, U'_{t|t-1,\cdot 1}$.
  - This will reduce the dimension of the space we have to integrate over.
  - It also improves the performance of the integration rules.
- Finally, set: $S_{t|t-1} \coloneqq U_{t|t-1,\cdot 1} \operatorname{diag} \sqrt{d_{t|t-1,1}}$
- Hence, if $\eta \in \mathbb{R}^{k_{t|t-1}}$ is a draw from $\mathrm{N}\left(0_{k_{t|t-1}}, I_{k_{t|t-1} \times k_{t|t-1}}\right)$, then the distribution of $\begin{bmatrix} \hat{x}_{t|t-1} \\ \hat{y}_{t|t-1} \end{bmatrix} + S_{t|t-1}\eta$ is approximately equal to that of $\begin{bmatrix} x_t \\ y_t \end{bmatrix} | \mathcal{F}_{t-1}$.

CKF update step (2/3)

- Consequently:

$$\mathbb{E}[m_t|\mathcal{F}_{t-1}] \approx \int_{\mathbb{R}^{k_{t|t-1}}} g\left(\begin{bmatrix}\hat{x}_{t|t-1}\\ \hat{y}_{t|t-1}\end{bmatrix} + S_{t|t-1}\eta\right)\phi_{k_{t|t-1}}(\eta)\,d\eta\,,$$

$$\mathrm{var}[m_t|\mathcal{F}_{t-1}] \approx \int_{\mathbb{R}^{k_{t-1|t-1}}} b_{t|t-1}(\eta)b_{t|t-1}(\eta)'\phi_{k_{t|t-1}}(\eta)\,d\eta + \Lambda,$$

$$\mathrm{cov}[x_t, m_t|\mathcal{F}_{t-1}] \approx \int_{\mathbb{R}^{k_{t-1|t-1}}} S_{t|t-1,1}\eta b_{t|t-1}(\eta)'\phi_{k_{t|t-1}}(\eta)\,d\eta\,,$$

  - where $S_{t|t-1,1}$ is the top n_x rows of $S_{t|t-1}$ and $b_{t|t-1}(\eta) \coloneqq g\left(\begin{bmatrix}\hat{x}_{t|t-1}\\ \hat{y}_{t|t-1}\end{bmatrix} + S_{t|t-1}\eta\right) - \mathbb{E}[m_t|\mathcal{F}_{t-1}]$.

- As before these Gaussian integrals may be simultaneously approximated by standard cubature methods.
- We then define:

$$\begin{aligned}\widehat{m}_{t|t-1} &\coloneqq \approx \mathbb{E}[m_t|\mathcal{F}_{t-1}],\\ Q_{t|t-1} &\coloneqq \approx \mathrm{var}[m_t|\mathcal{F}_{t-1}]\,,\\ R_{t|t-1} &\coloneqq \approx \mathrm{cov}[x_t, m_t|\mathcal{F}_{t-1}]\,.\end{aligned}$$

- So $\begin{bmatrix}x_t\\ m_t\end{bmatrix}|\mathcal{F}_{t-1}$ is approximately distributed as $\mathrm{N}\left(\begin{bmatrix}\hat{x}_{t|t-1}\\ \widehat{m}_{t|t-1}\end{bmatrix}, \begin{bmatrix}P_{t|t-1,11} & R_{t|t-1}\\ R'_{t|t-1} & Q_{t|t-1}\end{bmatrix}\right)$,
  - where $P_{t|t-1,11}$ is the upper left $n_x \times n_x$ block of $P_{t|t-1}$.

CKF update step (3/3)

- We want to know the distribution of $x_t|\mathcal{F}_t$, but by the definition of $\mathcal{F}_t$, this is just equal to the distribution of $x_t|m_t, \mathcal{F}_{t-1}$.
- Hence, if we define:

$$\hat{x}_{t|t} \coloneqq \hat{x}_{t|t-1} + R_{t|t-1} Q_{t|t-1}^{-1} \left(m_t - \hat{m}_{t|t-1}\right),$$
$$P_{t|t} \coloneqq P_{t|t-1,11} - R_{t|t-1} Q_{t|t-1}^{-1} R'_{t|t-1},$$

- then $x_t|\mathcal{F}_t$ is approximately distributed as $\mathrm{N}\left(\hat{x}_{t|t}, P_{t|t}\right)$.

- As before we can find $U_{t|t}$ (orthogonal) and $d_{t|t}$ such that $P_{t|t} = U_{t|t} \operatorname{diag} d_{t|t} \, U'_{t|t}$, and then set $S_{t|t} \coloneqq U_{t|t,\cdot 1} \operatorname{diag} \sqrt{d_{t|t,1}}$,
  - where $d_{t|t,1}$ contains the $k_{t|t}$ elements of $d_{t|t}$ that are greater than κ.
- After this, we have that the distribution of $x_t|\mathcal{F}_t$ is approximately $\mathrm{N}\left(\hat{x}_{t|t}, S_{t|t} S'_{t|t}\right)$, which completes one time-step.

