ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Meyer-Gohde, Alexander

Article — Published Version

Solving Linear DSGE Models with Bernoulli Iterations

Computational Economics

Provided in Cooperation with:
Springer Nature

Suggested Citation: Meyer-Gohde, Alexander (2024) : Solving Linear DSGE Models with Bernoulli Iterations, Computational Economics, ISSN 1572-9974, Springer US, New York, NY, Vol. 66, Iss. 1, pp. 593-643,
https://doi.org/10.1007/s10614-024-10708-z

This Version is available at:
https://hdl.handle.net/10419/330775

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.


https://creativecommons.org/licenses/by/4.0/

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

Computational Economics (2025) 66:593–643
https://doi.org/10.1007/s10614-024-10708-z

Solving Linear DSGE Models with Bernoulli Iterations

Alexander Meyer-Gohde[1]

Accepted: 29 August 2024 / Published online: 26 September 2024
© The Author(s) 2024

Abstract
This paper presents and compares Bernoulli iterative approaches for solving linear DSGE models. The methods are compared using 99 different models from the macroeconomic model data base (MMB) and different parameterizations of the monetary policy rule in the medium-scale new Keynesian model of Smets and Wouters (Am Econ Rev 97(3):586–606, 2007. https://doi.org/10.1257/aer.97.3.586) iteratively. I find that Bernoulli methods providing similar accuracy as measured by the forward error of the solution at a somewhat higher computation burden to the standard method of Dynare when solving DSGE models. The method, however, has convergence properties useful when a specific solution, e.g., unique stable, is sought and can be combined with other iterative methods, such as the Newton method, lending themselves especially to refining solutions—either when standard methods fail or when one moves through a parameter space iteratively—as I show in applications of the methods.

Keywords Functional iteration · Numerical accuracy · DSGE · Solution methods

JEL Classification C61 · C63 · E17

1 Introduction

Solving linear DSGE models can be reduced to solving a system of linear equations and a matrix quadratic equation with the latter being the more challenging computationally. Standard existing methods predominantly rely on a generalized Schur or QZ decomposition (Moler & Stewart, 1973; Golub & van Loan, 2013) to solve this matrix quadratic equation. Alternative methods from the applied mathematics literature to solve this matrix quadratic equation have yet to be systematically studied in a DSGE context. This paper fills part of that gap, using

✉ Alexander Meyer-Gohde
meyer-gohde@econ.uni-frankfurt.de

[1] Goethe-Universität Frankfurt and Institute for Monetary and Financial Stability (IMFS), Theodor-W.-Adorno-Platz 3, 60629 Frankfurt am Main, Germany

Bernoulli-based solution methods from the mathematics literature, introducing line-search and hybrid Newton based methods for matrix quadratic problems and applying them to the solution of linear DSGE models. The ability of Bernoulli methods to be formulated such that they converge to the solution with the smallest eigenvalues in magnitude (ideally the stable solution sought by the researcher) allow them to reliably compute the stable solution, generally with accuracy on the same order of magnitude but higher computational costs than QZ-based methods. When combined with Newton methods and their asymptotic quadratic rate of convergence, the iterative nature of the algorithms also enables the Bernoulli based methods I introduce to correct inaccurate solutions and to generate solutions more quickly than QZ-based methods when iteratively exploring a parameter space.

Bernoulli's method is originally a method for computing the largest root of a scalar polynomial[1] and Bernoulli-based functional iterations, familiar to economists in such root-finding settings—see, e.g., Judd (1998), are an alternative to QZ-based methods. The Bernoulli algorithm for matrix polynomials, see Dennis et al. (1978) and explicitly formulated for minimal solvent to a matrix quadratic in Higham and Kim (2000), was introduced by Binder and Pesaran (1999) to linear DSGE models and Rendahl (2017) extended the theoretical convergence analysis of Higham and Kim (2000) to potentially singular (or non monic matrix polynomials) models that are pervasive in the DSGE literature. Yet this algorithm and recent extensions from the applied mathematics literature such as Bai and Gao (2007) have not yet been systematically studied for their numerical properties. This paper does precisely this and introduces a number of algorithms for solving matrix quadratic problems that incorporate insights from other studies and algorithms, including exact line searches from Higham and Kim (2001) and the solution of DSGE models using Newton methods from Meyer-Gohde and Saecker (2024). To build intuition I begin Sect. 3 with the scalar case of the algorithm to build intuition before introducing the multidimensional algorithms. I show these extensions improve global convergence by making the Bernoulli method faster and more reliable, improving on the slow convergence (i.e., many iterations) of the baseline Bernoulli algorithm. Finally, I assess the accuracy of the different methods using the practical forward error bounds of Meyer-Gohde (2023), derived from a numerical stability approach incorporating conditioning number and backward error bounds.

In this paper, I present eleven different Bernoulli-based solution algorithms using a unified notation and for the application to solving linear DSGE models as an alternative to QZ-based methods. I engage in a number of experiments to compare the algorithms to QZ-based methods[2] and Meyer-Gohde and Saecker's (2024) Newton algorithms, following exactly their experiments to ensure comparability. First I apply the different methods to the models in the

[1] It can be cast as the power method, see Golub and van Loan (2013, pp. 366–367) operating on the polynomial's companion matrix.

[2] I use Dynare's (Adjemian et al., 2011) implementation of the QZ method, documented in Villemot (2011), for comparison. Varying implementations of the QZ or generalized Schur decomposition to solve linear DSGE models can be found in Uhlig (1999), Sims (2001), Klein (2000), Al-Sadoon (2018).

Macroeconomic Model Data Base (MMB) (see Wieland et al., 2012, 2016), comparing the performance to the QZ-based method of Dynare and the Newton method both unconditionally (i.e., replacing the QZ method) and then as a refinement (i.e., initializing the iterative methods with the solution generated from QZ). I find that the baseline Bernoulli method always converges to the unique stable solution, but provides a solution of the same order of magnitude of accuracy albeit at an order of magnitude higher computational cost than Newton methods. The different extended methods generally perform comparably—the exception being Bai and Gao's (2007) modified Bernoulli iteration that is several orders of magnitude more slow and frequently faces convergence issues—trading the convergence of the baseline algorithm off against more accuracy and/or less computational costs.

The algorithms are iterative in nature, enabling them to refine the solutions provided by the another method. Using the QZ solution from Dynare to initialize, the methods improve the accuracy of the solution at an additional computational cost generally between 0.1 and 2 times the original cost of Dynare's QZ, with convergence of all of the algorithms to the unique stable solvent for all of the models in the MMB, except Bai and Gao's (2007) modified Bernoulli and columnwise Newton–Bernoulli combinations. This iterative nature also lends itself to iterative parameter experiments or estimations and I compare the algorithms with Dynare's QZ method and Newton algorithm in solving for different parameterizations of the monetary policy rule in the celebrated Smets and Wouters (2007) model of the US economy. Filling in a grid with different values of the reaction of the nominal interest rate rule to inflation and real activity, whereas Dynare's QZ method starts anew at each parameterization, iterative methods can use the solution from the previous, nearby parameterization to initialize the algorithm. As the density of the grid increases, all of the methods eventually surpass QZ by roughly an order of magnitude in terms of computation cost. Finally, I show that when Dynare's QZ provides a numerically unstable solution with high forward errors that leads the predicted moments of the model's variables to be wrong in all digits, the Bernoulli method initialized at this solution provides a refined solution with improved predicted moments and forward errors. However, the Bernoulli algorithms are far from being uniform improvements over the existing standard in the literature and, in particular, outside of situations where an informative initialization is available, the standard QZ is to be preferred.

The paper is organized as follows: Sect. 2 lays out the general DSGE model and the unknown solution. In Sect. 3, I begin by presenting the Bernoulli method for a scalar quadratic equation to build intuition before I then present the set of different Bernoulli methods from the applied mathematics literature in a unified notation as they apply to the class of DSGE models. Section 4 examines practical and theoretical considerations such as the choice of initial value, accuracy, and convergence. In Sect. 5, I compare the different Bernoulli methods to the standard QZ method and Newton method of Meyer-Gohde and Saecker (2024) in two applications, one using the MMB of 99 different models and the second over a range of parameterizations within the Smets and Wouters (2007) model. Finally, Sect. 6 concludes.

2 Problem Statement

Standard numerical DSGE solution packages available to economists and policy makers—e.g., Dynare (Adjemian et al., 2011), Gensys (Sims, 2001), (Perturbation) AIM (Anderson & Moore, 1985; Anderson et al., 2006), Uhlig's Toolkit (Uhlig, 1999) and Solab (Klein, 2000)— analyze models that, when (log)linearized, can be expressed in the form of the system of n_y linear expectational difference equations

$$0 = AE_t\left[y_{t+1}\right] + By_t + Cy_{t-1} + D\varepsilon_t \tag{1}$$

where A, B, and C are $n_y \times n_y$ real valued matrices, $\mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$, that operate on $y_t \in \mathbb{R}^{n_y}$ the vector of n_y endogenous variables; D is an $n_y \times n_e$ real valued matrix, $\mathbb{R}^{n_e} \to \mathbb{R}^{n_y}$, that operate on $\varepsilon_t \in \mathbb{R}^{n_e}$ the vector of n_e exogenous, serially uncorrelated shocks with a known distribution, assumed mean zero[3]; where n_y and n_e are positive integers ($n_y, n_e \in \mathbb{N}$).

The solution to (1) is sought as the unknown linear solution in the form

$$y_t = P\, y_{t-1} + Q\, \varepsilon_t, \quad \mathbb{R}^{n_y+n_e} \to \mathbb{R}^{n_y} \tag{2}$$

a recursive solution in the time domain–solutions that posit y_t as a function of its own past, y_{t-1}, and exogenous innovations, ε_t, which reduces the problem to finding two real valued matrices, P and Q, $n_y \times n_y$ and $n_y \times n_e$ respectively.[4]

Inserting (2) into (1) and taking expectations ($E_t\left[\varepsilon_{t+1}\right] = 0$), yields the restrictions

$$0 = AP^2 + BP + C, \quad 0 = (AP + B)Q + D \tag{3}$$

Generally, a unique P with eigenvalues inside the closed unit circle is sought. Lan and Meyer-Gohde (2014) prove the latter can be uniquely solved for Q if such a P can be found. Hence, the hurdle is the former, matrix quadratic equation.

Most linear DSGE methods use a generalized Schur or QZ decomposition (Moler & Stewart, 1973; Golub & van Loan, 2013) of the companion linearization of (1) in some form or another. I will take a different route and instead now solve for P in (3) using Bernoulli iterations.

[3] This assumption follows Dynare for expediency at to facilitate the comparison and implementation in the applications of Sect. 5. Any pattern of serial correlation than can be captured by a finite order ARMA process can brought into the above form by expanding y_t appropriately. Meyer-Gohde and Neuhoff (2015) show that the matrix quadratic and its solution P is unchanged from the presentation here by this assumption—merely the mapping from exogenous shocks to endogenous variables becomes more involved.

[4] As above this is a brief statement of the linear problem, Binder and Pesaran (1997), Uhlig (1999), Klein (2000), Sims (2001), and Anderson (2010) provide overviews of the multivariate approaches that have become standard and, more recently, Al-Sadoon (2018) and Al-Sadoon (2020) provide a much more rigorous approach that go beyond the problem of solving for two matrices that I address here.

3 Bernoulli Iterations for Linear DSGE Models

I will begin by analyzing a univariate equation, see, e.g., Higham (2002, p. 508) for the functional iteration to find the unstable solution and Judd (1992, pp. 152–153) for functional iterations in general fix point problems for economists, to fix ideas and illustrate how Bernoulli functional iterations can be used to solve quadratic equations.

The problem generated by (3) is a (matrix) quadratic problem. To fix ideas, consider its univariate equivalent

$$0 = ax^2 + bx + c \tag{4}$$

where I consider (in accordance with the DSGE model), a, b, and $c \in \mathbb{R}^1$. A functional iteration will reformulate this as

$$x = f(x) \tag{5}$$

giving an iterative procedure to generate a solution

$$x_{j+1} = f\left(x_j\right), \quad \text{with some } x_0 \tag{6}$$

From the quadratic equation above, there are a number of possibilities,

$$f(x) = -\frac{c}{ax + b} \tag{7}$$

$$f(x) = -\frac{b + \frac{c}{x}}{a} \tag{8}$$

$$f(x) = -\frac{ax^2 + c}{b} \tag{9}$$

and so forth. I will focus on the first $f(x) = -\frac{c}{ax+b}$ as it is the univariate counterpart of the multivariate Bernoulli algorithm that will be introduced subsequently. Hence

$$x_{k+1} = f\left(x_k\right) = -\frac{c}{ax_k + b}, \quad k = 0, 1, 2..., x_0 \text{ given} \tag{10}$$

Characterize the two solutions via

$$\left(s_1 x - t_1\right)\left(s_2 x - t_2\right) = \underbrace{s_1 s_2}_{a} x^2 \underbrace{-\left(s_1 t_2 + s_2 t_1\right)}_{+b} x + \underbrace{t_1 t_2}_{c} \tag{11}$$

where the solutions are—in analogy to the generalized eigenvalue formulation with the generalized Schur of Klein (2000)

$$x^{(i)} = \left\{ \frac{t_i}{s_i}, \text{ if } s_i \neq 0; \emptyset, \text{ if } s_i = 0; \mathbb{R}, \text{ if } s_i = t_i = 0; i = 1, 2 \right\} \tag{12}$$

Following, e.g., Judd (1998, pp. 165–166), local convergence requires $|f'(x)| < 1$ for an x that solves $x = f(x)$ and as

$$f'(x) = \frac{ac}{(ax+b)^2} = \frac{s_1 s_2 t_1 t_2}{\left(s_1 s_2 x_k - \left(s_1 t_2 + s_2 t_1\right)\right)^2} \tag{13}$$

for, say, $x^{(1)} = \frac{t_1}{s_1}$

$$f'\left(x^{(1)}\right) = \frac{s_1 s_2 t_1 t_2}{\left(s_1 s_2 \frac{t_1}{s_1} - \left(s_1 t_2 + s_2 t_1\right)\right)^2} = \frac{s_1 s_2 t_1 t_2}{\left(s_1 t_2\right)^2} = \frac{s_2 t_1}{s_1 t_2} = \frac{x^{(1)}}{x^{(2)}} \tag{14}$$

and hence the iteration is convergent if

$$\left|f'\left(x^{(1)}\right)\right| = \left|\frac{x^{(1)}}{x^{(2)}}\right| < 1 \tag{15}$$

or there is local convergence to the minimal (smaller in modulus) root of the quadratic when the iteration is formulated via $f(x) = -\frac{c}{ax+b}$.

This highlights a significant advantage over Newton-based methods, namely that the functional iteration can be tailored to deliver convergence of the algorithm to a particular root: for example, $f(x) = -\frac{c}{ax+b}$ to the minimal (as above) and $f(x) = -\frac{b+\frac{c}{x}}{a}$ to the dominant (Higham & Kim, 2000) solution. As a particular solution or a solution with particular properties (namely the minimal solution in a saddle point stable problem) is sought in DSGE models, the Bernoulli iteration $x_{k+1} = f\left(x_k\right) = -\frac{c}{ax_k+b}$ is potentially very useful in solving DSGE models.

Turning now to the matrix problem, I will formalize the matrix quadratic equation in (3). For A, B, and $C \in \mathbb{R}^{n_y \times n_y}$, a matrix quadratic $M(P) : \mathbb{C}^{n_y \times n_y} \to \mathbb{C}^{n_y \times n_y}$ is defined as

$$M(P) \equiv A\,P^2 + B\,P + C \tag{16}$$

with its solutions, called solvents, given by $P \in \mathbb{C}^{n_y \times n_y}$ if and only if $M(P) = 0$. The eigenvalues of the solvent, called latent roots of the associated lambda matrix,[5] $M(\lambda) : \mathbb{C} \to \mathbb{C}^{n \times n}$ (here of degree two), are given via

$$M(\lambda) \equiv A\,\lambda^2 + B\,\lambda + C \tag{17}$$

The latent roots are (i) values of $\lambda \in \mathbb{C}$ such that $\det M(\lambda) = 0$ and (ii) $n_y - \text{rank}(A)$ infinite roots. An explicit link between the quadratic matrix problem and the quadratic eigenvalue problem is given via

$$\lambda \in \mathbb{C} : \left(A\lambda^2 + B\lambda + C\right)x = 0 \text{ for some } x \neq 0 \tag{18}$$

[5] See, e.g., Dennis et al. (1976, p. 835) or Gantmacher (1959, vol. I, p. 228).

which has been reviewed extensively by Tisseur and Meerbergen (2001) and for which Hammarling et al. (2013) provide a comprehensive method to improve the accuracy of its solutions.

The matrix quadratic (16) can be expanded following Higham and Kim (2001) as

$$M(P+\Delta P) = A\,(P+\Delta P)^2 + B\,(P+\Delta P) + C \tag{19}$$

$$= A\,P^2 + B\,P + C + A\,\Delta P^2 + A\,(P\Delta P + \Delta P P) + B\,\Delta P \tag{20}$$

$$= M(P) + (A\,\Delta P P + (A\,P + B\,)\Delta P) + A\,\Delta P^2 \tag{21}$$

$$= M(P) + \mathscr{D}_P(\Delta P) + A\,\Delta P^2 \tag{22}$$

where $\mathscr{D}_P M(\Delta P)$ is the Fréchet derivative of M at P in the direction ΔP.

3.1 Baseline Bernoulli Iteration

The multivariate counterpart to the algorithm $x_{k+1} = f\left(x_k\right) = -\frac{c}{ax_k+b}$ above is the following baseline Bernoulli method for the minimal solvent. Beginning with (16)

$$0 = (A\,P + B\,)P + C \tag{23}$$

and defining the iteration via

$$0 = \left(A\,P_j + B\,\right)P_{j+1} + C \tag{24}$$

gives the baseline Bernoulli method

$$P_{j+1} = -\left(A\,P_j + B\,\right)^{-1} C \tag{25}$$

This iteration was noted by Binder and Pesaran (1999) and is implemented in their `RBCQDE` Matlab and Gauss code,[6] but without discussion of its convergence or computational properties. Rendahl (2017) studies the Bernoulli recursion in (25) following Higham and Kim (2000) by analyzing the theoretical properties and exploring a few examples. The baseline Bernoulli recursion is summarized in Algorithm 1.

[6] See https://dge.repec.org/codes/binder/handbook/.

Algorithm 1: Baseline Bernoulli method

Given: A, B, C, an initial P_0, and a convergence criterion ϵ

While *criterion*$(P_j) > \epsilon$ **do**

Set $P_{j+1} = -\left(A P_j + B\right)^{-1} C$

Advance $j = j + 1$

end

Return: P_j

Higham and Kim (2000) show that the above recursion converges asymptotically at a linear rate to the minimal solvent, but only under the assumption that this solvent is invertible (e.g., ruling out zero latent roots in (17)) and that a dominant solvent also exists (e.g., ruling out "infinite" latent roots in (17)), both of which abound in the DSGE literature. I return to this issue in Sect. 4 and provide a proof that this recursion converges at least locally to the minimal solvent when this solvent is the unique stable solution. Both at such a solvent and for the first iteration if $P_0 = 0$ and B is of full rank, see Binder and Pesaran (1997) and Uhlig (1999), the coefficient matrix $A P_j + B$ will be invertible.[7]

3.2 Bai and Gao's (2007) Modified Bernoulli Method

The modified Bernoulli method of Bai and Gao (2007) is a straightforward extension of the Bernoulli method that updates the solution matrix column by column alongside the standard iteration above, incorporating the update of the previous column when solving for the current column. Bai and Gao (2007) note that the Bernoulli recursion

$$P_{j+1} = -\left(A P_j + B\right)^{-1} C \tag{26}$$

solves n linear equations at each iteration with $P_{i,j+1}$ and C_i the i'th columns of P_{j+1} and C respectively

[7] In practice, see the applications later, singularity is not a general issue, but in the few iterations where it did occur, I set P_{j+1} to the minimum 2-norm solution of $\left(A P_j + B\right) P_{j+1} = -C$, see Higham and Kim (2000, p. 512) determination that the algorithm can be rerun or reinitialized if numerical difficulties should be encountered.

$$\left(A\,P_j + B\right)P_{j+1} = -C \Leftrightarrow \left(A\,P_j + B\right)P_{i,j+1} = -C_i \text{ for } i = 1, ..., n \tag{27}$$

and that when solving these individual equations for $P_{i,j+1}$ in sequence for $i = 1, ..., n$, one can use the "real-time" value of P_j, updated to those columns that precede the current column

$$\left(A\,\tilde{P}_{i,j} + B\right)P_{i,j+1} = -C_i \text{ for } i = 1, ..., n \tag{28}$$

where

$$\tilde{P}_{i,j} = \left[P_{1,j+1} \;\cdots\; P_{i,j+1}\; P_{i+1,j} \;\cdots\; P_{n,j}\right] \tag{29}$$

That is, this method follows the Gauss–Seidel method by promptly updating while moving through the columns.

Instead of solving n systems at each iteration

$$P_{i,j+1} = -\left(A\,\tilde{P}_{i-1,j} + B\right)^{-1} C_i \text{ for } i = 1, ..., n \tag{30}$$

they use the Sherman–Morrison, see, e.g., Golub and van Loan (2013, p. 51) or Horn and Johnson (2012, p. 19), formula for updating matrices by noticing that $\tilde{P}_{i+1,j}$ is a rank one correction of $\tilde{P}_{i,j}$. Moving through the columns successively gives

$$P_{i+1,j+1} = -\left(A\,P_j + B\right)^{-1} C_{i+1} + \sum_{l=1}^{i} \frac{e_l' q_{l,j}}{1 + e_l' p_{l,j}} p_{l,j} \tag{31}$$

where

$$u_{i,j} = A\left(P_{i,j+1} - P_{i,j}\right) \tag{32}$$

$$q_{l,j} = \left(A\,P_j + B\right)^{-1} u_{i,j} \tag{33}$$

$$p_{l,j} = \left(A\,P_j + B\right)^{-1} C_{i+1} \tag{34}$$

and e_i' is a vector of zeros with a one in the i'th column. This gives the modified Bernoulli Algorithm 2.

Algorithm 2: Modified Bernoulli method

Given: A, B, C, an initial P_0, and a convergence criterion ϵ

While *criterion*$(P_j) > \epsilon$ **do**

  For $i = 1 : n$ **do**

    Solve for $P_{i,j+1}$ via $P_{i,j+1} = -\left(AP_j + B\right)^{-1} C_i$, where

    $P_{j+1} \equiv \begin{bmatrix} P_{1,j+1} & P_{2,j+1} & \dots & P_{n,j+1} \end{bmatrix}$

  end

  For $i = 1 : n-1$ **do**

    Update $P_{i,j+1}$ recursively via

    $u_{i,j} = A\left(P_{i,j+1} - P_{i,j}\right); \quad p_{l,j} = \left(AP_j + B\right)^{-1} C_{i+1}; \quad q_{l,j} = P_{i+1,j+1}$

    For $l = i+1 : n$ **do**

      $P_{l,j+1} = P_{l,j+1} + \frac{e_i' q_{i,j}}{1 + e_i' p_{i,j}} p_{i,j}$

    end

  end

  Advance $j = j + 1$

end

Return: $P_j \equiv \begin{bmatrix} P_{1,j} & P_{2,j} & \dots & P_{n,j} \end{bmatrix}$

The algorithm is written in two passes. The first pass updates all the columns once using the standard Bernoulli approach above—if we stop here, this is simply the baseline Bernoulli algorithm above. The second pass moves through the columns updating all columns that follow with update based on the current column. This two pass, triangular approach accomplishes (30) without solving n different systems at each iteration, as the first pass has the same coefficient matrix for all columns that can be implemented efficiently and the second pass contains rank one corrections.

Bai and Gao (2007) provide proof that this recursion converges to the minimal solvent, again under the assumption that a minimal and dominant solvent both exist. They explore variants that operate on blocks, several columns at once, alongside the column-by-column version I implement. While they argue that this recursion outperforms the baseline Bernoulli method, their own examples show that this outperformance is minimal with Bai and Gao's (2007, p. 509) tables 6.1 and 6.2 shwoing relative errors of the same magnitude and a savings of between 0 and 10% in the number of iterations of their method over the baseline Bernoulli method.

3.3 Bernoulli Line Search

Newton based methods, see Meyer-Gohde and Saecker (2024), allow for the straightforward incorporation of line search routines to find optimal increments, see Higham and Kim (2001). A line search to scale the increments can protect against perniciously large steps and accelerate convergence when the increments are timidly small. Bernoulli iterations generally require many more iterations than Newton based methods and their rate of convergence is governed by the ratio of the largest in modulus stable and smallest in modulus unstable latent roots, e.g. Higham and Kim (2000)—hence the primary difficulty will be slow convergence associated with timidly small increments.

The Bernoulli methods above, however, are functional iterations that work directly on the solution P. But simple reformulation reveals the implied increment in a Bernoulli step

$$P_{j+1} = -(AP_j + B)^{-1}C \tag{35}$$

$$\equiv \Delta P_j + P_i \tag{36}$$

$$= \underbrace{-(AP_j + B)^{-1} M(P_j)}_{K(P_j)} + P_j \tag{37}$$

In a Newton context, Higham and Kim (2001) motivate an exact line search by the inaccuracies of the linear approximation that delivers the Newton step: if P_j is far from a solvent ($P : M(P) = 0$), the update $P_{j+1} = P_j + \Delta P_j$ might be farther from a solvent than P_j. The same logic holds with the Bernoulli iteration, albeit for a different reason: with the linear convergence of the algorithm, the update $P_{j+1} = P_j + \Delta P_j$ is likely to be too conservative an update. Indeed, using the definition $K(P_j) = -(AP_j + B)^{-1}$ and comparing with (19), we see that $K(P_j) \approx \mathscr{D}_P(\Delta P)$, differing from the Fréchet derivative of M at P in the direction ΔP only by lacking the term $A\,\Delta P P$, giving the Bernoulli increment as a simplification of the Newton increment (see Meyer-Gohde & Saecker 2024). That is, in the Bernoulli approach the increment ΔP_j solves a standard linear system $(AP_j + B)\Delta P_j = -M(P_j)$, whereas the "sandwiched" term $A\,\Delta P P$ in the Newton derivatine requires the solution of a Sylvester equation $A\,\Delta P P + (AP_j + B)\Delta P_j = -M(P_j)$ to recover the increment.

I propose a line search, a multiple of the Bernoulli step above, $P_{j+1} = t\Delta P_j + P_j$ where t is an appropriate scalar. Obviously, if $t = 1$, the baseline Bernoulli iteration is recovered. Following Higham and Kim (2001), I select the multiple of the Bernoulli step by finding a t that minimizes the merit function

$$t = \underset{x \geq 1}{\operatorname{argmin}} \; \|M(P + x\Delta P)\|_F^2 \tag{38}$$

The term $M(P + x\Delta P)$ can be expressed explicitly as

$$M(P + x\Delta P) = A(P + x\Delta P)^2 + B(P + x\Delta P) + C \tag{39}$$

$$= AP^2 + BP + C + Ax^2\Delta P2 + A(Px\Delta P + x\Delta P \cdot P) + Bx\Delta P \tag{40}$$

$$= M(P) + x^2A\Delta P^2 + \underbrace{x\left(A(P\Delta P + \Delta P \cdot P) + B\Delta P\right)}_{\mathscr{D}_P(\Delta P)} \tag{41}$$

where $\mathscr{D}_P(\Delta P)$ is the Fréchet derivative of M at P in the direction ΔP. Higham and Kim (2001) show that this particular choice of merit function is convenient for Newton based algorithms as the Newton step sets the Fréchet derivative to the negative of the matrix quadratic ($\mathscr{D}_P(\Delta P) + M(P) = 0$) by construction allowing $M(P + x\Delta P)$ to be written as $M(P + x\Delta P) = (1 - x)M(P) + x^2A\Delta P^2$. With the Bernoulli iteration, we are not quite as fortunate, as

$$\mathscr{D}_P(\Delta P) = (AP + B)\Delta P + A\Delta P \cdot P \tag{42}$$

$$= -K(P)^{-1}\Delta P + A\Delta P \cdot P \tag{43}$$

$$= -M(P) + A\Delta P \cdot P \tag{44}$$

and hence

$$M(P + x\Delta P) = M(P) + x^2A\Delta P^2 - xM(P) + xA\Delta P \cdot P \tag{45}$$

$$= (1 - x)M(P) + xA\Delta P(x\Delta P + P) \tag{46}$$

The merit function (38), $g(x) \equiv \|M(P + x\Delta P)\|_F^2$ is a quartic polynomial with at most two minima and its first derivative is, see the "Appendix" for details,

$$g'(x) = 4\gamma x^3 + 3(\xi - \sigma)x^2 + 2(\alpha + \beta - \delta + \sigma)x + \delta - 2\alpha \tag{47}$$

where

$$\alpha = \|M(P)\|_F^2, \qquad \beta = \|A\Delta P \cdot P\|_F^2, \qquad \gamma = \left\|A(\Delta P)^2\right\|_F^2 \tag{48}$$

$$\xi = \operatorname{tr}\left((A\Delta P \cdot P)^*A(\Delta P)^2 + \left(A(\Delta P)^2\right)^*A\Delta P \cdot P\right) \tag{49}$$

$$\sigma = \operatorname{tr}\left(M(P)^*A(\Delta P)^2 + \left(A(\Delta P)^2\right)^*M(P)\right) \tag{50}$$

$$\delta = \operatorname{tr}\left(M(P)^*A\Delta P \cdot P + (A\Delta P \cdot P)^*M(P)\right) \tag{51}$$

As the goal is to minimize $g(x)$ over the interval $[1, \infty)$, I need to find minima and compare their values to $g(1)$. Finding extrema corresponds to finding zeros of the

cubic polynomial $g'(x)$ in this range. Implementing t from (38) is straightforward as either there is a single real zero of $g'(x)$ which lies in the range and is a minimum of $g(x)$ (as $\alpha > 0$) or $g'(x)$ has three real zeros, of which at most two correspond to minima of $g(x)$. Hence, finding the zeros of $g'(x)$ and comparing the associated values of $g(x)$ with the value of $g(1)$ enables t from (38) to be readily found.

This gives the Bernoulli procedure with exact line searches as Algorithm 3.

Algorithm 3: Bernoulli with exact line searches

Given: A, B, C, an initial P_0, and a convergence criterion ϵ

While *criterion*$(P_j) > \epsilon$ **do**

Solve for $\Delta P_j = -(AP_j + B)^{-1} M(P_j)$

Solve for t_j in $t_j = \operatorname{argmin}_{x \geq 1} \left\| M(P_j + x\Delta P_j) \right\|_F^2$

Set $P_{j+1} = P_j + t_j \Delta P_j$

Advance $j = j + 1$

end

Return: P_j

Line searches have been shown by Higham and Kim (2001) to not interfere with the asymptotic convergence rate, hence the expectation of this algorithm is a factor but not order increase in the convergence speed.

3.4 Combining Bernoulli and Newton

From the previous algorithm, the Bernoulli iteration $P_j \to P_{j+1}$ can be described via an increment, $\Delta_B P_j$,

$$P_{B,j+1} = \Delta_B P_j + P_j \tag{52}$$

The same holds for the Newton methods studied in Meyer-Gohde and Saecker (2024) with the increment $\Delta_N P_j$,

$$P_{N,j+1} = \Delta_N P_j + P_j \tag{53}$$

where $\Delta_N P_j$ is the increment that sets the first-order in ΔP approximation of the Fréchet derivative of M at P_j in the direction ΔP, see (22), to zero.

This observation motivates the following iteration

$$P_{j+1} = s_j \Delta_B P_j + \left(1 - s_j\right) \Delta_N P_j + P_j, \quad 0 \leq s_j \leq 1 \tag{54}$$

with an average of the Bernoulli and Newton increments being weighted by s_j. A simple choice of weights stems from the observations that motivated line searches both for Newton and Bernoulli iterations. For Newton based methods, there is a potential of taking too large steps and in the wrong direction (i.e., the algorithm is

known to overshoot and is not guaranteed to converge towards the minimal solvent) and for Bernoulli too small steps. Hence, one would like to take the potentially larger Newton step when it is going in the same general direction that the Bernoulli step would take and to take the Bernoulli step when the Newton step would go in a different direction (presumably to another solvent).

This has an interpretation in the vein of momentum modifications of gradient descent methods like Newton that go back at least to Polyak (1964). Whereas a momentum approach would attenuate movements of a Newton iteration via

$$P_{j+1} = P_j + \Delta_N P_j + \beta_j \Delta_N P_{j-1}, \quad 0 \leq \beta_j \leq 1 \tag{55}$$

where β_j might be fixed or chosen based on some measure of the gradients at the iteration j, the approach I propose can be written

$$P_{j+1} = P_j + \Delta_N P_j + s_j\left(\Delta_B P_j - \Delta_N P_j\right), \quad 0 \leq s_j \leq 1 \tag{56}$$

So instead of simply "slowing down" the iteration, I propose slower it towards the algorithm with better convergence properties, the Bernoulli step.

Hence, measuring the angle, θ between the two increments via (Horn & Johnson, 2012, p.15)

$$\cos\theta_j = \frac{\operatorname{vec}\left(\Delta_N P_j\right)' \operatorname{vec}\left(\Delta_B P_j\right)}{\left\|\Delta_N P_j\right\|_F \left\|\Delta_B P_j\right\|_F} \tag{57}$$

the weight between the increments is defined as

$$s_j \equiv \theta_j / \pi \tag{58}$$

So when the angle between the two increments is zero (i.e., they are moving in the same direction), the cosine of θ is one, θ is zero, and s_j is zero: $P_{j+1} = \Delta_N P_j + P_i$, the Newton step with its asymptotically quadratic rate of convergence is taken. When this angle is π (i.e., the increments are moving in opposite directions), the cosine of θ is -1, θ is π, and s_j is one: $P_{j+1} = \Delta_B P_j + P_i$, the Bernoulli step with its asymptotic convergence to the minimal solvent (see the next section for a proof) is taken.

Alternatively, the increments could be weighted columnwise, that is comparing the increments associated with the solutions for the individual variables in y_t,

$$P_{j+1} \equiv \begin{bmatrix} P_{1,j+1} & \cdots & \cdots & P_{n,j+1} \end{bmatrix} \tag{59}$$

$$= \begin{bmatrix} s_{1,j}\Delta_B P_{1,j} + \left(1 - s_{1,j}\right)\Delta_N P_{1,j} & \cdots & \cdots & s_{n,j}\Delta_B P_{n,j} + \left(1 - s_{n,j}\right)\Delta_N P_{n,j} \end{bmatrix} + P_j \tag{60}$$

where the weight between the columns of the increments ($0 \leq s_{i,j} \leq 1$) is defined as

$$s_{i,j} \equiv \theta_{i,j} / \pi \tag{61}$$

via the angle, θ_i between the i'th columns of the two increments via (Horn & Johnson, 2012, p.15)

$$\cos\theta_{i,j} = \frac{\Delta_N P'_{i,j} \Delta_B P_{i,j}}{\left\|\Delta_N P_{i,j}\right\|_2 \left\|\Delta_B P_{i,j}\right\|_2} \tag{62}$$

or the weighting above could be tilted towards one or the other increment

$$\tilde{s}_j(p) = s_j^p \tag{63}$$

with an exponent $p < 1$ shifting the weight in favor of the Bernoulli increment (and hence $\tilde{s}_j(p)$ replacing s_j in (54)), perhaps to reduce the likelihood of the algorithm converging to a solvent other than the unique stable one.

This gives a combined Bernoulli–Newton procedure as Algorithm 4.

Algorithm 4: Combined Bernoulli–Newton

Given: A, B, C, an initial P_0, a convergence criterion ϵ, and a weight tilt p

While *criterion*$(P_j) > \epsilon$ **do**

Solve for $\Delta_B P_j = -(AP_j + B)^{-1} M(P_j)$

Solve for $\Delta_N P_j$ in $A\,\Delta_N P_j P_j + \left(A\,P_j + B\right)\Delta_N P_j = -M(P_j)$

Determine $\cos\theta$ (or $\cos\theta_i$ for all columns i)

Calculate $s_j = \arccos(\theta)/\pi$ (or $s_{i,j} = \arccos(\theta_i)/\pi$ for all columns i)

Set $P_{j+1} = s_j^p \Delta_B P_j + \left(1 - s_j^p\right)\Delta_N P_j + P_j$

(or $P_{i,j+1} = s_{i,j}^p \Delta_B P_{i,j} + \left(1 - s_{i,j}^p\right)\Delta_N P_{i,j} + P_{i,j}$ for all columns i)

Advance $j = j + 1$

end

Return: P_j

While this procedure will hopefully yield the best of both worlds: convergence to the desired solvent via Bernoulli and quadratic convergence via Newton, it might also do just the opposite—combining the linear convergence of Bernoulli to the unpredictable solvent of Newton. Tilting the weight towards one or the other procedure gives the user flexibility, but it is still not clear a priori how that tilt might be chosen to deliver a procedure with the desired properties.

3.5 Bernoulli and Newton Line Search

As the weighting of the Bernoulli and Newton increments relied on the same intuition that guided their respective line search algorithms, the next logical step would be to combine both the line searches for the optimal magnitudes of the respective increments and then weight the two to produce a combined Bernoulli and Newton increment with line searches. That is, if combining Bernoulli and Newton might bring the slow convergence of Bernoulli together with the

unpredictable convergence destination of Newton, performing line searches on the steps of both of these might attenuate this danger. From above, the step size of Bernoulli can be increased past 1 to improve the speed of convergence, and the step size of Newton can be adjusted optimally to balance the danger of taking too large of steps.

This gives a combined Bernoulli–Newton procedure with line searches as Algorithm 5.

Algorithm 5: Combined Bernoulli–Newton with line searches

Given: A, B, C, an initial P_0, a convergence criterion ϵ, and a weight tilt p

While *criterion*$(P_j) > \epsilon$ **do**

Solve for $\Delta_B P_j = -(AP_j + B)^{-1} M(P_j)$

Solve for $t_{B,j}$ in $t_{B,j} = \text{argmin}_{x \geq 1} \left\| M(P_j + x\Delta_B P_j) \right\|_F^2$

Solve for $\Delta_N P_j$ in $A\,\Delta_N P_j P_j + \left(A\,P_j + B\right)\Delta_N P_j = -M(P_j)$

Solve for $t_{N,j}$ in $t_{N,j} = \text{argmin}_{x \in [0,2]} \left\| M(P_j + x\Delta_N P_j) \right\|_F^2$

Determine $\cos\theta$

Calculate $s_j = \arccos(\theta)/\pi$

Set$P_{j+1} = s_j^p t_{B,j} \Delta_B P_j + \left(1 - s_j^p\right) t_{N,j} \Delta_N P_j + P_j$

Advance $j = j + 1$

end

Return: P_j

As above, the underlying steps in the procedure would tend to limit the drawbacks of the two procedures on their own. Yet, it is not a priori certain how they will perform when combined.

3.6 Optimal Bernoulli and Newton

The final set of algorithms explicitly take the optimality approach when determining s_j, minimizing the same merit function used in the line searches above. First for the initial increments

$$s = \underset{0 \leq x \leq 1}{\text{argmin}} \left\| M(x\Delta_B P + (1-x)\Delta_N P + P) \right\|_F^2 \tag{64}$$

and then for the line-search optimized increments

$$s = \underset{0 \leq x \leq 1}{\text{argmin}} \left\| M(x t_B \Delta_B P + (1-x) t_N \Delta_N P + P) \right\|_F^2 \tag{65}$$

where

$$t_i = \underset{x \in X_i}{\operatorname{argmin}} \left\| M(x\Delta_i P + P) \right\|_F^2, \text{ for } i = B, N \tag{66}$$

where X_B restricts x to be greater or equal to one, see above, and X_N restricts x to be between zero and two (Higham & Kim, 2001).

This gives an optimally (in the sense of the merit function) combined Bernoulli–Newton procedure in algorithm 6.