- Finally, we also have that:

$$\log p(m_t|\mathcal{F}_{t-1}) = -\frac{1}{2} \log \left|Q_{t|t-1}\right| - \frac{1}{2} \left(m_t - \hat{m}_{t|t-1}\right)' Q_{t|t-1}^{-1} \left(m_t - \hat{m}_{t|t-1}\right) - \frac{n_m}{2} \log 2\pi \,.$$

Obtaining the initial distribution of the state

- DynareOBC obtains the initial distribution of the state by finding the distribution of the state from the CKF following an infinite run of empty observations.
  - This is more natural than using the true stationary distribution, as with the true stationary distribution, empty observations would change the state distribution.

- This is solved as a fixed point iteration problem.

- The number of fixed point iterations may be adjusted by invoking DynareOBC with e.g.

```
EstimationFixedPointMaxIterations=1000.
```

The CKF in DynareOBC

- DynareOBC implements the CKF for models solved up to third order, with or without occasionally binding constraints.

- At present only maximum likelihood (ML) and maximum a posteriori (MAP) estimation is supported.
  - MCMC is likely to be prohibitively expensive on models with occasionally binding constraints.
  - Many papers supposedly doing Bayesian estimation actually base all conclusions on the posterior mode i.e. the MAP estimate.

- The prior is specified by invoking DynareOBC with e.g. `EstimationPrior=MyPrior`, where MyPrior is a function which takes in the parameter vector and returns the log density of the prior at that point (up to a constant).
  - DynareOBC defaults to using a flat prior, giving ML estimates.

The data file for the CKF in DynareOBC

- To turn on estimation, invoke DynareOBC with the `estimation` option.
- By default, DynareOBC looks for a data file name ModFileName.xlsx.
- To specify a different file, invoke DynareOBC with the `EstimationDataFile=MyDataFile.xlsx` option.

- The data file should have two sheets.
- Sheet one should contain the data in columns, with a header row giving the name of the MLV giving the particular observable.
  - There should not be a header column with dates.
  - Empty cells are interpreted as missing observations.
- Sheet two should contain three rows.
  - The first row should give the names of parameters to be entered.
  - The second row should give the lower bound on those parameters, with empty cells being interpreted as $-\infty$.
  - The third row should give the upper bounds on those parameters, with empty cells being interpreted as ∞.

DynareOBC options controlling likelihood evaluation

- `EstimationStdDevThreshold=FLOAT` (default: `1e-6`)
  - Specifies κ, the threshold below which the standard deviation of the state is set to zero, for dimension reduction.
- `EstimationPredictSparseCubatureDegree=INTEGER` (default: `0`)
  - If this is greater than zero, then DynareOBC uses the Genz and Keister (1996) integration rule for evaluating the integrals in the predict step.
  - The rule exactly integrates a polynomial of degree INTEGER.
  - In a model without bounds, there is no need to have INTEGER larger than twice the order of approximation.
- `EstimationUpdateSparseCubatureDegree=INTEGER` (default: `0`)
  - If this is greater than zero, then DynareOBC uses the Genz and Keister (1996) integration rule for evaluating the integrals in the update step.
  - The rule exactly integrates a polynomial of degree INTEGER.
  - 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.

DynareOBC minimisation (etc.) options

- `EstimationMinimisationFunctions=STRING`

  (default: `CMAESWrapper#FMinConWrapper`)
  - A # delimitated 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).
- `EstimationFixedParameters=STRING` (default: '')
  - A # delimitated 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.
- `EstimationTimeOutLikelihoodEvaluation=INTEGER` (defaut: 60)
  - Any likelihood evaluations that take longer than this number of seconds will be terminated prematurely. Beware that this may bias parameter estimates.
- `EstimationSkipStandardErrors`
  - Makes DynareOBC skip calculation of standard errors for the estimated parameters.

Simulation options with DynareOBC estimation

- DynareOBC's estimation options can be combined with any of its simulation options.
- For reasonable running times, `nocubature` is probably a good idea, but Holden (2016c) does illustrate the improvement in estimates that come from using better cubature methods.

- In large models, it may be a good idea to use FirstOrderAroundMean, which (you recall) takes a first order approximation around the model's mean (where the mean may be calculated with a third order approximation).
  - For models without OBCs, this produces a linear model that still captures e.g. risk effects and stochastic volatility.
  - Estimating linear models this way was suggested by Meyer-Gohde (2014), and we will discuss using it for linear models via Dynare later today.
  - For models with OBC, this helps as it reduces the integration dimension, significantly speeding up estimation.