Algorithm 6: Optimal Bernoulli–Newton

Given: A, B, C, an initial P_0, and a convergence criterion ϵ

While *criterion*$(P_j) > \epsilon$ **do**

Solve for $\Delta_B P_j = -(AP_j + B)^{-1} M(P_j)$

Either set $t_{B,j} = t_{N,j} = 1$, or Solve for $t_{B,j}$ in $t_{B,j} = \operatorname{argmin}_{x \geq 1} \left\| M(P_j + x\Delta_B P_j) \right\|_F^2$

Solve for $\Delta_N P_j$ in $A\,\Delta_N P_j P_j + \left(A\,P_j + B\right)\Delta_N P_j = -M(P_j)$

Solve for Solve for $t_{N,j}$ in $t_{N,j} = \operatorname{argmin}_{x \in [0,2]} \left\| M(P_j + x\Delta_N P_j) \right\|_F^2$

Solve for s_j in $s_j = \operatorname{argmin}_{0 \leq x \leq 1} \left\| M(x t_{B,j} \Delta_B P_j + (1-x) t_{N,j} \Delta_N P_j + P_j) \right\|_F^2$

Set $P_{j+1} = s_j t_{B,j} \Delta_B P_j + \left(1 - s_j\right) t_{N,j} \Delta_N P_j + P_j$

Advance $j = j + 1$

end

Return: P_j

This algorithm has the advantage of weighting the Bernoulli and Newton increments in a non-arbitrary manner. Yet this comes at a cost, here of an additional optimization problem to be solved, and is likely biased towards the Newton increment as it—intuitively via quadratic convergence—generally takes larger steps, making each increment more likely to be favored over the timid Bernoulli one. This carries again the potential of losing the advantage of Bernoulli as formulated in the baseline algorithm that guarantees convergence to a particular solvent.

4 Theoretical and Practical Considerations

4.1 Initial Value

All Bernoulli iterations need an initial value, P_0. In contrast to Newton methods, the baseline Bernoulli method has strong convergence results, see the next subsection, and hence this initial value is of lesser importance. Yet many of the algorithms presented above combine Bernoulli and Newton methods and are subject to this concern. As the goal is to obtain the minimal solvent P, I choose the initial value $P_0 = 0$. In the absence of any other guidance, this choice satisfies the requirement of having all eigenvalues inside the unit circle.

Furthermore, Higham and Kim (2000, p. 512) note that the Bernoulli algorithm can be rerun or restated with a different initialization should numerical difficulties be encountered. For the baseline Bernoulli iteration which solves

$$\left(A P_j + B\right)P_{j+1} = -C \tag{67}$$

the (near) singularity of $A P_j + B$ would pose such a difficulty. In this case, the rank deficiency of the leading coefficient matrix admits multiple solutions and I chose the norm $\left(\min \left\| \left(A P_j + B\right)P_{j+1} + C\right\|_F\right)$ minimizing $P_{j+1} = -\left(A P_j + B\right)^{+} C$, where $^{+}$ indicates the Moore-Penrose inverse.

4.2 Convergence

While Higham and Kim (2000) and Bai and Gao (2007) provide convergence results for Bernoulli iterations, their analyses assume that A is nonsingular and that both a minimal and a dominant solvent exist. This is untenable in DSGE models where singular A's abound—associated with variables that arise only at time t and $t-1$—and singular C's—associated with variables that arise only at time $t+1$ and t—prevent the application of their results to the reverse quadratic. Rendahl (2017) adapts Higham and Kim's (2000) approach to DSGE models and provides proofs of the global convergence of the Bernoulli iteration for arbitrarily small perturbations of an originally singular model that produce a nonsingular system. Complementing this, I provide a local convergence proof of the original unperturbed problem in the following.

Recall the baseline Bernoulli method,

$$P_{j+1} = -\left(A P_j + B\right)^{-1} C = F\left(P_j\right) \tag{68}$$

the Fréchet derivative of F at P_j in the direction ΔP_j is $\mathscr{D}_{P_j} F\left(\Delta P_j\right)$ and be defined implicity by differentiating $\left(A P_j + B\right)F\left(P_j\right) = -C$

$$A\Delta P_j F\left(P_j\right) + \left(A P_j + B\right)\mathscr{D}_{P_j} F\left(\Delta P_j\right) = 0 \tag{69}$$

$$\mathscr{D}_{P_j} F\left(\Delta P_j\right) = -\left(A P_j + B\right)^{-1} A\Delta P_j F\left(P_j\right) \tag{70}$$

$$\mathscr{D}_{P_j} F\left(\Delta P_j\right) = \left(A P_j + B\right)^{-1} A\Delta P_j \left(A P_j + B\right)^{-1} C \tag{71}$$

using the Kronecker / vectorized representation ($\mathrm{vec}\,(ABC) = (C' \otimes A)\,\mathrm{vec}\,(B)$)

$$\mathrm{vec}\left(\mathscr{D}_{P_j} F\left(\Delta P_j\right)\right) = \left(\left[\left(A P_j + B\right)^{-1} C\right]' \otimes \left[\left(A P_j + B\right)^{-1} A\right]\right) \mathrm{vec}\left(\Delta P_j\right) \tag{72}$$

the algorithm converges (locally) to the minimal solvent.

Theorem 1 (Convergence to the unique stable solvent P) *Assume there exists a unique solvent P of $M(P) \equiv A P^2 + B P + C$ in (16) such that the eigenvalues of P comprise all the latent roots, λ of $M(\lambda)$—see 17—stable with respect to the closed unit circle, i.e., less than or equal to one in absolute value. Then the Bernoulli iteration $P_{j+1} = -\left(A P_j + B\right)^{-1} C$ is stable in the neighborhood of P.*

Proof See the "Appendix". □

The conditions for the existence of the unique stable solvent P are Blanchard and Kahn's (1980) celebrated rank and order conditions, see Lan and Meyer-Gohde (2014) and Meyer-Gohde (2023) for the conditions expressed in terms of the general class of multivariate singular leading A models pervasive in the literature today. So conditional on its existence, the Bernoulli method above will converge asymptotially to P.

Equally interesting is the performance in the absence of a unique stable solvent P, either in the case of nonexistence or, especially, indeterminacy. In the latter case, the model permits an entire continuum of solutions and some studies, such as Lubik and Schorfheide (2003), analyze the implications of the model in such a case. Both the results here, theorem 1, and Higham and Kim's (2000) approach in the absence of singular coefficient matrices, show that the Bernoulli method converges to the minimal solvent. That is, that solvent P of $0 = M(P) \equiv A P^2 + B P + C$ in such that the eigenvalues of P the smallest latent roots, λ of $M(\lambda)$, that is, with the smallest possible spectral radius. In the case of indeterminacy, more latent roots than the dimension of the problem (i.e., more than n_y the length of the vector y_t and the number of equations in f), the Bernoulli method will return that stable solution with the smallest roots, leaving the largest stable root(s) out of the solution. For a non existence, the Bernoulli method will return that solution with the smallest (in absolute value) roots again, but some of them with be outside the unit circle and hence the solution returned will be unstable. This follows locally from theorem 1 as the crossproducts of the n_y smallest latest roots with inverses of the n_y largest latest roots will be bounded above by one generically.[8]

While the convergence to a specific solvent (in this formulation, the unique stable one) is an advantage over Newton methods, which cannot guarantee convergence to a particular solvent (see Higham & Kim, 2001), Bernoulli methods converge only at a linear rate (given above by the ratio of the largest stable and smallest unstable eigenvalues) instead of Newton methods' quadratic rate. In practice, I follow Higham and Kim (2001) and use the relative residual $\left\|M(P_j)\right\|_F / \left(\|A\|_F \left\|P_j^2\right\|_F + \|B\|_F \left\|P_j\right\|_F + \|C\|_F \right) < n_y \epsilon$ to assess whether convergence has occurred.

[8] The exception being if eigenvalues of identical magnitude are included in both sets. Limiting the permissible cases to $|\lambda_1| \leq \ldots \leq \|\lambda_{n_y}| < |\lambda_{n_y+1} \leq \|\lambda_{2n_y}|$ rules out this exceptional case.

4.3 Accuracy

The practical forward error bounds of Meyer-Gohde (2023) can be used to assess the accuracy of a computed solution $\hat{P}$

$$\underbrace{\frac{\left\|P-\hat{P}\right\|_F}{\|P\|_F}}_{\text{Forward Error}} \leq \underbrace{\frac{\left\|H_{\hat{P}}^{-1}\text{vec}\left(R_{\hat{P}}\right)\right\|_2}{\left\|\hat{P}\right\|_F}}_{\text{Forward Error Bound 1}} \leq \underbrace{\left\|H_{\hat{P}}^{-1}\right\|_2 \frac{\|R_{\hat{P}}\|_F}{\left\|\hat{P}\right\|_F}}_{\text{Forward Error Bound 2}} \tag{73}$$

where $R_{\hat{P}} = A\hat{P}^2 + B\hat{P} + C$ is the residual of the matrix quadratic and $H_{\hat{P}} = I_{n_y} \otimes \left(A\hat{P} + B\right) + \hat{P}' \otimes A$. Stewart's (1971) separation function, see also Kågström (1994), Kågström and Poromaa (1996), and Chen and Lv (2018), is

$$\text{sep}\left[\left(A, A\hat{P}+B\right), \left(I, -\hat{P}\right)\right] = \min_{\|X\|_F=1} \left\|AX\hat{P} + \left(A\hat{P}+B\right)X\right\|_F \tag{74}$$

$$= \min_{\|\text{vec}(X)\|_2=1} \left\|H_{\hat{P}}\text{vec}(X)\right\|_2 \tag{75}$$

$$= \sigma_{\min}\left(H_{\hat{P}}\right) \leq \min\left|\lambda\left(A, A\hat{P}+B\right) - \lambda\left(\hat{P}\right)\right| \tag{76}$$

where $\lambda\left(A, A\hat{P}+B\right)$ is the spectrum or set of (generalized) eigenvalues of the pencil $\left(A, A\hat{P}+B\right)$ (and, accordingly, $\lambda\left(\hat{P}\right)$ the set of eigenvalues of $\hat{P}$) and the last line holds with equality for $A = I$ and $\hat{P}$ and $\hat{P} + B$ regular—hence, the separation between the two pencils—the smallest singular value of $H_{\hat{P}}$—is generically smaller than the minimal separation between their spectra. Analogously to the generalized Sylvester and algebraic Riccati equations, the separation function provides the natural extension of the conditioning number from standard linear equations to these structured problems, and the a posteriori condition number for the matrix quadratic is given by $\text{sep}^{-1}\left[\left(A, A\hat{P}+B\right), \left(I, -\hat{P}\right)\right] = \left\|H_{\hat{P}}^{-1}\right\|_2 = \sigma_{\min}\left(H_{\hat{P}}\right)^{-1}$, which—from above—can be arbitrarily larger than the inverse of the minimal distance between the spectra of the pencils $\left(A, A\hat{P}+B\right), \left(I, -\hat{P}\right)$. This inverse of the separation relates an upper bound to the forward error directly to the residual, like the condition number for a standard linear system, and a tighter bound takes into account the structure more carefully and considers the linear operator $H_{\hat{P}}$ and the residual $R_{\hat{P}}$ jointly.

5 Applications

I conduct two sets of experiments to assess the performance of the Bernoulli algorithms presented above. These two sets are chosen to assess the different methods in a specific, policy relevant model but also in as non-model specific an

environment as possible. This is in contrast to much of the existing literature that compares alternate solution in a single model, frequently a stochastic growth or real business cycle model, such as Taylor and Uhlig (1990), Aruoba et al. (2006), or Caldara et al. (2012) or in a few stylistic models and in Rendahl (2017). To more systematically asses the different Bernoulli based algorithms from above, I compare these algorithms with Dynare's QZ implementation[9] and the baseline Newton algorithm from Meyer-Gohde and Saecker (2024).[10] both in the model of Smets and Wouters (2007) and on the suite of models in the Macroeconomic Model Data Base (MMB) (see Wieland et al., 2012, 2016), a model comparison initiative at the Institute for Monetary and Financial Stability (IMFS),.[11] The performance is measured in terms of accuracy, computational time, and convergence to the stable solvent, initializing both from zero matrix (an uninformed initialization of a stable solvent) and the output from the QZ algorithm.

5.1 Smets and Wouters's (2007) Model

I begin with the medium scale, estimated model of Smets and Wouters (2007) that is arguably the benchmark for policy analysis. In their model they analyze and estimate a New Keynesian DSGE model with US data featuring the usual frictions, sticky prices and wages, inflation indexation, consumption habit formation as well as production frictions concerning investment, capital and fixed costs. Among the equations is the following log-linearized monetary policy rule that will be the focus of the final experiment, assessing the accuracy of the methods here when solving under alternate, but nearby parameterizations.

$$r_t = \rho r_{t-1} + (1-\rho)(r_\pi \pi_t + r_Y(y_t - y_t^p)) + r_{\Delta y}((y_t - y_t^p) - (y_{t-1} - y_{t-1}^p)) + \varepsilon_t^r, \tag{77}$$

This Taylor rule sets the interest rate r_t according to inflation π_t, the current output gap $(y_t - y_t^p)$ and the change in the output gap, with the parameters r_π, r_Y and $r_{\Delta y}$ describing the strength of each of these reactions and ρ controlling the degree of interest rate smoothing. The monetary policy shock, ε_t^r, follows an AR(1)-process with an iid normal error. The Bayesian estimation of the model employs seven macroeconomic time series from the US economy to estimate the model parameters and the authors show that the model matches the US macroeconomic data very closely and that out-of-sample forecasting performance is favorable compared to (B) VAR models.

[9] A further development of Klein (2000), see Villemot (2011). Note that Dynare uses the real Schur decomposition as provided by LAPACK's routine DGGES, see https://git.dynare.org/Dynare/dynare/-/tree/master/mex/sources/mjdgges.

[10] Additionally, note that I follow Dynare and reduce the dimensionality of the problem by grouping variables and structuring the matrix quadratic according to the classification of "static", "purely forward", "purely backward looking", and "mixed" variables. The details are in the appendix and Meyer-Gohde and Saecker (2024).

[11] See http://www.macromodelbase.com

Table 1 Results: model of Smets and Wouters (2007), posterior mode

Method	Relative performance		Forward errors		Iterations
	Run time	Max Abs. Diff	Bound 1	Bound 2	
Dynare (QZ)	0.00063		5.5e−14	2.4e−11	1
Baseline Newton	2.4	108	4.1e−14	1.6e−09	11
Baseline Bernoulli	12	7.7e−13	3.8e−14	2.6e−11	436
Modified Bernoulli (MBI)	132	8.8e−13	3.5e−14	2.5e−11	423
Bernoulli with Line Searches	20	6.9e−13	3.7e−14	2.4e−11	423
Newton–Bernoulli	8.9	108	2.3e−14	1.5e−09	39
Newton–Bernoulli Column	611	6.9e−13	0.49	3.8e+06	2385
Newton–Bernoulli 1/3	11	6.8e−13	4.7e−15	4.1e−12	47
Newton–Bernoulli LS	8.5	6.7e−13	1.1e−14	2.2e−12	33
Newton–Bernoulli Column LS	9.3	7.2e−13	5.1e−15	9.6e−12	32
Newton–Bernoulli LS 1/3	15	5.9e−13	1.1e−14	5.1e−12	57
Newton–Bernoulli Opt	5.6	108	2.3e−14	1e-09	18
Newton–Bernoulli Opt LS	6.5	7.7e−13	1.5e−15	2.2e−12	19

For Dynare, refer to Adjemian et al. (2011)

Run Time for Dynare in seconds, for all others, run time relative to Dynare

Max Abs. Diff. measures the largest absolute difference in the computed P of each method from the P produced by Dynare

Forward error 1 and 2 are the upper bounds for the true forward error, see (73)

Table 1 summarizes the results at the posterior mode calibration of the model of Smets and Wouters (2007). The baseline Bernoulli method takes 12 times longer than Dynare's QZ, which would appear to put it at a disadvantage compared to the baseline Newton, however the large maximal absolute difference to the QZ solution of the Newton algorithm shows that it has converged to a different solution than the unique stable solution found by Dynare's QZ. Indeed, this danger looms with Newton related algorithms as can be seen here for the Newton–Bernoulli and Newton–Bernoulli optimal algorithms, both of which also converged to a different solvent. The baseline Bernoulli required 440 iterations and line searches reduced this number to 420, but the reduction in iterations was outweighed by the costliness of the line search algorithm, resulting altogether in a longer computation time. The columnwise Newton–Bernoulli and the modified Bernoulli both took extraordinarily long times to solve the model, with the Newton–Bernoulli with line searches and optimal Newton–Bernoulli with line searches providing solutions within an order of magnitude of computation time relative to Dynare's QZ, 33 and 19 iterations respectively to do so, and providing solutions that are an order of magnitude more accurate than QZ.

Table 2 assesses the different methods as solution refinement techniques, by parameterizing the model of Smets and Wouters (2007) within the prior to

Table 2 Results: model of Smets and Wouters (2007), numerically problematic parameterization

Method	Relative performance		Forward errors		Iterations
	Run time	Variance π_t	Bound 1	Bound 2	
Dynare (QZ)	0.0046	0.28	1e-11	4.6	1
Baseline Newton	4.3	0.45	2.4e−14	0.00058	4
Baseline Bernoulli	3	0.39	3.8e−13	0.018	90
Modified Bernoulli (MBI)	2814	–	2.9	6.6e+06	5e+04
Bernoulli with Line Searches	504	–	0.99	1.9e+11	5e+04
Newton–Bernoulli	5.1	0.39	3.3e−15	0.00081	8
Newton–Bernoulli Column	3.7	0.4	3.7e−14	0.019	5
Newton–Bernoulli 1/3	1.2	0.46	1.4e−14	0.0027	8
Newton–Bernoulli LS	7.7	0.39	7.9e−15	0.0023	6
Newton–Bernoulli Column LS	5.7	0.44	1.7e−14	0.00079	6
Newton–Bernoulli LS 1/3	1	0.38	3.3e−14	0.0022	8
Newton–Bernoulli Opt	3.7	0.45	2.4e−14	0.00058	4
Newton–Bernoulli Opt LS	4.2	0.5	1.3e−15	0.0011	4

For Dynare, refer to Adjemian et al. (2011)

Run time for Dynare in seconds, for all others, run time relative to Dynare

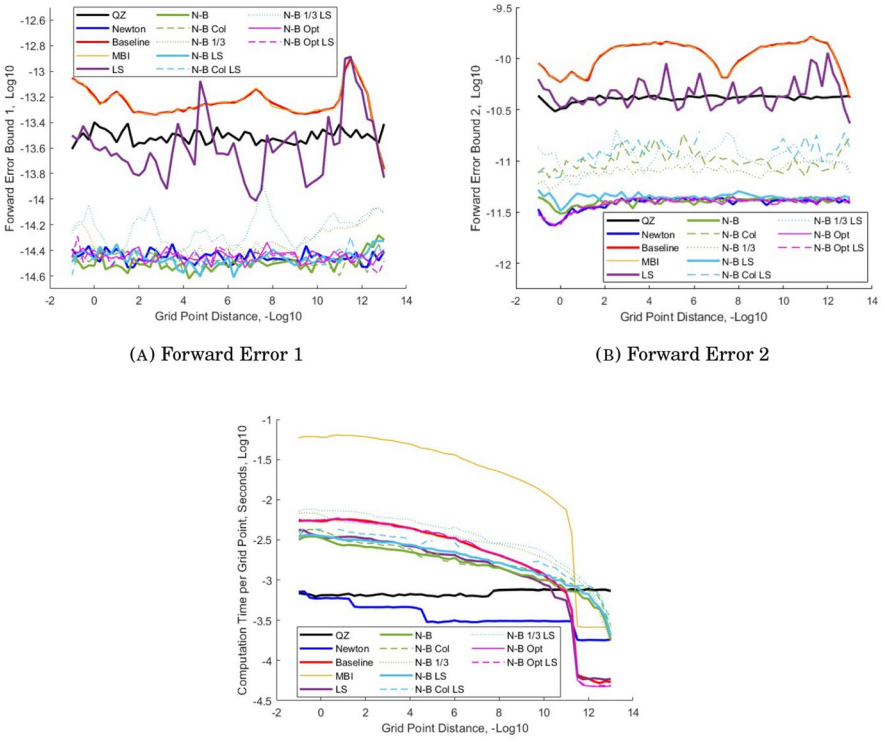
Variance π_t gives the associated value for the population or theoretical variance of inflation—note that two algorithms did not converge to a stable P and hence the variance could not be calculated for them

Forward error 1 and 2 are the upper bounds for the true forward error, see (73)

demonstrate a numerical instability whose consequences are economically relevant, with moments of variables predicted by Dynare's QZ differing in all digits. The second column now displays the variance of inflation as predicted by the different solution methods,[12] At this parameterization, Dynare's QZ solution predicts an inflation variance of 0.28. However, even the lower of the two upper bounds on the forward error is consistent with a numerical instability being several orders of magnitude beyond machine precision.

In the final experiment with the model of Smets and Wouters (2007), I use algorithms to solve iteratively for different parameterizations of the Taylor rule. The goal here is to explore whether solutions from previous, nearby parameterizations can be used to efficiently initialize the Bernoulli methods similarly to the experiment above with the QZ solution as the initial guess. For the parameters determining the Taylor rule reaction to inflation and the long run reaction to the output gap, the experiment iterates through a grid of 10×10 parameter values varying the size of the interval considered—setting $r_\pi \in [1.5, 1.5\ (1 + 10^{-x})]$ and $r_Y \in [0.125, 0.125\ (1 + 10^{-x})]$, where $x \in [-1, 8]$ (Smets & Wouters (2007) calibrate them to $r_\pi = 2.0443$ and

[12] See Meyer-Gohde (2023) for more details on the parameterization. Smets and Wouters (2007) report a variance of inflation in the entire sample of 0.62 and 0.55 and 0.25 in two subsamples.



(A) Forward Error 1

(B) Forward Error 2

(C) Computation Time per Grid Point

Fig. 1 Forward errors and computation time per grid point for different parameterizations of the model by Smets and Wouters (2007). Figure 1a, b plot the upper forward error bounds 1 and 2 against the grid size, log10 scale on both axes. Figure 1c plots the computation per grid point against the number of grid points, log10 scale on both axes

$r_Y = 0.0882$). The algorithm iterates through the two-dimensional grid taking the solution under the previous parameterization as the initialization for the next iteration. A decrease in the spacing between the 100 grid points thus increases the precision of the starting guess, the solution from the previous parameterization.[13]

Figure 1 summarizes the experiment graphically. Figure 1c confirms a decrease in run time per grid point with a narrower grid for the iterative algorithms here and an irrelevance of the grid spacing for QZ. As the grid becomes narrower, the iterative Bernoulli and Newton procedures increasingly benefit from starting from the solution of the previous iteration as it becomes closer to the unknown solution of the current iteration. The QZ algorithm does not operate

[13] Alternatively, one could fix the end points of the grid and increase the number of grid points—this would provide the same message but would possess a computationally prohibitive number of grid points at the narrowest spacing I examine here.

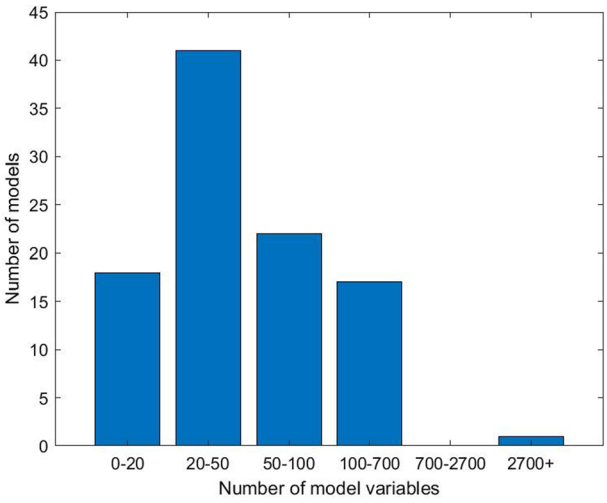


Fig. 2 Histogram over the number of variables for the 99 MMB models. Figure 2 plots the number of model variables over the amount of MMB models. Currently the total amount of models considered is 99

iteratively and, hence, demonstrates no such benefit, solving for each grid point anew. The baseline Newton algorithm moves in a clear step like fashion—i.e., as the number of iterations it requires is few, a saved iteration is visually apparent. All of the Bernoulli algorithms demonstrate the same pattern, fewer iterations are needed as the grid becomes tighter and, when only one iteration is needed, the algorithms take a large step down in computation costs, with the baseline, line search, and optimal algorithms significantly less costly for tight enough grids than both Dynare's QZ and the baseline Newton method. According to Fig. 1a, b, overall, all algorithms involving a Newton step are significantly more precise, the line search equivalently precise, and the baseline and modified Bernoulli somewhat less precise than QZ.

5.2 MMB Suite Comparison

The results above are inherently model specific. While potentially indicative, it is unclear what to expect in other settings. To attenuate this, I turn to the Macroeconomic Model Data Base (MMB) (see Wieland et al., 2012, 2016), a model comparison initiative at the Institute for Monetary and Financial Stability (IMFS) traditionally used to compare the predicted outcomes of different policies across a broad set of macroeconomic models. Version 3.1 contains 151 different models, ranging from small scale, pedagogical models to large scale, estimated models of the US, EU, and multi-country economies. While certainly invaluable for exploring the possible outcomes of policy interventions, this database presents the literature with a useful tool

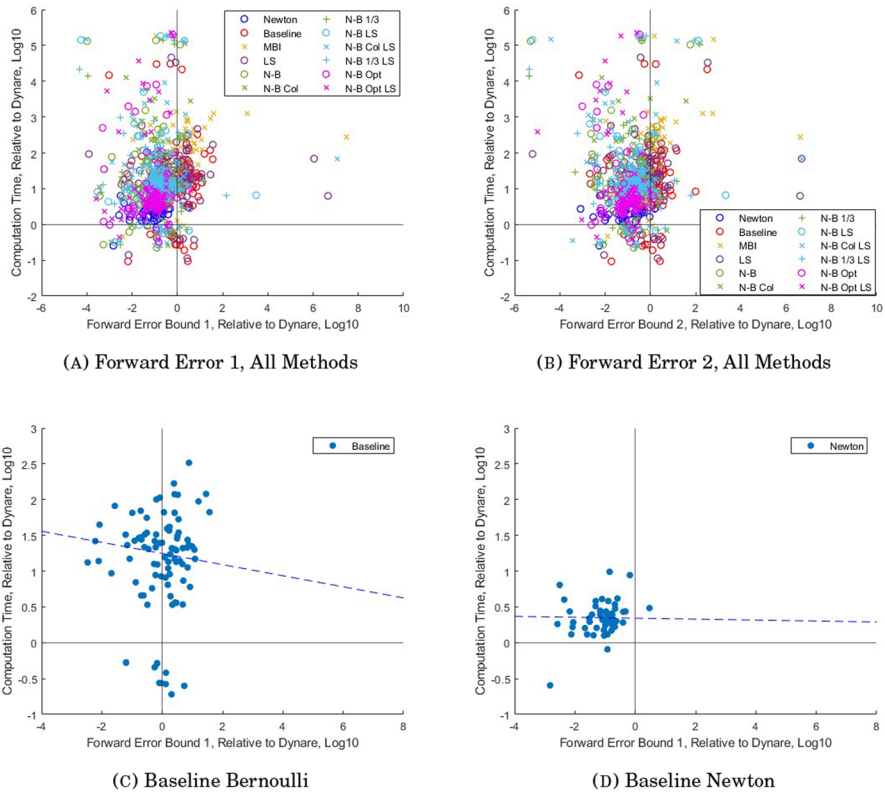
Table 3 Results: 99 MMB models (starting guess: zero-matrix)

Method	Convergence	Run time			Forward Error 1			Forward Error 2			Iterations
		Median	Min	Max	Median	Min	Max	Median	Min	Max	
Dynare (QZ)	99	1	1	1	1	1	1	1	1	1	1
Baseline Newton	53	2.1	0.26	9.8	0.1	5.7e−17	3	0.087	4.1e−17	1.5	8
Baseline Bernoulli	99	21	0.093	3.1e+04	1.3	0.00096	37	1.1	0.0007	325	545
Modified Bernoulli (MBI)	62	224	0.46	1.8e+05	2.2	0.0039	3.1e+07	1.8	0.011	4.1e+06	650
Bernoulli with Line Searches	95	22	0.11	4.6e+04	1	0.00012	4.6e+06	0.96	6.2e−06	4.9e+06	447
Newton–Bernoulli	84	7.2	0.22	1.4e+05	0.1	0.0001	2.2e+12	0.13	5.1e−06	5.2e+28	25
Newton–Bernoulli Column	87	11	0.26	1.5e+05	0.16	0.00035	8.8e+12	0.22	0.003	1.6e+17	33
Newton–Bernoulli 1/3	95	14	0.23	1.4e+05	0.18	0.00011	14	0.19	4.4e−06	1.5e+02	54
Newton–Bernoulli LS	87	11	0.27	1.8e+05	0.12	5.5e−05	3e+12	0.12	6.2e−06	6.9e+13	27
Newton–Bernoulli Column LS	91	14	0.3	1.8e+05	0.14	9.3e−05	3.3e+10	0.24	3.9e−05	8.6e+16	37
Newton–Bernoulli LS 1/3	93	17	0.29	1.8e+05	0.19	4.7e−05	1.4e+02	0.18	4.4e−06	88	51
Newton–Bernoulli Opt	57	3.7	0.32	2e+05	0.098	0.0005	0.95	0.085	0.00098	1.3	10
Newton–Bernoulli Opt LS	69	5.7	0.4	2.3e+05	0.1	0.00026	3.9	0.083	1e-05	0.99	11

For Dynare, refer to Adjemian et al. (2011)

Run time and forward errors relative to Dynare, number of models converging to the stable solution and median of number of iterations in absolute terms

Forward error 1 and 2 are the upper bounds for the true forward error, see (73)



(A) Forward Error 1, All Methods

(B) Forward Error 2, All Methods

(C) Baseline Bernoulli

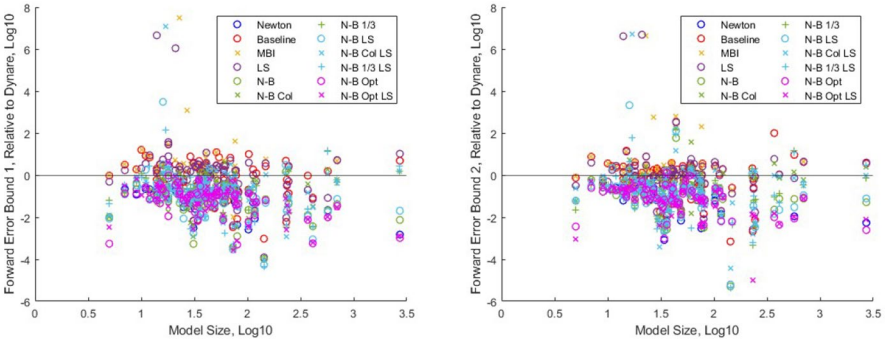
(D) Baseline Newton

Fig. 3 Forward errors and computation time relative to dynare, log10 scales, for the macroeconomic model data base (MMB)

for assessing the potential of different solution methods in a more model-robust context than is currently done in the DSGE literature. Accordingly, I apply the methods of this paper to the set of models appropriate for reproduction,[14] the varying sizes of which are summarized in Fig. 2. Reiterating this point, this is the same suite of models used in Meyer-Gohde and Saecker (2024), Meyer-Gohde (2023), and Huber et al. (2023) which facilitates the comparison of the methods.

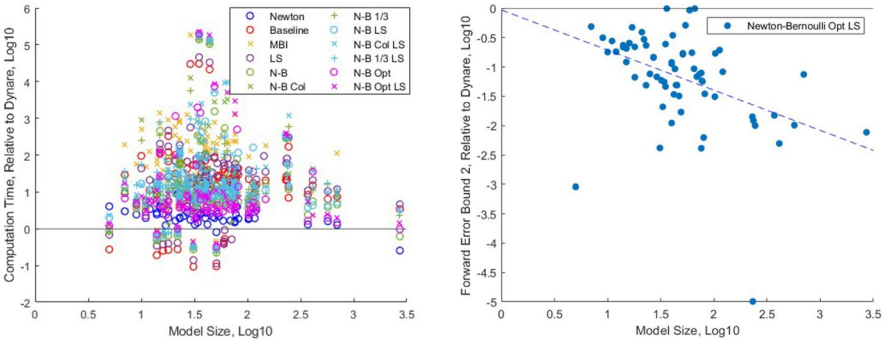
Now I examine the remainder of the models of the MMB and compare the various Bernoulli methods to Dynare's QZ and Newton methods. I solve each applicable model in the MMB 100 times, initializing the methods with a zero matrix and present the results as the average within the middle three quintiles across runs to reduce the effects of outliers.

[14] Currently, this is 99 models, ranging from small scale DSGE models to models from policy institutions containing hundreds of variables. Some of the models in the database are deterministic and/or use nonlinear or non-rational (e.g., adaptive) expectations and, hence, are not appropriate for our comparison here.



(A) Forward Error 1, Relative to Dynare

(B) Forward Error 2, Relative to Dynare

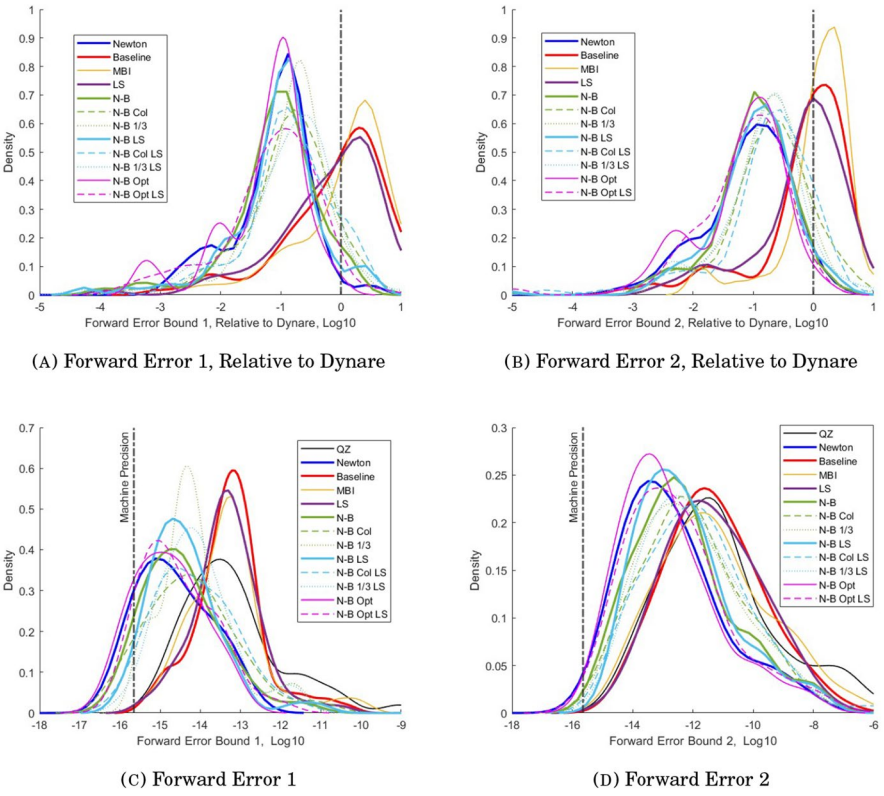


(C) Computation Time, Relative to Dynare

(D) Forward Error 2, Relative to Dynare

Fig. 4 Forward errors, computation time and number of variables for the macroeconomic model data base (MMB). Figure 4a, b plot the upper bounds of the forward error 1 and 2 against model size for all methods, log10 scale on both axes

Table 3 summarizes the results. The first column of results counts the number of models for which the method in question converged to the unique stable solution, highlighting the well-known (Higham & Kim, 2001) drawback of Newton methods, namely the unpredictability of which solution the algorithm will converge to. This problem is not faced by the baseline Bernoulli method, as is to be expected following theorem 1 above, which converged for all models in the MMB. Note that all of the algorithms here performed better than the baseline Newton algorithm of Meyer-Gohde and Saecker (2024) in this respect. Increasing the weight towards the Bernoulli increment (with $p = 1/3$) increases the number of models for the combined Newton–Bernoulli algorithms converged—both with and without line searches. In general, the accuracy is either comparable or improved within an order of magnitude relative to QZ and the computational time is several to about twenty times larger for all the algorithms—apart from the modified Bernoulli algorithm, which again performed poorly, having the highest computational costs and lowest accuracy.



(A) Forward Error 1, Relative to Dynare

(B) Forward Error 2, Relative to Dynare

(C) Forward Error 1

(D) Forward Error 2

Fig. 5 Distribution of forward error bounds relative to dynare for the macroeconomic model data base (MMB). Figure 5a, b plot the distribution of model solutions against the upper bounds of the forward error 1 and 2 for all algorithms, log10 scale on the x axis, 99 MMB models (starting guess: zero matrix)

Figure 3 provides a model-by-model comparison of the different algorithms' performance relative to Dynare's QZ. In Fig. 3a, b, the computation times and forward errors (bound 1 and bound 2) relative to Dynare's QZ are plotted in log10. Hence the cloud of results being primarily in the upper left quadrants leads me to the conclusion that the algorithms are generally more accurate, but computationally more expensive than Dynare's QZ. In Fig. 3c, d the clouds are plotted for the baseline Bernoulli algorithm and baseline Newton individually using bound 1. Here the conclusion regarding the higher accuracy at marginally higher costs for the Newton algorithm of is apparent. While the cloud for the baseline Bernoulli algorithm straddles the y-axis, implying that it does not systematically provide higher accuracy, the cloud's location above the x-axis indicates that it is computationally more expensive than QZ. The negative trendline indicates, however, a tradeoff, whereby higher accuracy is associated with higher computational costs.

Figure 4 continues the model-by-model comparison of the different algorithms' performance relative to Dynare's QZ, but now with a focus on the effect of model size on the algorithms. In Fig. 4a, b, model sizes—as measured by the number of

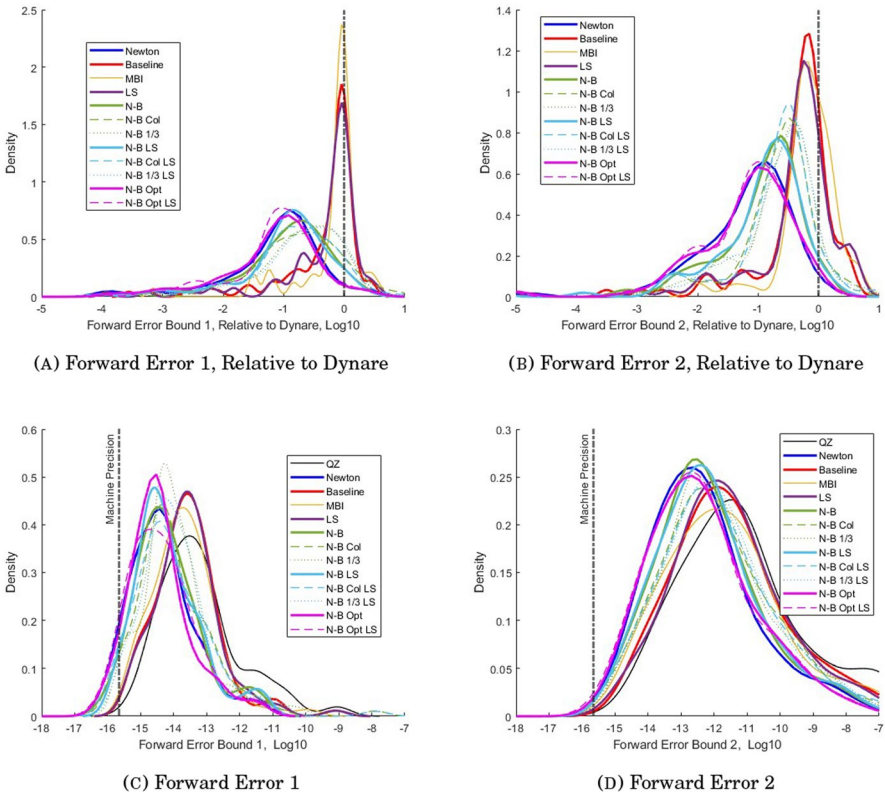
Table 4 Results: 99 MMB models (starting guess: solution Dynare (QZ))

Method	Convergence	Run time			Forward Error 1			Forward Error 2			Iterations
		Median	Min	Max	Median	Min	Max	Median	Min	Max	
Dynare (QZ)	99	1	1	1	1	1	1	1	1	1	1
Baseline Newton	99	0.971	0.0287	290	0.101	3.49e−15	2.79	0.0932	2.49e−15	1.52	1
Baseline Bernoulli	99	0.174	0.00588	3.49e+03	0.825	0.000275	2.91	0.598	0.000315	214	1
Modified Bernoulli (MBI)	77	1.08	0.052	2.37e+04	0.875	0.000694	3.89	0.705	0.0089	875	1
Bernoulli with Line Searches	99	0.316	0.0224	4.4e+03	0.802	0.000117	3.92	0.578	7.94e−06	299	1
Newton–Bernoulli	99	1.03	0.0512	3.19e+04	0.155	0.000138	1.84	0.15	4.47e−06	91.7	3
Newton–Bernoulli Column	98	1.16	0.0475	3.04e+04	0.205	0.000114	16.2	0.296	0.000126	354	3
Newton–Bernoulli 1/3	99	0.921	0.0333	3.49e+04	0.212	8.66e−05	6.5	0.296	4.69e−06	232	3
Newton–Bernoulli LS	99	1.19	0.0923	3.68e+04	0.147	9.27e−05	2.26	0.168	4.19e−06	184	3
Newton–Bernoulli Column LS	98	1.36	0.0917	3.27e+04	0.159	0.000197	17.4	0.286	0.000128	50.4	3
Newton–Bernoulli LS 1/3	99	1.14	0.0255	3.58e+04	0.207	6.54e−05	2.13	0.289	4.8e−06	169	3
Newton–Bernoulli Opt	99	1.57	0.075	3.71e+04	0.0835	3.49e−15	2.22	0.0851	2.49e−15	1.03	3
Newton–Bernoulli Opt LS	99	1.65	0.0775	3.93e+04	0.0894	8.81e−05	2.79	0.086	5.8e−06	1.03	3

For Dynare, refer to Adjemian et al. (2011)

Run Time and foward errors relative to Dynare

Forward error 1 and 2 are the upper bounds for the true forward error, see (73)



(A) Forward Error 1, Relative to Dynare

(B) Forward Error 2, Relative to Dynare

(C) Forward Error 1

(D) Forward Error 2

Fig. 6 Distribution of forward error bounds relative to dynare for the macroeconomic model data base (MMB). Figure 6a, b plot the distribution of model solutions against the upper bounds of the forward error 1 and 2 for all algorithms, log10 scale on the x axis, 99 MMB models (starting guess: solution Dynare(QZ))

endogenous variables—and forward errors (bound 1 and bound 2) relative to QZ are plotted in log10. The most striking result is the difference of accuracy, with algorithms involving a Newton step more often below the x axis than the remaining methods, indicating that they are generally associated with more accuracy—an observation I return to shortly in a density comparison. There appears to be a downward trend, indicating that the methods become more accurate relative to Dynare's QZ for larger models, which is confirmed by looking at Fig. 4d, which shows the clear downward trend for the optimal Bernoulli–Newton method with line searches and is exemplary for many of the methods. Finally, Fig. 4c shows that there appears to be a relationship between the size of the model and the computation time relative to Dynare for at least the very large models towards the right of the figure, indicating that the methods here are likely to be particularly competitive alternatives for larger scale applications, with Meyer-Gohde and Saecker's (2024) Newton algorithm presenting the most convincing evidence in this regard.