DynareOBC estimation example

- There is an example of estimating a simple model with an OBC in the Examples\BoundedProductivityEstimation folder of DynareOBC.

- As with most of the DynareOBC examples folders, full replication details are given in the ReplicationDetails.txt file.

- We will go through some of these examples.

The CKF in Dynare

- As an alternative to using DynareOBC, you can also use a version of the CKF from within Dynare.
- Relative to the DynareOBC version this has a few disadvantages:
  - It only supports using a second order approximation for the model.
  - It also takes a second order approximation to the observation equations which is unnecessary.
  - It does not support occasionally binding constraints.
  - It sets the covariance of the initial state to the variance in a first order approximation to the model.
  - It does not perform dimension reduction, which means that at high orders it is integrating over a very high dimensional space.
  - Integrating over a high dimensional space results in worse performance of the cubature rules.
  - The covariance matrix of the state is not guaranteed to remain positive semi-definite when some cubature rules.
  - In “off the shelf” Dynare, it does not support evaluating the likelihood in parallel, to speed up derivative calculation.
- However, it does support full Bayesian integration with MCMC.

The Holden version of unstable Dynare (1/2)

- I maintain an alternative version of Dynare unstable with assorted additional features.
  - Given that the interface to the CKF (and particle filters) has changed substantially from Dynare 4.4.3 to Unstable, it makes sense to use Dynare Unstable in any case, and given this, I suggest using my version.
- My version is available from: https://github.com/tholden/dynare
- As with DynareOBC, the recommended way to get it on your computer is:
  1. Download and install GitHub Desktop, from https://desktop.github.com/. (Note that this does not require admin rights!).
  2. Click on the "clone or download" button on the DynareOBC website.
  3. Click "open in desktop".
- If you cannot install GitHub desktop due to machine restrictions, you may still be able to install PotableGit from here:

  https://github.com/git-for-windows/git/releases

  - Then, using the git command prompt, type:

```
git clone https://github.com/tholden/dynare --recurse-submodules
```

The Holden version of unstable Dynare (2/2)

- Alternatively, there is a release made especially for this course available from https://github.com/tholden/dynare/releases.
  - Unlike DynareOBC, Dynare does not auto update, so I recommend you switch to Git when you can.

- For the CKF, the chief improvement is supporting estimation in parallel, which helps given the slower likelihood evaluations.
- My version of Dynare Unstable also contains a variety of examples of nonlinear estimation using computationally tractable methods.
  - These examples are in the examples\nonlinear-estimation directory.

- We will go through the Dynare and the DynareOBC versions of the CKF:

  examples\nonlinear-estimation\NKCKFEstimate.mod

  examples\nonlinear-estimation\NKDynareOBCEstimate.mod
- Examining these files is the best way of seeing the required options.

Session outline: Day 2, Session 3 (1 hour)

- The Kollman (2013) non-linear estimation approach.
- Estimating using the Kollman approach in the Holden Unstable Dynare version.
- The Meyer-Gohde (2014) non-linear estimation approach.
- Estimating using the Meyer-Gohde approach in the Holden Unstable Dynare version.

The Kollmann trick for non-linear DSGE models (1/3)

- Kollmann (2013) showed that the Kalman filter could be used for approximate filtering of pruned perturbation approximations.
- Recall that the second order pruned perturbation approximation (Kim et al. 2008) to a DSGE model takes the form:

$$x_{1,t} = A_1 x_{1,t-1} + \varepsilon_t,$$

$$x_{2,t} = A_0 + A_1 x_{2,t-1} + A_{11}\left(x_{1,t-1} \otimes x_{1,t-1}\right) + A_{12}\left(x_{1,t-1} \otimes \varepsilon_t\right) + A_2 \varepsilon_t + A_{22}(\varepsilon_t \otimes \varepsilon_t),$$

$$x_t = x_0 + x_{1,t} + x_{2,t},$$

  - where $\varepsilon_t \sim \mathrm{NIID}(0, \Sigma)$.
- The initial justification for pruning was a little ad hoc, and met with some hostility, however this process has subsequently been put on much firmer theoretical ground by Lombardo (2010) and Lan and Meyer-Gohde (2013), who also extend the procedure to higher orders.
  - The Lan and Meyer-Gohde (2013) procedure seems to be the most accurate, but differences only become noticeable at third order or higher.
  - Dynare does not at present use this one, but DynareOBC does, even for models without OBCs.