Figure 5 provides an overview of the entire distribution of forward errors, the upper row relative to those from Dynare's QZ method and the lower in absolute terms. Forward errors left of the vertical line are thus smaller than Dynare for both figures in the upper row. For both the first, Fig. 5a, and second, Fig. 5b, upper bounds on the forward error, there is an obvious shift to the left of about one order of magnitude for all the methods involving a Newton step and less visually compelling evidence for Bernoulli algorithms not combined with Newton (again, the modified Bernoulli algorithm performs worst). From the lower row, this entails tightening the distributions as well as shifting them closer to machine precision—a lower convergence criterion would allow more iterations and likely bring yet more solutions below machine precision.

To assess the potential for improving on solutions, I repeat the exercise, but now initialize with the solution provided by QZ, see table 4. Here the baseline Bernoulli method is the top performer—with its low per iteration cost, it runs one iteration at a small fraction of the original Dynare QZ cost and provides a significant improvement in accuracy. The modified Bernoulli algorithm again performs unsatisfactorily and, interestingly, the combined Bernoulli and Newton methods all require more than one iteration to converge, which would seem to imply that the Newton and Bernoulli steps individually were generally pulling in different directions in the vicinity of the solution provided by Dynare's QZ. Note that the modified Bernoulli algorithm along with the Newton Bernoulli Column algorithms did not always converge—all three of these algorithms operate column-wise on the problem which apparently can interfere with the convergence, even when initialized close to the solution.

Figure 6, like Fig. 5 but now initialized at the Dynare's QZsolution, provides an overview of the entire distribution of forward errors, the upper row relative to those from Dynare's QZ method and the lower in absolute terms. Forward errors left of the vertical line are thus smaller than Dynare's QZ for both figures in the upper row. For both the first, Fig. 6a, and second, Fig. 6b, upper bounds on the forward error, there is again an obvious shift to the left of about one order of magnitude for all the methods involving a Newton step and a marginal at best improvement using algorithms without a Newton step. This is consistent with the quadratic convergence properties of Newton methods, see Meyer-Gohde and Saecker (2024), when close to a solution. The modified Bernoulli algorithm now performs comparably to the other methods without a Newton step, highlighting some applicability of the conclusions of Bai and Gao (2007) to the DSGE context.

6 Conclusion

I have applied and extended Bernoulli-based methods for solving the matrix quadratic equation underlying the solution of linear DSGE models. This adds a set of alternatives alongside Meyer-Gohde and Saecker's (2024) Newton-based and Huber et al. (2023) doubling algorithms to the current standard of a generalized Schur or QZ decomposition (Moler & Stewart, 1973; Golub & van Loan, 2013). Applying the

methods to the suite of models in the Macroeconomic Model Data Base (MMB), I demonstrate that Bernoulli-based methods are a potential alternative, with a tradeoff between convergence to a particular solution (here the unique, stable solvent) and performance in terms of computational costs.

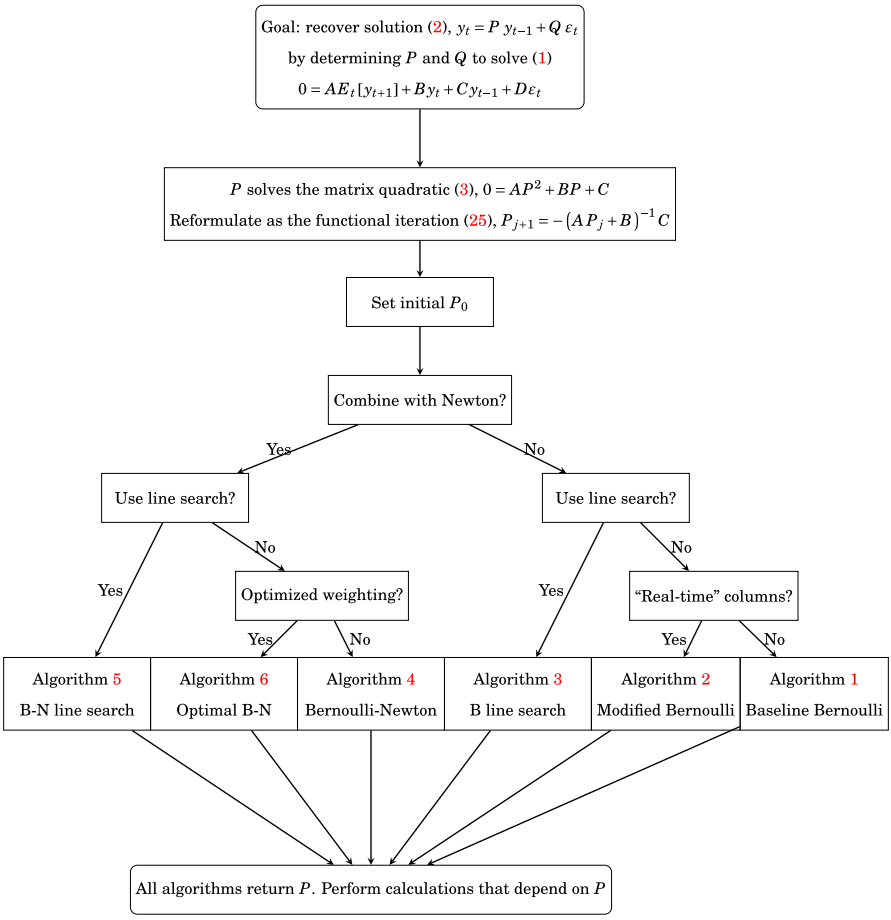
Particularly in iterative environments or when a solution refinement is sought do these algorithms show potential for future application. In filling in an increasingly dense grid of parameterizations for the Taylor rule in the model of Smets and Wouters (2007), iterative methods like the Bernoulli-based methods here can initialize with the solution from the previous parameterization and significantly outperform the current generalized Schur or QZ method both in terms of computational costs and forward error. Taking the solution from QZ as the initialization, the methods provide roughly an order of magnitude improvement in the accuracy of the solution at a fraction of the original computational cost. This initialization and iteration makes applying the set of Bernoulli methods to improving the accuracy of solutions to linear DSGE models a potentially useful direction of application, which is demonstrated here with a problematic parameterization of the Smets and Wouters (2007) model as presented by Meyer-Gohde (2023). Starting with the QZ solution that underpredicts the variance of inflation by about 25% and demonstrates large forward errors, the Bernoulli method refines this solution, providing forward errors several orders of magnitude smaller and a predicted variance of inflation in line with Newton methods.

That being said, the baseline Bernoulli algorithm is not a uniform improvement over the standard QZ method and, in particular, for most models when an informative initialization is not available, the standard QZ is to be preferred. Future research might explore the application of the methods here to reduce the computational burden associated with solving the model for iterative estimation procedures and might be adapted to more quickly and/or accurately perform likelihood calculations or solve heterogenous agent models.

Appendix

Overview

The following flow chart visualizes the decision process that leads to each algorithm.



The code to replicate the analysis can be found at https://github.com/AlexMeyer-Gohde/Linear-DSGE-with-Bernoulli The directory `algorithm` contains the algorithms used here (with the exception of Dynare with can be downloaded at https://www.dynare.org/). The file `bernoulli_matrix_quadratic.m` contains the Bernoulli algorithms introduced here and outlined by the flowchart above. The directory `mmb_experiment` contains the MMB experiment starting at the zero matrix in Table 3 and the Smets and Wouters (2007) comparison of Table 1 (the model of Smets and Wouters (2007) is among the 99 MMB models used in the comparison, themselves contained in the directory `mmb_experiment`).

The directory `mmb_experiment_2` contains the MMB experiment starting at the QZ solution of Table 4. The numerically problematic parameterization of Smets and Wouters (2007) displayed in Table 2 can be found in the directory `improvement_experiment`. Finally the grid size experiment for Smets and Wouters (2007) of Fig. 1 is in the directory `policy_experiment_2`.

Detailed Dynare Variable Classification

Here I summarize the details in the matrix quadratic that follows from the classification of variables from Dynare as laid out in Villemot (2011). See Meyer-Gohde and Saecker (2024) for details.

Subdividing the system of equations in accordance with the QR decomposition yields

$$\underbrace{\begin{array}{c} \\ n^s \\ n^{--} \\ n^m \\ n^{++} \end{array}\!\!\begin{array}{c} \begin{array}{cccc} n^s & n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} \mathbf{0} & \mathbf{0} & \underset{n^s\times n^+}{\check{\mathbf{A}}^+} \\ \mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \underset{n^d\times n^+}{\tilde{\mathbf{A}}^+} \\ \mathbf{0} & \mathbf{0} & \end{bmatrix} \end{array}}_{\underset{n\times n}{A}} \underset{n\times n}{\mathbf{P}^2} + \underbrace{\begin{array}{c} \\ n^s \\ n^{--} \\ n^m \\ n^{++} \end{array}\!\!\begin{array}{c} \begin{array}{cccc} n^s & n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} \underset{n^s\times n^s}{\check{\mathbf{A}}^{0s}} & \underset{n^s\times n^d}{\check{\mathbf{A}}^{0d}} \\ \mathbf{0} & \\ \mathbf{0} & \underset{n^d\times n^d}{\tilde{\mathbf{A}}^0} \\ \mathbf{0} & \end{bmatrix} \end{array}}_{\underset{n\times n}{B}} \underset{n\times n}{\mathbf{P}} + \underbrace{\begin{array}{c} \\ n^s \\ n^{--} \\ n^m \\ n^{++} \end{array}\!\!\begin{array}{c} \begin{array}{ccc} n^s & n^{--} & n^m \quad n^{++} \end{array} \\ \begin{bmatrix} \mathbf{0} & \underset{n^s\times n^-}{\check{\mathbf{A}}^-} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \\ \mathbf{0} & \underset{n^d\times n^-}{\tilde{\mathbf{A}}^-} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \end{bmatrix} \end{array}}_{\underset{n\times n}{C}} = \underset{n\times n}{\mathbf{0}}$$

where n^d is the number of dynamic variables, the sum of number of purely backward-looking, n^{--}, mixed n^m, and purely forward-looking variables, n^{++}. The number of forward-looking variables, n^+, is the sum of the number of mixed, n^m, and purely forward-looking variables, n^{++}, and the number of backward-looking variables, n^-, is the sum of the number of purely backward-looking, n^{--} and mixed variables n^m. Hence, the number of endogenous variables is the sum of the number of static, n^s, and dynamic variables, n^d, or the sum of the number of static, n^s, purely backward-looking, n^{--}, mixed n^m, and purely forward-looking variables, n^{++}. The dimensions satisfy the following

$$\begin{aligned} n^d &= n^{--} + n^m + n^{++}, \quad & n^+ &= n^m + n^{++}, \\ n^- &= n^{--} + n^m, \quad & n &= n^s + n^d = n^s + n^{--} + n^m + n^{++} \end{aligned}$$

The transition matrix, P, from (2) that solves the matrix equation (16) can be subdivided in accordance to Dynare's typology as

$$\mathbf{P} = \begin{array}{c} \\ n^s \\ n^{--} \\ n^m \\ n^{++} \end{array}\!\!\begin{array}{c} \begin{array}{cccc} n^s & n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} \mathbf{P}_{s,s} & \mathbf{P}_{s,--} & \mathbf{P}_{s,m} & \mathbf{P}_{s,++} \\ \mathbf{P}_{--,s} & \mathbf{P}_{--,--} & \mathbf{P}_{--,m} & \mathbf{P}_{--,++} \\ \mathbf{P}_{m,s} & \mathbf{P}_{m,--} & \mathbf{P}_{m,m} & \mathbf{P}_{m,++} \\ \mathbf{P}_{++,s} & \mathbf{P}_{++,--} & \mathbf{P}_{++,m} & \mathbf{P}_{++,++} \end{bmatrix} \end{array} = \begin{array}{c} \\ n \end{array}\!\!\begin{array}{c} \begin{array}{cccc} n^s & n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} \mathbf{P}_{\bullet,s} & \mathbf{P}_{\bullet,--} & \mathbf{P}_{\bullet,m} & \mathbf{P}_{\bullet,++} \end{bmatrix} \end{array} = \begin{array}{c} \\ n^s \\ n^{--} \\ n^m \\ n^{++} \end{array}\!\!\begin{array}{c} \begin{array}{c} n \end{array} \\ \begin{bmatrix} \mathbf{P}_{s,\bullet} \\ \mathbf{P}_{--,\bullet} \\ \mathbf{P}_{m,\bullet} \\ \mathbf{P}_{++,\bullet} \end{bmatrix} \end{array}$$

The matrix quadratic can be expressed as

$$\begin{aligned}\underset{n\times n}{\mathbf{M}}(\underset{n\times n}{\mathbf{P}}) &= \underset{n\times n}{\mathbf{A}}\mathbf{P}^2 + \underset{n\times n}{\mathbf{B}}\,\mathbf{P} + \underset{n\times n}{\mathbf{C}} \\ &= \underbrace{\left(\mathbf{A}\mathbf{P}+\mathbf{B}\right)}_{\equiv\mathbf{G}}\mathbf{P} + \mathbf{C}\end{aligned}$$

For a solvent P of the matrix quadratic, taking the structure of C from the Dynare typology above into account yields

$$\mathbf{M}(\mathbf{P}) = 0 = \mathbf{G}\mathbf{P} + \mathbf{C}$$

$$= \mathbf{G}\;\; \begin{matrix} \\ n \end{matrix}\!\!\begin{array}{c|c|c|c} \overset{n^s}{} & \overset{n^{--}}{} & \overset{n^m}{} & \overset{n^{++}}{} \\ \mathbf{P}_{\bullet,s} & \mathbf{P}_{\bullet,--} & \mathbf{P}_{\bullet,m} & \mathbf{P}_{\bullet,++} \end{array} + \begin{array}{c} \\ n^s \\ n^{--} \\ n^m \\ n^{++} \end{array}\!\!\begin{array}{c|c|c} \overset{n^s}{} & \overset{n^{--}}{} & \overset{n^{++}}{} \\ \mathbf{0} & \underset{n^s\times n^-}{\breve{\mathbf{A}}^-} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \\ \mathbf{0} & \underset{n^d\times n^-}{\tilde{\mathbf{A}}^-} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \end{array}$$

Following Meyer-Gohde and Saecker (2024), who apply corollary 4.5 of Lan and Meyer-Gohde (2014), if P is the unique solvent of $M(P)$ stable with respect to the closed unit circle, $\mathbf{G}$ has full rank and hence the columns of P associated with nonzero columns in C, the static and forward-looking variables are zero $\rightarrow \underset{n\times n^s}{\mathbf{P}_{\bullet,s}} = \underset{n\times n^s}{\mathbf{0}}$, $\underset{n\times n^{++}}{\mathbf{P}_{\bullet,++}} = \underset{n\times n^{++}}{\mathbf{0}}$, whence $\mathbf{P}$ is $\begin{matrix} \\ n \end{matrix}\!\!\begin{array}{c|c|c|c} \overset{n^s}{} & \overset{n^{--}}{} & \overset{n^m}{} & \overset{n^{++}}{} \\ \mathbf{0} & \mathbf{P}_{\bullet,--} & \mathbf{P}_{\bullet,m} & \mathbf{0} \end{array}$ and $\mathbf{M}(\mathbf{P}) = \left[\underset{n\times n^s}{\mathbf{0}}\;\&\underset{n\times n^{--}}{\mathbf{M}(\mathbf{P})^{--}}\;\underset{n\times n^m}{\mathbf{M}(\mathbf{P})^{m}}\;\underset{n\times n^{++}}{\mathbf{0}}\right]$. Consequentially, the first n^s rows of the matrix quadratic, taking $\begin{matrix} \\ n^{--} \\ n^m \\ n^{++} \end{matrix}\!\!\begin{array}{|c|} \overset{n}{} \\ \mathbf{P}_{--,\bullet} \\ \hline \mathbf{P}_{m,\bullet} \\ \hline \mathbf{P}_{++,\bullet} \end{array}$ as given, yield $\begin{matrix} \\ n^s \end{matrix}\!\!\begin{array}{c|c} \overset{n^{--}}{} & \overset{n^m}{} \\ \mathbf{P}_{s,--} & \mathbf{P}_{s,m} \end{array}$ as

$$\begin{matrix} & n^{--} & n^{m} \\ n^{s} & \mathbf{P}_{s,--} & \mathbf{P}_{s,m} \end{matrix} = -\left[\underset{n^s\times n^s}{\breve{\mathbf{A}}^{0s}}\right]^{-1}\left(\underset{n^s\times n^+}{\breve{\mathbf{A}}^{+}}\begin{matrix} & n^{--} & n^{m} \\ n^{m} & \mathbf{P}_{m,--} & \mathbf{P}_{m,m} \\ n^{++} & \mathbf{P}_{++,--} & \mathbf{P}_{++,m} \end{matrix}\begin{matrix} & n^{--} & n^{m} \\ n^{--} & \mathbf{P}_{--,--} & \mathbf{P}_{--,m} \\ n^{m} & \mathbf{P}_{m,--} & \mathbf{P}_{m,m} \end{matrix}\right.$$

$$\left. + \underset{n^s\times n^d}{\breve{\mathbf{A}}^{0d}}\begin{matrix} & n^{--} & n^{m} \\ n^{--} & \mathbf{P}_{--,--} & \mathbf{P}_{--,m} \\ n^{m} & \mathbf{P}_{m,--} & \mathbf{P}_{m,m} \\ n^{++} & \mathbf{P}_{++,--} & \mathbf{P}_{++,m} \end{matrix} + \underset{n^s\times n^-}{\breve{\mathbf{A}}^{-}}\right)$$

and the first n^s rows of P are $\underset{n^s\times n}{\mathbf{P}_{s,\bullet}} = \begin{matrix} & n^{s} & n^{--} & n^{m} & n^{++} \\ n^{s} & \mathbf{0} & \mathbf{P}_{s,--} & \mathbf{P}_{s,m} & \mathbf{0} \end{matrix}$.

The last n^d columns and rows of P solve the reduced matrix quadratic equation

$$\begin{matrix} & n^{--} & n^{m} & n^{++} \\ n^{--} & \mathbf{0} & & \\ n^{m} & \mathbf{0} & \underset{n^d\times n^+}{\tilde{\mathbf{A}}^{+}} & \\ n^{++} & \mathbf{0} & & \end{matrix}\underbrace{\begin{matrix} & n^{--} & n^{m} & n^{++} \\ n^{--} & \mathbf{P}_{--,--} & \mathbf{P}_{--,m} & \mathbf{P}_{--,++} \\ n^{m} & \mathbf{P}_{m,--} & \mathbf{P}_{m,m} & \mathbf{P}_{m,++} \\ n^{++} & \mathbf{P}_{++,--} & \mathbf{P}_{++,m} & \mathbf{P}_{++,++} \end{matrix}}_{\underset{n^d\times n^d}{\tilde{\mathbf{P}}}}\cdot\underset{n^d\times n^d}{\tilde{\mathbf{P}}} + \underset{n^d\times n^d}{\tilde{\mathbf{A}}^{0}}\underset{n^d\times n^d}{\tilde{\mathbf{P}}}$$

$$\begin{matrix} & n^{--} & n^{m} & n^{++} \\ n^{--} & & & \mathbf{0} \\ + \; n^{m} & & \underset{n^d\times n^-}{\tilde{\mathbf{A}}^{-}} & \mathbf{0} \\ n^{++} & & & \mathbf{0} \end{matrix}$$

$$= \underset{n^d\times n^d}{\tilde{\mathbf{M}}(\tilde{\mathbf{P}})} = n^{d}\begin{matrix} & n^{--} & n^{m} & n^{++} \\ & \tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{--} & \tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{m} & \mathbf{0} \end{matrix} = \underset{n^d\times n^d}{\mathbf{0}}$$

Recalling that $\mathbf{P}_{\bullet,++} = \underset{n\times n^{++}}{\mathbf{0}}$, $\tilde{\mathbf{P}}$ can be reduced and two submatrices $\overline{\mathbf{P}}$ and $\widehat{\mathbf{P}}$ defined via

$$\tilde{\mathbf{P}} = \begin{matrix} & n^{--} & n^{m} & n^{++} \\ n^{--} & \mathbf{P}_{--,--} & \mathbf{P}_{--,m} & \mathbf{P}_{--,++} \\ n^{m} & \mathbf{P}_{m,--} & \mathbf{P}_{m,m} & \mathbf{P}_{m,++} \\ n^{++} & \mathbf{P}_{++,--} & \mathbf{P}_{++,m} & \mathbf{P}_{++,++} \end{matrix} = \begin{matrix} & n^{--} & n^{m} & n^{++} \\ n^{--} & \mathbf{P}_{--,--} & \mathbf{P}_{--,m} & \mathbf{0} \\ n^{m} & \mathbf{P}_{m,--} & \mathbf{P}_{m,m} & \mathbf{0} \\ n^{++} & \mathbf{P}_{++,--} & \mathbf{P}_{++,m} & \mathbf{0} \end{matrix} \equiv \begin{matrix} & n^{--} & n^{m} & n^{++} \\ n^{--} & \underset{n^{--}\times n^-}{\overline{\mathbf{P}}} & & \mathbf{0} \\ n^{m} & & & \mathbf{0} \\ n^{++} & \underset{n^+\times n^-}{\widehat{\mathbf{P}}} & & \mathbf{0} \end{matrix}$$

where $\underset{n^{--}\times n^{-}}{\overline{\mathbf{P}}} \equiv \begin{array}{c} \\ n^{--} \end{array}\overset{\begin{array}{cc} n^{--} & n^{m} \end{array}}{\left[\begin{array}{c|c} \mathbf{P}_{--,--} & \mathbf{P}_{--,m} \end{array}\right]}$ and $\underset{n^{+}\times n^{-}}{\widehat{\mathbf{P}}} \equiv \begin{array}{c} n^{m} \\ n^{++} \end{array}\overset{\begin{array}{cc} n^{--} & n^{m} \end{array}}{\left[\begin{array}{c|c} \mathbf{P}_{m,--} & \mathbf{P}_{m,m} \\ \hline \mathbf{P}_{++,--} & \mathbf{P}_{++,m} \end{array}\right]}$ allow the matrix quadratic to be written as

$$\left(\begin{array}{c} \begin{array}{cc} n^{--} & \begin{array}{cc} n^{m} & n^{++} \end{array} \end{array} \\ \begin{array}{c} n^{--} \\ n^{m} \\ n^{++} \end{array}\left[\begin{array}{c|cc} \mathbf{0} & & \\ \hline \mathbf{0} & \underset{n^{d}\times n^{+}}{\tilde{\mathbf{A}}^{+}} & \\ \mathbf{0} & & \end{array}\right] \end{array}\right)\begin{array}{c} \begin{array}{cc} n^{--} & \begin{array}{cc} n^{m} & n^{++} \end{array} \end{array} \\ \begin{array}{c} n^{--} \\ n^{m} \\ n^{++} \end{array}\left[\begin{array}{c|cc} \underset{n^{--}\times n^{-}}{\overline{\mathbf{P}}} & & \mathbf{0} \\ \hline & \underset{n^{+}\times n^{-}}{\widehat{\mathbf{P}}} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \end{array}\right] \end{array} + \underset{n^{d}\times n^{d}}{\tilde{\mathbf{A}}^{0}}\left(\begin{array}{c} \begin{array}{cc} n^{--} & \begin{array}{cc} n^{m} & n^{++} \end{array} \end{array} \\ \begin{array}{c} n^{--} \\ n^{m} \\ n^{++} \end{array}\left[\begin{array}{c|cc} \underset{n^{--}\times n^{-}}{\overline{\mathbf{P}}} & & \mathbf{0} \\ \hline & \underset{n^{+}\times n^{-}}{\widehat{\mathbf{P}}} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \end{array}\right] \end{array}\right)$$

$$+ \underset{n^{d}\times n^{-}}{\tilde{\mathbf{A}}^{-}} = \underset{n^{d}\times n^{d}}{\tilde{\mathbf{M}}(\tilde{\mathbf{P}})} = \begin{array}{c} \\ n^{d} \end{array}\overset{\begin{array}{ccc} n^{--} & n^{m} & n^{++} \end{array}}{\left[\begin{array}{c|c|c} \tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{--} & \tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{m} & \mathbf{0} \end{array}\right]} = \underset{n^{d}\times n^{d}}{\mathbf{0}}$$

which can be reduced to

$$\left(\underset{n^{d}\times n^{+}}{\tilde{\mathbf{A}}^{+}} \begin{array}{c} \\ n^{+} \end{array}\overset{\begin{array}{cc} n^{-} & n^{++} \end{array}}{\left[\begin{array}{c|c} \underset{n^{+}\times n^{-}}{\widehat{\mathbf{P}}} & \mathbf{0} \end{array}\right]} + \underset{n^{d}\times n^{d}}{\tilde{\mathbf{A}}^{0}}\right)\begin{array}{c} \begin{array}{c} n^{-} \end{array} \\ \begin{array}{c} n^{--} \\ n^{+} \end{array}\left[\begin{array}{c} \underset{n^{--}\times n^{-}}{\overline{\mathbf{P}}} \\ \hline \underset{n^{+}\times n^{-}}{\widehat{\mathbf{P}}} \end{array}\right] \end{array} + \underset{n^{d}\times n^{-}}{\tilde{\mathbf{A}}^{-}} = \begin{array}{c} \\ n^{d} \end{array}\overset{\begin{array}{cc} n^{--} & n^{m} \end{array}}{\left[\begin{array}{c|c} \tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{--} & \tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{m} \end{array}\right]} = \underset{n^{d}\times n^{-}}{\mathbf{0}}$$

This leads to the Bernoulli iteration

$$\begin{array}{c} \begin{array}{c} n^{-} \end{array} \\ \begin{array}{c} n^{--} \\ n^{+} \end{array}\left[\begin{array}{c} \underset{}{\overline{\mathbf{P}}_{j}} \\ \hline \underset{}{\widehat{\mathbf{P}}_{j}} \end{array}\right] \end{array} = -\left(\underset{n^{d}\times n^{+}}{\tilde{\mathbf{A}}^{+}} \begin{array}{c} \\ n^{+} \end{array}\overset{\begin{array}{cc} n^{-} & n^{++} \end{array}}{\left[\begin{array}{c|c} \underset{n^{+}\times n^{-}}{\widehat{\mathbf{P}}_{j-1}} & \mathbf{0} \end{array}\right]} + \underset{n^{d}\times n^{d}}{\tilde{\mathbf{A}}^{0}}\right)^{-1} \underset{n^{d}\times n^{-}}{\tilde{\mathbf{A}}^{-}}$$

Detailed Dynare Topology-Line Search

The line search methods in the text require finding the minimum of the merit function

$$\begin{aligned}
\|M(P + x\Delta P)\|_F^2 &= (1-x)^2\|M(P)\|_F^2 + x^2\|A\Delta P(x\Delta P + P)\|_F^2 \\
&\quad + (1-x)x * \mathrm{tr}(M(P)^* A\Delta P(x\Delta P + P) \\
&\quad + (x\Delta P^* + xP^*)\Delta P^* A^* M(P)) \\
&= (1-x)^2\|M(P)\|_F^2 + x^2\|A\Delta P(x\Delta P + P)\|_F^2 \\
&\quad + (1-x)x^2 * \mathrm{tr}(M(P)^* A\Delta P^2 + \Delta P^{*\,2} A^* M(P)) \\
&\quad + (1-x)x * \mathrm{tr}(M(P)^* A\Delta P \cdot P + P^* \Delta P^* A^* M(P)) \\
&= (1-x)^2\|M(P)\|_F^2 + x^4\left\|A\Delta P^2\right\|_F^2 + x^2\|A\Delta P \cdot P\|_F^2 \\
&\quad + x^3 * \mathrm{tr}(\Delta P^{*\,2} A^* A\Delta P \cdot P + P^* \Delta P^* A^* A\Delta P^2) \\
&\quad + (1-x)x^2 * \mathrm{tr}(M(P)^* A\Delta P^2 + \Delta P^{*\,2} A^* M(P)) \\
&\quad + (1-x)x * \mathrm{tr}(M(P)^* A\delta P \cdot P + P^* \Delta P^* A^* M(P)) \\
&\equiv g(x)
\end{aligned}$$

The first order condition for an interior solution requires finding the zeros of the polynomial

$$g'(x) = 4\gamma x^3 + 3(\xi - \sigma)x^2 + 2(\alpha + \beta - \delta + \sigma)x + \delta - 2\alpha \tag{A1}$$

where $\alpha = \|M(P)\|_F^2$, $\beta = \|Ad\mathbf{P} \cdot P\|_F^2$, $\gamma = \left\|A(d\mathbf{P})^2\right\|_F^2$, $\xi = \mathrm{tr}\left((Ad\mathbf{P} \cdot \mathbf{P})^* A(d\mathbf{P})^2 + \left(A(d\mathbf{P})^2\right)^* Ad\mathbf{P} \cdot \mathbf{P}\right)$, $\sigma = \mathrm{tr}\left(M(P)^* A(d\mathbf{P})^2 + \left(A(d\mathbf{P})^2\right)^* M(P)\right)$, and $\delta = \mathrm{tr}\left(M(P)^* Ad\mathbf{P} \cdot \mathbf{P} + (Ad\mathbf{P} \cdot \mathbf{P})^* M(P)\right)$, which follows from

$$
\begin{aligned}
g'(x) = {} & 2(1-x)(-1)\|M(P)\|_F^2 \\
& + 2x\|A\Delta P \cdot P\|_F^2 + 4x^3\left\|A\Delta P^2\right\|_F^2 \\
& + 3x^2 * \mathrm{tr}(\Delta P^{*\,2} A^* A\Delta P \cdot P + P^* \Delta P^* A^* A\Delta P^2) \\
& + [(1-x)2x + (-1)x^2] * \mathrm{tr}(M(P)^* A\Delta P^2 + \Delta P^{*\,2} A^* M(P)) \\
& + [(1-x)x + (-1)x] * \mathrm{tr}(M(P)^* A\Delta P \cdot P + P^* \Delta P^* A^* M(P)) \\
= {} & -2\|M(P)\|_F^2 + 2\|M(P)\|_F^2 x + 2\|A\Delta P \cdot P\|_F^2 x + 4\left\|A\Delta P^2\right\|_F^2 x^3 \\
& + 3 * \mathrm{tr}(\Delta P^{*\,2} A^* A\Delta P \cdot P + P^* \Delta P^* A^* A\Delta P^2)x^2 \\
& + \mathrm{tr}(M(P)^* A\Delta P \cdot P + P^* \Delta P^* A^* M(P)) - 2 * \mathrm{tr}(M(P)^* A\Delta P \cdot P + P^* \Delta P^* A^* M(P))x \\
& + 2 * \mathrm{tr}(M(P)^* A\Delta P^2 + \Delta P^{*\,2} A^* M(P))x \\
& - 3 * \mathrm{tr}(M(P)^* A\Delta P^2 + \Delta P^{*\,2} A^* M(P))x^2 \\
= {} & 4\left\|A\Delta P^2\right\|_F^2 x^3 + (3 * \mathrm{tr}(\Delta P^{*\,2} A^* A\Delta P \cdot P + P^* \Delta P^* A^* A\Delta P^2) \\
& - 3\mathrm{tr}(M(P)^* A\Delta P^2 + \Delta P^{*\,2} A^* M(P)))x^2 \\
& + 2\|M(P)\|_F^2 x + 2\|A\Delta P \cdot P\|_F^2 x \\
& - 2 * \mathrm{tr}(M(P)^* A\Delta P \cdot P + P^* \Delta P^* A^* M(P))x \\
& + 2 * \mathrm{tr}(M(P)^* A\Delta P^2 + \Delta P^{2*} A^* M(P))x \\
& + \mathrm{tr}(M(P)^* A\Delta P \cdot P + P^* \Delta P^* A^* M(P)) - 2\|M(P)\|_F^2
\end{aligned}
$$

Using the typology from Dynare and the results above

$$\alpha = ||\mathbf{M}(\mathbf{P})||_F^2 = \mathrm{tr}\left(\mathbf{M}(\mathbf{P})^* \mathbf{M}(\mathbf{P})\right)$$

$$= \mathrm{tr}\begin{pmatrix} & n & \\ n^s & \mathbf{0} & \\ n^{--} & \mathbf{M}(\mathbf{P})^*_{--} & \\ n^m & \mathbf{M}(\mathbf{P})^*_m & \\ n^{++} & \mathbf{0} & \end{pmatrix} n \begin{bmatrix} n^s & n^{--} & n^m & n^{++} \\ \mathbf{0} & \mathbf{M}(\mathbf{P})_{--} & \mathbf{M}(\mathbf{P})_m & \mathbf{0} \end{bmatrix}$$

$$= \mathrm{tr}\left(\mathbf{M}(\mathbf{P})^*_{--}\mathbf{M}(\mathbf{P})_{--}\right) + \mathrm{tr}\left(\mathbf{M}(\mathbf{P})^*_m \mathbf{M}(\mathbf{P})_m\right)$$

$$\underset{n\times n}{\mathbf{M}}(\underset{n\times n}{\mathbf{P}}) = \underset{n\times n}{\mathbf{A}}\,\mathbf{P}^2 + \underset{n\times n}{\mathbf{B}}\,\mathbf{P} + \underset{n\times n}{\mathbf{C}}$$

$$= \begin{array}{c} \begin{matrix} & n^s & n^{--} & n^m & n^{++} \end{matrix} \\ \begin{matrix} n^s \\ n^{--} \\ n^m \\ n^{++} \end{matrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \underset{n^s\times n^+}{\breve{\mathbf{A}}^+} \\ \mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \underset{n^d\times n^+}{\tilde{\mathbf{A}}^+} \\ \mathbf{0} & \mathbf{0} & \end{bmatrix} \end{array} n \begin{bmatrix} n^s & n^{--} & n^m & n^{++} \\ \mathbf{0} & \mathbf{PP}_{\bullet,--} & \mathbf{PP}_{\bullet,m} & \mathbf{0} \end{bmatrix}$$

$$+ \begin{array}{c} \begin{matrix} & n^s & n^{--} & n^m & n^{++} \end{matrix} \\ \begin{matrix} n^s \\ n^{--} \\ n^m \\ n^{++} \end{matrix} \begin{bmatrix} \underset{n^s\times n^s}{\breve{\mathbf{A}}^{0s}} & \underset{n^s\times n^d}{\breve{\mathbf{A}}^{0d}} & & \\ \mathbf{0} & & & \\ \mathbf{0} & \underset{n^d\times n^d}{\tilde{\mathbf{A}}^0} & & \\ \mathbf{0} & & & \end{bmatrix} \end{array} n \begin{bmatrix} n^s & n^{--} & n^m & n^{++} \\ \mathbf{0} & \mathbf{P}_{\bullet,--} & \mathbf{P}_{\bullet,m} & \mathbf{0} \end{bmatrix}$$

$$+ \begin{array}{c} \begin{matrix} & n^s & n^{--} & n^m & n^{++} \end{matrix} \\ \begin{matrix} n^s \\ n^{--} \\ n^m \\ n^{++} \end{matrix} \begin{bmatrix} \mathbf{0} & \underset{n^s\times n^-}{\breve{\mathbf{A}}^-} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \\ \mathbf{0} & \underset{n^d\times n^-}{\tilde{\mathbf{A}}^-} & \mathbf{0} \\ \mathbf{0} & & \mathbf{0} \end{bmatrix} \end{array}$$

$$= n \begin{bmatrix} n^s & & n^- & & n^{++} \\ \mathbf{0} & \begin{bmatrix} \mathbf{0}_{n^s\times n^s} & \mathbf{0}_{n^s\times n^{--}} & \underset{n^s\times n^+}{\breve{\mathbf{A}}^+} \\ \mathbf{0}_{n^{--}\times n^s} & \mathbf{0}_{n^{--}\times n^{--}} & \\ \mathbf{0}_{n^m\times n^s} & \mathbf{0}_{n^m\times n^{--}} & \underset{n^d\times n^+}{\tilde{\mathbf{A}}^+} \\ \mathbf{0}_{n^{++}\times n^s} & \mathbf{0}_{n^{++}\times n^{--}} & \end{bmatrix} & \underset{n\times n}{\mathbf{P}} \begin{bmatrix} \underset{n\times n^{--}}{\mathbf{P}_{\bullet,--}} & \underset{n\times n^m}{\mathbf{P}_{\bullet,m}} \end{bmatrix} & \mathbf{0} \end{bmatrix}$$

$$+ n \begin{bmatrix} n^s & & n^- & & n^{++} \\ \mathbf{0} & \begin{bmatrix} \underset{n^s\times n^s}{\breve{\mathbf{A}}^{0s}} & \underset{n^s\times n^d}{\breve{\mathbf{A}}^{0d}} \\ \mathbf{0}_{n^{--}\times n^s} & \\ \mathbf{0}_{n^m\times n^s} & \underset{n^d\times n^d}{\tilde{\mathbf{A}}^0} \\ \mathbf{0}_{n^{++}\times n^s} & \end{bmatrix} \begin{bmatrix} \underset{n\times n^{--}}{\mathbf{P}_{\bullet,--}} & \underset{n\times n^m}{\mathbf{P}_{\bullet,m}} \end{bmatrix} & \mathbf{0} \end{bmatrix} + n \begin{bmatrix} n^s & n^- & n^{++} \\ \mathbf{0} & \begin{bmatrix} \underset{n^s\times n^-}{\breve{\mathbf{A}}^-} \\ \underset{n^d\times n^-}{\tilde{\mathbf{A}}^-} \end{bmatrix} & \mathbf{0} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{0}_{n^s\times n^s} & \mathbf{0}_{n^s\times n^{--}} & \underset{n^s\times n^+}{\breve{\mathbf{A}}^+} \\ \mathbf{0}_{n^{--}\times n^s} & \mathbf{0}_{n^{--}\times n^{--}} & \\ \mathbf{0}_{n^m\times n^s} & \mathbf{0}_{n^m\times n^{--}} & \underset{n^d\times n^+}{\tilde{\mathbf{A}}^+} \\ \mathbf{0}_{n^{++}\times n^s} & \mathbf{0}_{n^{++}\times n^{--}} & \end{bmatrix} \underset{n\times n}{\mathbf{P}} \begin{bmatrix} \underset{n\times n^{--}}{\mathbf{P}_{\bullet,--}} & \underset{n\times n^m}{\mathbf{P}_{\bullet,m}} \end{bmatrix} = \begin{bmatrix} \underset{n^s\times n^+}{\breve{\mathbf{A}}^+} \\ \underset{n^d\times n^+}{\tilde{\mathbf{A}}^+} \end{bmatrix} \underset{n^+\times n}{\mathbf{P}_{m/++,\bullet}}\, \underset{n\times n^-}{\mathbf{P}_{\bullet,--/m}}$$