The Kollmann trick for non-linear DSGE models (2/3)

- Now consider the augmented state vector, $[x'_{1,t} \quad x'_{2,t} \quad \tilde{x}'_{1,t} \otimes \tilde{x}'_{1,t}]'$, where $\tilde{}$ denote the subset of l state variables, which (by standard properties of Kronecker products) evolves according to:

$$\begin{bmatrix} x_{1,t} \\ x_{2,t} \\ \tilde{x}_{1,t} \otimes \tilde{x}_{1,t} \end{bmatrix} = o + P \begin{bmatrix} x_{1,t-1} \\ x_{2,t-1} \\ \tilde{x}_{1,t-1} \otimes \tilde{x}_{1,t-1} \end{bmatrix} + Q\xi_t,$$

- where $\xi_t := \begin{bmatrix} \varepsilon_t \\ \varepsilon_t \otimes \varepsilon_t - \operatorname{vec}\Sigma \\ \tilde{x}^{(1)}_{t-1} \otimes \varepsilon_t \end{bmatrix}$, which satisfies $\mathbb{E}\xi_t = 0$, $\mathbb{E}\xi_t\xi'_t = \Omega$,
- for some constant vector o and constant matrices P, Q and Ω.
  - See Holden (2016b) for the formulas, and for the third order generalisation.
- The Kollmann (2013) trick for estimating these pruned perturbation approximations is to approximate the distribution of ξ_t by $\mathrm{N}(0, \Omega)$.

The Kollmann trick for non-linear DSGE models (3/3)

- Assuming the measurement equations are linear in the augmented state, the result is a system which is linear in the augmented state, with approximately Gaussian shocks.

- Thus, the standard Kalman filter may be applied (with some loss of econometric efficiency due to the non-Gaussian shocks).
  - Further intuition for the efficiency loss is provided by the fact that we are ignoring the relationship between $x_{1,t}$ and $x_{1,t} \otimes x_{1,t}$ in the augmented state.

- The econometric efficiency cost is arguably worth paying to have a computationally efficient algorithm.

Using the Kollmann method in the Holden Dynare version

- This algorithm is implemented for second order estimation in the Holden version of Dynare Unstable.
  - It is not implemented in standard Dynare Unstable.

- The implementation works by fooling Dynare into thinking it is actually estimating a linear model.
  - As a result, all standard Dynare estimation features may be used.

- To use it, one just has to add the following two instructions to the MOD file:
  - `options_.gaussian_approximation = 1;`
  - `options_.underlying_order = 2;`
- There is an example of using this algorithm in the file:

  examples\nonlinear-estimation\NKGaussianApproximationEstimate.mod

The Meyer-Gohde (2014) nonlinear estimation approach (1/2)

- Meyer-Gohde (2014) proposes an even simpler way of estimating non-linear DSGE models.

- Due to the extended state space representation of pruned perturbation solutions, it is easy to derive a closed for expression for either the mean or the risky steady state (r.s.s.) of the approximation.

- Since a third order pruned approximation usually provides a reasonable approximation to the true model, it will also provide a good approximation to the true mean or r.s.s..

- Having found this point, we can then easily take a first order approximation around it (of the original third order approximation).

The Meyer-Gohde (2014) nonlinear estimation approach (2/2)

- Since the mean or r.s.s. of the third order approximation contains risk adjustment terms, so too will the mean and r.s.s. of the first order approximation around the selected point.

- Since around the mean or r.s.s. of the third order approximation, shocks to volatility have non-trivial effects, these shocks will also have non-trivial effects in the first order approximation around the mean or r.s.s..
  - Recall that first order gives no effect of volatility; second order gives a constant effect of steady-state volatility; and third order gives a dynamic effect of volatility.

- Since a first order approximation is linear, the standard Kalman filter may then be used for estimation.

Using the Meyer-Gohde method in the Holden Dynare version

- This algorithm is implemented for second and third order estimation in the Holden version of Dynare Unstable.
  - It is not implemented in standard Dynare Unstable.

- Again, the implementation works by fooling Dynare into thinking it is actually estimating a linear model.
  - As a result, all standard Dynare estimation features may be used.

- To use it, one just has to add the following instructions to the MOD file:
  - Either `options_.non_central_approximation = 1;` for approximation around the r.s.s., or `options_.non_central_approximation = 2;` for approximating around the mean.
  - `options_.underlying_order = 3;`
- There is an example of using this algorithm in the file:

examples\nonlinear-estimation\NKNonCentralEstimate.mod

Closing remarks

- Really hope you enjoyed the course.
- Any feedback on it would be much appreciated.
- I also hope these tools are of some use to you in future.
- I'm more than happy to provide e-mail support on them.
- Do consider signing up for the Surrey Advanced DSGE Summer Course for further topics.