where

$$\underset{n\times n^{--}}{\mathbf{P}_{\bullet,--/m}} = n\left[\begin{array}{c|c} \overset{n^{--}}{\mathbf{P}_{\bullet,--}} & \overset{n^{m}}{\mathbf{P}_{\bullet,m}} \end{array}\right] \quad \text{and} \quad \underset{n\times n^{+}}{\mathbf{P}_{\bullet,m/++}} = n\left[\begin{array}{c|c} \overset{n^{m}}{\mathbf{P}_{\bullet,m}} & \overset{n^{++}}{\mathbf{P}_{\bullet,++}} \end{array}\right]$$

$$\underset{n\times n}{\mathbf{M}(\mathbf{P})} = n\left[\begin{array}{c|c|c} \overset{n^{s}}{\mathbf{0}} & \left(\left[\begin{array}{c} \underset{n^{s}\times n^{+}}{\breve{\mathbf{A}}^{+}} \\ \\ \underset{n^{d}\times n^{+}}{\tilde{\mathbf{A}}^{+}} \end{array}\right]\underset{n^{+}\times n}{\mathbf{P}_{m/++,\bullet}} + \overset{n^{-}}{\left[\begin{array}{cc} \underset{n^{s}\times n^{s}}{\breve{\mathbf{A}}^{0s}} & \underset{n^{s}\times n^{d}}{\breve{\mathbf{A}}^{0d}} \\ \underset{n^{--}\times n^{s}}{\mathbf{0}} & \\ \underset{n^{m}\times n^{s}}{\mathbf{0}} & \underset{n^{d}\times n^{d}}{\tilde{\mathbf{A}}^{0}} \\ \underset{n^{++}\times n^{s}}{\mathbf{0}} & \end{array}\right]}\right)\underset{n\times n^{--}}{\mathbf{P}_{\bullet,--/m}} + \left[\begin{array}{c} \underset{n^{s}\times n^{-}}{\breve{\mathbf{A}}^{-}} \\ \underset{n^{d}\times n^{-}}{\tilde{\mathbf{A}}^{-}} \end{array}\right] & \overset{n^{++}}{\mathbf{0}} \end{array}\right]$$

$$= n\left[\begin{array}{c|c|c} \overset{n^{s}}{\mathbf{0}} & \overset{n^{-}}{\mathbf{X}} & \overset{n^{++}}{\mathbf{0}} \end{array}\right]$$

where $\mathbf{X}$ is defined as

$$\underset{n\times n^{-}}{\mathbf{X}} \equiv \begin{matrix} n^{s} \\ n^{d} \end{matrix}\overset{n^{-}}{\left[\begin{array}{c} \mathbf{X}_1 \\ \mathbf{X}_2 \end{array}\right]} \equiv \begin{matrix} n^{s} \\ n^{d} \end{matrix}\overset{n^{-}}{\left[\begin{array}{c} \left(\underset{n^{s}\times n^{+}}{\breve{\mathbf{A}}^{+}}\underset{n^{+}\times n}{\mathbf{P}_{m/++,\bullet}} + \left[\begin{array}{cc} \underset{n^{s}\times n^{s}}{\breve{\mathbf{A}}^{0s}} & \underset{n^{s}\times n^{d}}{\breve{\mathbf{A}}^{0d}} \end{array}\right]\right)\underset{n\times n^{--}}{\mathbf{P}_{\bullet,--/m}} + \underset{n^{s}\times n^{-}}{\breve{\mathbf{A}}^{-}} \\ \underset{n^{d}\times n^{+}}{\tilde{\mathbf{A}}^{+}}\underset{n^{+}\times n}{\mathbf{P}_{m/++,\bullet}}\underset{n\times n^{--}}{\mathbf{P}_{\bullet,--/m}} + \left[\begin{array}{cc} \underset{n^{d}\times n^{s}}{\mathbf{0}} & \underset{n^{d}\times n^{d}}{\tilde{\mathbf{A}}^{0}} \end{array}\right]\underset{n\times n^{--}}{\mathbf{P}_{\bullet,--/m}} + \underset{n^{d}\times n^{-}}{\tilde{\mathbf{A}}^{-}} \end{array}\right]}$$

$$= \begin{matrix} n^{s} \\ n^{d} \end{matrix}\overset{n^{-}}{\left[\begin{array}{c} \left(\underset{n^{s}\times n^{+}}{\breve{\mathbf{A}}^{+}}\underset{n^{+}\times n}{\mathbf{P}_{m/++,\bullet}} + \left[\begin{array}{cc} \underset{n^{s}\times n^{s}}{\breve{\mathbf{A}}^{0s}} & \underset{n^{s}\times n^{d}}{\breve{\mathbf{A}}^{0d}} \end{array}\right]\right)\underset{n\times n^{--}}{\mathbf{P}_{\bullet,--/m}} + \underset{n^{s}\times n^{-}}{\breve{\mathbf{A}}^{-}} \\ \underset{n^{d}\times n^{+}}{\tilde{\mathbf{A}}^{+}}\underset{n^{+}\times n}{\mathbf{P}_{m/++,\bullet}}\underset{n\times n^{--}}{\mathbf{P}_{\bullet,--/m}} + \underset{n^{d}\times n^{d}}{\tilde{\mathbf{A}}^{0}}\underset{n^{d}\times n^{--}}{\mathbf{P}_{d,--/m}} + \underset{n^{d}\times n^{-}}{\tilde{\mathbf{A}}^{-}} \end{array}\right]}$$

$$= \begin{matrix} n^{s} \\ n^{d} \end{matrix}\overset{n^{-}}{\left[\begin{array}{c} \underset{n^{s}\times n^{s}}{\breve{\mathbf{A}}^{0s}}\underset{n^{s}\times n^{--}}{\mathbf{P}_{s,--/m}} + \left(\underset{n^{s}\times n^{+}}{\breve{\mathbf{A}}^{+}}\underset{n^{+}\times n^{d}}{\mathbf{P}_{m/++,d}} + \underset{n^{s}\times n^{d}}{\breve{\mathbf{A}}^{0d}}\right)\underset{n^{d}\times n^{--}}{\mathbf{P}_{d,--/m}} + \underset{n^{s}\times n^{-}}{\breve{\mathbf{A}}^{-}} \\ \underset{n^{d}\times n^{+}}{\tilde{\mathbf{A}}^{+}}\underset{n^{+}\times n}{\mathbf{P}_{m/++,\bullet}}\underset{n\times n^{--}}{\mathbf{P}_{\bullet,--/m}} + \underset{n^{d}\times n^{d}}{\tilde{\mathbf{A}}^{0}}\underset{n^{d}\times n^{--}}{\mathbf{P}_{d,--/m}} + \underset{n^{d}\times n^{-}}{\tilde{\mathbf{A}}^{-}} \end{array}\right]}$$

Accordingly

$$\alpha = ||\mathbf{M}(\mathbf{P})||_F^2 = \text{tr}\left(\mathbf{M}(\mathbf{P})^*\mathbf{M}(\mathbf{P})\right) = \text{tr}\,(\mathbf{X}^*\mathbf{X}) = \text{tr}\,(\mathbf{X}_1^*\mathbf{X}_1) + \text{tr}\,(\mathbf{X}_2^*\mathbf{X}_2)$$

by construction $\underset{n^s\times n^-}{\mathbf{X}_1} = \underset{n^s\times n^-}{\mathbf{0}}$ and hence

$$\alpha = ||\mathbf{M}(\mathbf{P})||_F^2 = \text{tr}\,(\mathbf{X}_2^*\mathbf{X}_2) = \text{tr}\left(\tilde{\mathbf{M}}(\tilde{\mathbf{P}})^*\tilde{\mathbf{M}}(\tilde{\mathbf{P}})\right)$$

Turning to $\gamma = \left|\left|\mathbf{A}d\mathbf{P}^2\right|\right|_F^2$

$$\gamma = \left|\left|\mathbf{A}d\mathbf{P}^2\right|\right|_F^2 = \text{tr}\left(\left(\mathbf{A}d\mathbf{P}^2\right)^*\mathbf{A}d\mathbf{P}^2\right)$$

$$\mathbf{A}d\mathbf{P}^2 = \overset{n^s}{\underset{}{}} \quad \overset{n^-}{\underset{}{}} \quad \overset{n^{++}}{\underset{}{}}$$

$$= n\left[\begin{array}{c|c|c} \mathbf{0} & \underset{n\times n}{\mathbf{A}}\, d\, \underset{n\times n}{\mathbf{P}}\, d\underset{n\times n^-}{\mathbf{P}_{\bullet,--/m}} & \mathbf{0} \end{array}\right]$$

$$= n\left[\begin{array}{c|c|c} \mathbf{0} & \underset{n^s\times n^+}{\begin{bmatrix}\breve{\mathbf{A}}^+\\ \tilde{\mathbf{A}}^+\end{bmatrix}} d\underset{n^+\times n}{\mathbf{P}_{m/++,\bullet}} d\underset{n\times n^-}{\mathbf{P}_{\bullet,--/m}} & \mathbf{0} \end{array}\right]$$

$$= n\left[\begin{array}{c|c|c} \mathbf{0} & \underset{n^s\times n^+}{\begin{bmatrix}\breve{\mathbf{A}}^+\\ \tilde{\mathbf{A}}^+\end{bmatrix}} d\underset{n^+\times n^-}{\mathbf{P}_{m/++,--/m}} d\underset{n^-\times n^-}{\mathbf{P}_{--/m,--/m}} & \mathbf{0} \end{array}\right]$$

$$= n\left[\begin{array}{c|c|c} \mathbf{0} & \underset{n^s\times n^-}{\begin{bmatrix}\mathbf{Y}_1\\ \mathbf{Y}_2\end{bmatrix}} & \mathbf{0} \end{array}\right]$$

Hence, γ is

$$\gamma = \text{tr}\,(\mathbf{Y}_1^*\mathbf{Y}_1) + \text{tr}\,(\mathbf{Y}_2^*\mathbf{Y}_2)$$

By analogy, $\beta = ||\mathbf{A}d\mathbf{P}\cdot\mathbf{P}||_F^2$ is given by

$$\beta = \text{tr}\,(\mathbf{Z}_1^*\mathbf{Z}_1) + \text{tr}\,(\mathbf{Z}_2^*\mathbf{Z}_2)$$

where

$$\underset{n^s\times n^-}{\begin{bmatrix}\mathbf{Z}_1\\ \mathbf{Z}_2\end{bmatrix}}_{n^d\times n^-} = \underset{n^s\times n^+}{\begin{bmatrix}\breve{\mathbf{A}}^+\\ \tilde{\mathbf{A}}^+\end{bmatrix}}_{n^d\times n^+} d\underset{n^+\times n^-}{\mathbf{P}_{m/++,--/m}}\underset{n^-\times n^-}{\mathbf{P}_{--/m,--/m}}$$

Continuing, ξ is given by $\operatorname{tr}\left((Ad\mathbf{P}\cdot\mathbf{P})^{*}A(d\mathbf{P})^{2}+\left(A(d\mathbf{P})^{2}\right)^{*}Ad\mathbf{P}\cdot\mathbf{P}\right)$ and as $\operatorname{tr}(A^{*}B)=\operatorname{tr}(B^{*}A)$, from

$$(Ad\mathbf{P}\cdot\mathbf{P})^{*}A(d\mathbf{P})^{2}=\begin{array}{c} \\ n^{s} \\ n^{d} \\ n^{++}\end{array}\overset{\begin{array}{c}n\end{array}}{\left[\begin{array}{c}\mathbf{0}\\ \hline \begin{bmatrix}\underset{n^{-}\times n^{s}}{\mathbf{Y}_{1}^{*}} & \underset{n^{-}\times n^{d}}{\mathbf{Y}_{2}^{*}}\end{bmatrix}\\ \hline \mathbf{0}\end{array}\right]}\begin{array}{c}\\ n\end{array}\overset{\begin{array}{ccc}n^{s} & n^{-} & n^{++}\end{array}}{\left[\begin{array}{c|c|c}\mathbf{0} & \begin{bmatrix}\underset{n^{s}\times n^{-}}{\mathbf{Z}_{1}}\\ \underset{n^{d}\times n^{-}}{\mathbf{Z}_{2}}\end{bmatrix} & \mathbf{0}\end{array}\right]}$$

follows

$$\operatorname{tr}\left((Ad\mathbf{P}\cdot\mathbf{P})^{*}A(d\mathbf{P})^{2}\right)=\operatorname{tr}\left(\mathbf{Y}_{1}^{*}\mathbf{Z}_{1}\right)+\operatorname{tr}\left(\mathbf{Y}_{2}^{*}\mathbf{Z}_{2}\right)$$

Hence,

$$\begin{aligned}\xi&=\operatorname{tr}\left((Ad\mathbf{P}\cdot\mathbf{P})^{*}A(d\mathbf{P})^{2}+\left(A(d\mathbf{P})^{2}\right)^{*}Ad\mathbf{P}\cdot\mathbf{P}\right)\\&=2\operatorname{tr}\left(\mathbf{Y}_{1}^{*}\mathbf{Z}_{1}\right)+2\operatorname{tr}\left(\mathbf{Y}_{2}^{*}\mathbf{Z}_{2}\right)\end{aligned}$$

Finally, $\sigma=\operatorname{tr}\left(M(P)^{*}A(d\mathbf{P})^{2}+\left(A(d\mathbf{P})^{2}\right)^{*}M(P)\right)$, and $\delta=\operatorname{tr}(M(P)^{*}Ad\mathbf{P}\cdot\mathbf{P}+(Ad\mathbf{P}\cdot\mathbf{P})^{*}M(P))$. Using the results from above,

$$\begin{aligned}\operatorname{tr}\left(\mathbf{A}d\mathbf{P}^{2}\cdot\mathbf{M}(\mathbf{P})^{*}\right)&=\operatorname{tr}\left(\begin{array}{c}\begin{array}{ccc}\quad n^{s}\quad & \quad n^{-}\quad & \quad n^{++}\quad\end{array}\\ \overset{n}{\left[\begin{array}{c|c|c}\mathbf{0} & \begin{bmatrix}\underset{n^{s}\times n^{-}}{\mathbf{Y}_{1}}\\ \underset{n^{d}\times n^{-}}{\mathbf{Y}_{2}}\end{bmatrix} & \mathbf{0}\end{array}\right]}\end{array}\cdot\begin{array}{c}\begin{array}{c}\quad n\quad\end{array}\\ \overset{n^{s}}{\underset{n^{++}}{n^{-}}}\left[\begin{array}{c}\mathbf{0}\\ \hline \begin{bmatrix}\underset{n^{-}\times n^{s}}{\mathbf{0}} & \underset{n^{-}\times n^{d}}{\tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{*}}\end{bmatrix}\\ \hline \mathbf{0}\end{array}\right]\end{array}\right)\\&=\operatorname{tr}\left(\begin{bmatrix}\underset{n^{s}\times n^{-}}{\mathbf{Y}_{1}}\\ \underset{n^{d}\times n^{-}}{\mathbf{Y}_{2}}\end{bmatrix}\begin{bmatrix}\underset{n^{-}\times n^{s}}{\mathbf{0}} & \underset{n^{-}\times n^{d}}{\tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{*}}\end{bmatrix}\right)\\&=\operatorname{tr}\left(\underset{n^{d}\times n^{-}}{\mathbf{Y}_{2}}\ \underset{n^{-}\times n^{d}}{\tilde{\mathbf{M}}(\tilde{\mathbf{P}})^{*}}\right)\end{aligned}$$

and so

$$\begin{aligned}\sigma &= \operatorname{tr}\left(M(P)^* A(d\mathbf{P})^2 + \left(A(d\mathbf{P})^2\right)^* M(P)\right) \\ &= 2\operatorname{tr}\left(M(P)^* A(d\mathbf{P})^2\right) = 2\operatorname{tr}\left(A(d\mathbf{P})^2 M(P)^*\right) \\ &= 2\operatorname{tr}\left(\underset{n^d\times n^-}{\mathbf{Y}_2}\ \underset{n^-\times n^d}{\tilde{\mathbf{M}}(\tilde{\mathbf{P}})^*}\right)\end{aligned}$$

and by analogy

$$\delta = 2\operatorname{tr}\left(\underset{n^d\times n^-}{\mathbf{Z}_2}\ \underset{n^-\times n^d}{\tilde{\mathbf{M}}(\tilde{\mathbf{P}})^*}\right)$$

Detailed Dynare Topology-Combining Bernoulli and Newton

Let $\Delta_B \mathbf{P}_j$ be the Bernoulli increment and $\mathbf{x}_i \in [1, \infty)$ its line-search multiple, and $\mathbf{\Delta_N P}_j$ be the Newton increment with $\mathbf{y}_i \in (0, 2]$, its line search multiple.

The iteration $\mathbf{P}_{j+1} = \mathbf{S}_j \mathbf{x}_j \mathbf{\Delta_B P}_j + (1 - \mathbf{S}_j)\mathbf{y}_j \mathbf{\Delta}_j + \mathbf{P}_j$ for $\mathbf{S}_j \in [0, 1]$ is a weighted average of the line-search multiples of the Bernoulli and Newton increments. I propose determining S_j according to the same merit function that was used for $\mathbf{x}_i$, and $\mathbf{y}_i$.

$$\mathbf{t} = \min_{0\le s\le 1} \left\|\mathbf{M}(s \cdot \mathbf{x} \cdot \mathbf{\Delta_B P} + (1 - s)\mathbf{y}\mathbf{\Delta_N P} + \mathbf{P})\right\|_F^2$$

Where I have omitted "j" to minizime clutter.

$\mathbf{M}(s \cdot \mathbf{x} \cdot \mathbf{\Delta_B P} + (1 - s) \cdot \mathbf{y} \cdot \Delta_N \mathbf{P} + \mathbf{P})$ can be expressed explicitly as

$$
\begin{aligned}
&\mathbf{M}(s\cdot\mathbf{x}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)\cdot\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+\mathbf{P})\\
&\quad=\mathbf{A}\cdot(s\cdot\mathbf{x}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)\cdot\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}+\mathbf{P})^2\\
&\qquad+\mathbf{B}\cdot(s\cdot\mathbf{x}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)\cdot\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+\mathbf{P})+\mathbf{C}\\
&\quad=\mathbf{A}[\mathbf{P}^2+s\cdot\mathbf{x}\cdot\mathbf{P}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)\cdot\mathbf{y}\cdot\mathbf{P}\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+s\cdot\mathbf{x}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\cdot\mathbf{P}\cdot\mathbf{P}\\
&\qquad+s\cdot(1-s)\cdot\mathbf{x}\cdot\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}\boldsymbol{\Delta}_{\mathbf{N}}\cdot\mathbf{P}+s^2\cdot\mathbf{x}^2\cdot(\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P})^2\\
&\qquad+(1-s)\cdot\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}\cdot\mathbf{P}\\
&\qquad+(1-s)\cdot s\cdot\mathbf{y}\cdot\mathbf{x}\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)^2\cdot\mathbf{y}^2(\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P})^2]\\
&\qquad+\mathbf{B}\cdot(s\cdot\mathbf{x}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)\cdot\mathbf{y}\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+\mathbf{P})+\mathbf{C}\\
&\quad=\mathbf{A}\cdot\mathbf{P}^2+\mathbf{B}\cdot\mathbf{P}+\mathbf{C}\\
&\qquad+(1-s)\cdot\mathbf{y}\cdot(\mathbf{A}\cdot\mathbf{P}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+\mathbf{A}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}\cdot\mathbf{P}+\mathbf{B}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P})\\
&\qquad+s\cdot\mathbf{x}\cdot(\mathbf{A}\cdot\mathbf{P}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+\mathbf{A}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}\cdot\mathbf{P}+\mathbf{B}\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P})\\
&\qquad+\mathbf{A}(s\cdot\mathbf{x}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)\cdot\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P})^2
\end{aligned}
$$

Note that

$$
\begin{aligned}
\mathbf{A}(\mathbf{P}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}\cdot\mathbf{P})+\mathbf{B}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}&=-\mathbf{M}(\mathbf{P})\\
\mathbf{A}(\mathbf{P}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}\cdot\mathbf{P})+\mathbf{B}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}&=-\mathbf{M}(\mathbf{P})+\mathbf{A}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}\cdot\mathbf{P}\\
\mathbf{A}\mathbf{P}^2+\mathbf{B}\mathbf{P}+\mathbf{C}&=\mathbf{M}(\mathbf{P})
\end{aligned}
$$

So

$$
\begin{aligned}
&\mathbf{M}(s\cdot\mathbf{x}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)\cdot\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+\mathbf{P})\\
&\quad=\mathbf{M}(\mathbf{P})-(1-s)\cdot\mathbf{y}\cdot\mathbf{M}(\mathbf{P})-s\cdot\mathbf{x}\cdot\mathbf{M}(\mathbf{P})\\
&\qquad+s\cdot\mathbf{x}\cdot\mathbf{A}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}\cdot\mathbf{P}\\
&\qquad+\mathbf{A}(s\cdot\mathbf{x}\cdot\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}+(1-s)\cdot\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P})^2\\
&\quad=(1-\mathbf{y}+s(\mathbf{y}-\mathbf{x}))\cdot\mathbf{M}(\mathbf{P})+s\cdot\mathbf{x}\cdot\mathbf{A}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}\cdot\mathbf{P}\\
&\qquad+\mathbf{A}\left(\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}+s\cdot\left(\mathbf{x}\cdot\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}-\mathbf{y}\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}\right)\right)^2\\
&\quad=\underbrace{(1-y)\mathbf{M}(\mathbf{P})+\mathbf{A}\tilde{\boldsymbol{\Delta}}_{\mathbf{N}}\mathbf{P}\cdot\mathbf{P}^2}_{a}\\
&\qquad+s\cdot\underbrace{((\mathbf{y}-\mathbf{x})\cdot\mathbf{M}(\mathbf{P})+\mathbf{A}\cdot\tilde{\boldsymbol{\Delta}}_{\mathbf{B}}\mathbf{P}\cdot\mathbf{P}+\mathbf{A}\cdot\tilde{\boldsymbol{\Delta}}_{\mathbf{N}}\mathbf{P}\cdot\tilde{\boldsymbol{\Delta}}\mathbf{P}+\mathbf{A}\cdot\tilde{\boldsymbol{\Delta}}\mathbf{P}\cdot\tilde{\boldsymbol{\Delta}}_{\mathbf{N}}\mathbf{P})}_{\mathbf{b}}\\
&\qquad+s^2\cdot\underbrace{\mathbf{A}\cdot(\tilde{\boldsymbol{\Delta}}P)^2}_{\mathbf{c}}=\mathbf{a}+s\cdot\mathbf{b}+s^2\mathbf{c}
\end{aligned}
$$

Where $\tilde{\boldsymbol{\Delta}}_{\mathbf{N}}\mathbf{P}=\mathbf{y}\cdot\boldsymbol{\Delta}_{\mathbf{N}}\mathbf{P}$, $\tilde{\boldsymbol{\Delta}}_{\mathbf{B}}\mathbf{P}=\mathbf{x}\boldsymbol{\Delta}_{\mathbf{B}}\mathbf{P}$, $\tilde{\boldsymbol{\Delta}}\mathbf{P}=\tilde{\boldsymbol{\Delta}}_{\mathbf{B}}\mathbf{P}-\tilde{\boldsymbol{\Delta}}_{\mathbf{N}}\mathbf{P}$. So

$$\begin{aligned}\|\mathbf{M}(\cdot)\|_F^2 &= \|\mathbf{a}\|_F^2 + s(\mathrm{tr}(\mathbf{a}^* \cdot \mathbf{b}) + \mathrm{tr}(\mathbf{b}^* \cdot \mathbf{a})) + s^2 \cdot \left(\|\mathbf{b}\|_F^2 + \mathrm{tr}(\mathbf{a}^*\mathbf{c}) + \mathrm{tr}(\mathbf{c}^*\mathbf{a})\right) \\ &\quad + s^3(\mathrm{tr}(\mathbf{b}^* \cdot \mathbf{c}) + \mathrm{tr}(\mathbf{c}^* \cdot \mathbf{b})) + s^4 \cdot \|\mathbf{c}\|_F^2 \\ &= \|\mathbf{a}\|_F^2 + s \cdot 2 \cdot \mathrm{tr}(\mathbf{a}^*\mathbf{b}) + s^2 \cdot \left(\|\mathbf{b}\|_F^2 + 2\mathrm{tr}(\mathbf{a}^*\mathbf{c})\right) + s^3 \cdot 2\mathrm{tr}(\mathbf{b}^* \cdot \mathbf{c}) + s^4 \cdot \|\mathbf{c}\|_F^2 \\ &\equiv \mathbf{t}(s) \\ \mathbf{t}'(s) &= 2 \cdot \mathrm{tr}(\mathbf{a}^*\mathbf{b}) + 2 \cdot (\|\mathbf{b}\|_F^2 + 2 \cdot \mathrm{tr}(\mathbf{a}^*\mathbf{c}) \cdot s + 6 \cdot \mathrm{tr}(\mathbf{b}^* \cdot \mathbf{c}) \cdot s^2 + 4 \cdot \|\mathbf{c}\|_F^2 \cdot s^3\end{aligned}$$

Unsing the Dynare typology above

$$\underset{n \times n^{s}+n^{-}+n^{++}}{\mathbf{a}} = n \left[\begin{array}{c|c|c} \underset{n \times n^{s}}{\mathbf{0}} & \underset{n \times n^{-}}{\tilde{\mathbf{a}}_1 + \tilde{\mathbf{a}}_2} & \underset{n \times n^{++}}{\mathbf{0}} \end{array} \right]$$

$$\underset{n \times n^{-}}{\tilde{\mathbf{a}}_1} = \begin{matrix} \\ n^s \\ n^d \end{matrix} \begin{matrix} \overset{n^-}{} \\ \left[\begin{array}{c} \mathbf{0} \\ (\mathbf{1}-\mathbf{y})\mathbf{x}_2 \end{array} \right] \end{matrix}, \quad \underset{n^d \times n^-}{\mathbf{x}_2} = \underset{n^d \times n^+}{\tilde{\mathbf{A}}^+} + \underset{n^+ \times n}{\mathbf{P}_{m/++,\bullet}} \, \underset{n \times n^-}{\mathbf{P}_{\bullet,--/m}} + \underset{n^d \times n^d}{\tilde{\mathbf{A}}^\circ} \, \underset{n^d \times n^-}{\mathbf{P}_{d,--/m}} + \underset{n^d \times n^-}{\tilde{\mathbf{A}}^-}$$

$$\underset{n \times n^{-}}{\tilde{\mathbf{a}}_2} = \mathbf{y}^2 \cdot \begin{matrix} \\ n^s \\ n^d \end{matrix} \begin{matrix} \overset{n^+}{} \\ \left[\begin{array}{c} \tilde{\mathbf{A}}^+ \\ \tilde{\mathbf{A}}^+ \end{array} \right] \end{matrix} (\mathbf{d_N} \underset{n^+ \times n}{\mathbf{P}_{m/++,\bullet}} \, \mathbf{d_N} \underset{n \times n^-}{\mathbf{P}_{\bullet,--/m}})$$

$$\underset{n \times n^{s}+n^{-}+n^{++}}{\mathbf{b}} = n \left[\begin{array}{c|c|c} \underset{n \times n^{s}}{\mathbf{0}} & \underset{n \times n^{-}}{\tilde{\mathbf{b}}_1 + \tilde{\mathbf{b}}_2} & \underset{n \times n^{++}}{\mathbf{0}} \end{array} \right]$$

$$\underset{n \times n^{-}}{\tilde{\mathbf{b}}_1} = \begin{matrix} \\ n^s \\ n^d \end{matrix} \begin{matrix} \overset{n^-}{} \\ \left[\begin{array}{c} \mathbf{0} \\ (y-x)x_2 \end{array} \right] \end{matrix}$$

$$\underset{n \times n^{-}}{\tilde{\mathbf{b}}_2} = \begin{matrix} \\ n_s \\ n_d \end{matrix} \begin{matrix} \overset{n_+}{} \\ \left[\begin{array}{c} \check{A}^+ \\ \tilde{A}^+ \end{array} \right] \end{matrix} \cdot \Big(\mathbf{x}\mathbf{d_B} \underset{n^+ \times n}{\mathbf{P}_{m/++,\bullet}} \cdot \underset{n \times n^-}{\mathbf{P}_{\bullet,--m}} + \mathbf{y}\mathbf{d_N} \underset{n^+ \times n}{\mathbf{P}_{m/++,\bullet}} (\mathbf{x}\mathbf{d_B} \underset{n \times n^-}{\mathbf{P}_{\bullet,--/m}} - \mathbf{y}\mathbf{d_N} \underset{n \times n^-}{\mathbf{P}_{\bullet,--/m}}) + \mathbf{y} \, (\mathbf{x}\mathbf{d_B} \underset{n^+ \times n}{\mathbf{P}_{m/++,\bullet}} - \mathbf{y}\mathbf{d_N} \underset{n^+ \times n}{\mathbf{P}_{m/++,\bullet}}) \mathbf{d_N} \underset{n \times n^-}{\mathbf{P}_{\bullet,--/m}} \Big)$$

$$\underset{n \times n^{s}+n^{-}+n^{++}}{\mathbf{c}} = n \left[\begin{array}{c|c|c} \overset{n^s}{\mathbf{0}} & \begin{matrix} \overset{n^+}{} \\ \left[\begin{array}{c} \check{\mathbf{A}}^+ \\ \tilde{\mathbf{A}}^+ \end{array} \right] \end{matrix} (\mathbf{x} \, \mathbf{d_B} \underset{n^+ \times n}{\mathbf{P}_{m/++,\bullet}} - \mathbf{y} \, \mathbf{d_N} \underset{n^+ \times n}{\mathbf{P}_{m/++,\bullet}}) (\mathbf{x} \, \mathbf{d_B} \underset{n^- \times n}{\mathbf{P}_{\bullet,--/m}} - \mathbf{y} \, d_N \underset{n^- \times n}{\mathbf{P}_{\bullet,--/m}}) & \overset{n_{++}}{\mathbf{0}} \end{array} \right]$$

Proof of Theorem 1

Following Higham (2008, Section 4.9.4.), local stability requires the Fréchet derivative of F at P to have bounded powers, which holds if its spectral radius is less than one. At P, (72) is

$$\operatorname{vec}\left(\mathscr{D}_P F(\Delta P)\right) = \left(\left[(A\,P+B\,)^{-1}C\right]' \otimes \left[(A\,P+B\,)^{-1}A\right]\right)\operatorname{vec}\left(\Delta P\right) \tag{A2}$$

$$= \left(P' \otimes \left[(A\,P+B\,)^{-1}A\right]\right)\operatorname{vec}\left(\Delta P\right) \tag{A3}$$

Hence the spectral radius is equal to the eigenvalue of $P' \otimes \left[(A\,P+B\,)^{-1}A\right]$ with the largest magnitude. As the eigenvalues of the Kronecker product of two matrices is the set of all crossproducts of the eigenvalues of the respective matrices, the spectral radius is the product of the eigenvalues of P and $(A\,P+B\,)^{-1}A$ with the largest magnitudes. Following Lan and Meyer-Gohde (2014, Corollary 4.2.), (17) can be factored as

$$M(\lambda) \equiv A\,\lambda^2 + B\,\lambda + C = (A\lambda + A\,P + B)(\lambda - P) \tag{A4}$$

Hence, the latent roots associated with $A\lambda + A\,P + B$ are those roots of $M(\lambda)$ not contained in set of eigenvalues of P. By assumption, the eigenvalues of P are inside the closed unit circle and the roots associated with $A\lambda + A\,P + B$ are outside the open unit circle. By inspection, the eigenvalues of $(A\,P+B\,)^{-1}A$ are the inverses of the roots associated with $A\lambda + A\,P + B$ and, therefore, are all inside the open unit circle. Hence the product of the largest magnitude eigenvalue of P and that of $(A\,P+B\,)^{-1}A$ is less than one in absolute value. That is, the spectral radius of the Fréchet derivative of F at P, the unique bounded solution, is less than one, completing the proof.

Acknowledgements I am grateful to Johanna Saecker, Pablo Winant, and participants of the 29th International Conference on Computing in Economics and Finance (2023 CEF) for useful comments and suggestions and to Elena Schlipphack and Maximilian Thomin for invaluable research assistance. The code to replicate the analysis can be found at https://github.com/AlexMeyer-Gohde/Linear-DSGE-with-Bernoulli. Any and all errors are entirely my own.

Funding Open Access funding enabled and organized by Projekt DEAL. This research was supported by the DFG through grant no. 465469938 "Numerical diagnostics and improvements for the solution of linear dynamic macroeconomic models".

Declarations

Competing Interests The authors have not disclosed any competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission

directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Adjemian, S., Bastani, H., Juillard, M., Mihoubi, F., Perendia, G., Ratto, M., & Villemot, S. (2011). Dynare: Reference manual, Version 4. Dynare Working Papers 1. CEPREMAP.

Al-Sadoon, M. M. (2018). The linear systems approach to linear rational expectations models. *Econometric Theory, 34*(3), 628–658. https://doi.org/10.1017/S0266466617000160

Al-Sadoon, M. M. (2020). *Regularized solutions to linear rational expectations models*. Discussion paper.

Anderson, G. S. (2010). A reliable and computationally efficient algorithm for imposing the saddle point property in dynamic models. *Journal of Economic Dynamics and Control, 34*(3), 472–489. https://doi.org/10.1016/S1474-6670(17)40506-4

Anderson, G. S., Levin, A., & Swanson, E. (2006). Higher-order perturbation solutions to dynamic discrete-time rational expectations models. Discussion Paper 2006-01. Federal Reserve Bank of San Francisco Working Paper Series.

Anderson, G. S., & Moore, G. (1985). A linear algebraic procedure for solving linear perfect foresight models. *Economics Letters, 17*(3), 247–252. https://doi.org/10.1016/0165-1765(85)90211-3

Aruoba, S. B., Fernández-Villaverde, J., & Rubio-Ramírez, J. F. (2006). Comparing solution methods for dynamic equilibrium economies. *Journal of Economic Dynamics and Control, 30*(12), 2477–2508. https://doi.org/10.1016/j.jedc.2005.07.008

Bai, Z.-Z., & Gao, Y.-H. (2007). Modified Bernoulli iteration methods for quadratic matrix equation. *Journal of Computational Mathematics, 25*(5), 498–511.

Binder, M., & Pesaran, M. H. (1997). Multivariate linear rational expectations models: Characterization of the nature of the solutions and their fully recursive computation. *Econometric Theory, 13*(6), 877–88. https://doi.org/10.1017/S0266466600006307

Binder, M., & Pesaran, M. H. (1999). Multivariate rational expectations models and macroeconometric modeling: A review and some new results. Macroeconomics. In M. H. Pesaran & M. Wickens (Eds.), *Handbook of applied econometrics* (3rd ed., Vol. 1, pp. 111–155). London: Wiley. https://doi.org/10.1111/b.9780631215585.1999.00004.x

Blanchard, O. J., & Kahn, C. M. (1980). The solution of linear difference models under rational expectations. *Econometrica, 48*(5), 1305–1311. https://doi.org/10.2307/1912186

Caldara, D., Fernández-Villaverde, J., Rubio-Ramírez, J., & Yao, W. (2012). Computing DSGE models with recursive preferences and stochastic volatility. *Review of Economic Dynamics, 15*(2), 188–206. https://doi.org/10.1016/j.red.2011.10.001

Chen, X. S., & Lv, P. (2018). On estimating the separation between (A, B) and (C, D) associated with the generalized Sylvester equation $AXD - BXC = E$. *Journal of Computational and Applied Mathematics, 330*, 128–140. https://doi.org/10.1016/j.cam.2017.07.025

Dennis, J. E., Jr., Traub, J. F., & Weber, R. P. (1976). The algebraic theory of matrix polynomials. *SIAM Journal on Numerical Analysis, 13*(6), 831–845. https://doi.org/10.1137/071306

Dennis, J. E., Jr., Traub, J. F., & Weber, R. P. (1978). Algorithms for solvents of matrix polynomials. *SIAM Journal on Numerical Analysis, 15*(3), 523–533. https://doi.org/10.1137/0715034

Gantmacher, F. R. (1959). *The theory of matrices* (Vol. I, II). Chelsea Publishing Company.

Golub, G. H., & van Loan, C. F. (2013). *Matrix computations* (4th ed.). The Johns Hopkins University Press.

Hammarling, S., Munro, C. J., & Tisseur, F. (2013). An algorithm for the complete solution of quadratic eigenvalue problems. *ACM Transactions on Mathematical Software, 39*(3), 18:1-18:19. https://doi.org/10.1145/2450153.2450156

Higham, N. J. (2002). *Accuracy and stability of numerical algorithms* (2nd ed.). Society for Industrial and Applied Mathematics.

Higham, N. J. (2008). *Functions of matrices: Theory and computation*. Society for Industrial and Applied Mathematics.

Higham, N. J., & Kim, H.-M. (2000). Numerical analysis of a quadratic matrix equation. *IMA Journal of Numerical Analysis, 20*, 499–519. https://doi.org/10.1093/imanum/20.4.499

Higham, N. J., & Kim, H.-M. (2001). Solving a quadratic matrix equation by Newton's method with exact line searches. *SIAM Journal on Matrix Analysis and Applications, 23*(2), 499–519. https://doi.org/10.1137/S0895479899350976

Horn, R. A., & Johnson, C. R. C. R. (2012). *Matrix analysis* (2nd ed.). Cambridge University Press.

Huber, J., Meyer-Gohde, A., & Saecker, J. (2023). Solving linear DSGE models with structure-preserving doubling methods. IMFS Working Paper Series 195. Goethe University Frankfurt, Institute for Monetary and Financial Stability (IMFS).

Judd, K. L. (1992). Projection methods for solving aggregate growth models. *Journal of Economic Theory, 58*(2), 410–452. https://doi.org/10.1016/0022-0531(92)90061-L

Judd, K. L. (1998). *Numerical methods in economics*. MIT Press.

Klein, P. (2000). Using the generalized Schur form to solve a multivariate linear rational expectations model. *Journal of Economic Dynamics and Control, 24*(10), 1405–1423. https://doi.org/10.1016/S0165-1889(99)00045-7

Kågström, B. (1994). A perturbation analysis of the generalized Sylvester equation $(AR - LB, DR - LE) = (C, F)$. *SIAM Journal on Matrix Analysis and Applications, 15*(4), 1045–1060. https://doi.org/10.1137/S0895479893246212

Kågström, B., & Poromaa, P. (1996). LAPACK-style algorithms and software for solving the generalized Sylvester equation and estimating the separation between regular matrix pairs. *ACM Transactions on Mathematical Software (TOMS), 22*(1), 78–103. https://doi.org/10.1145/225545.225552

Lan, H., & Meyer-Gohde, A. (2014). Solvability of perturbation solutions in DSGE models. *Journal of Economic Dynamics and Control, 45*, 366–388. https://doi.org/10.1016/j.jedc.2014.06.005

Lubik, T. A., & Schorfheide, F. (2003). Computing sunspot equilibria in linear rational expectations models. *Journal of Economic Dynamics and Control, 28*(2), 273–285. https://doi.org/10.1016/S0165-1889(02)00153-7

Meyer-Gohde, A. (2023). Numerical stability analysis of linear DSGE model—Backward errors, forward errors and condition numbers. IMFS Working Paper Series 193. Goethe University Frankfurt, Institute for Monetary and Financial Stability (IMFS).

Meyer-Gohde, A., & Neuhoff, D. (2015). Solving and estimating linearized DSGE models with VARMA shock processes and filtered data. *Economics Letters, 133*, 89–91. https://doi.org/10.1016/j.econlet.2015.05.024

Meyer-Gohde, A., & Saecker, J. (2024). Solving linear DSGE models with Newton methods. *Economic Modelling, 133*, 106670. https://doi.org/10.1016/j.econmod.2024.106670

Moler, C. B., & Stewart, G. W. (1973). An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis, 10*(2), 241–256. https://doi.org/10.1137/0710024

Polyak, B. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics, 4*(5), 1–17. https://doi.org/10.1016/0041-5553(64)90137-5

Rendahl, P. (2017). *Linear time iteration*. Discussion paper: IHS economics series.

Sims, C. A. (2001). Solving linear rational expectations models. *Computational Economics, 20*(1–2), 1–20. https://doi.org/10.1023/A:1020517101123

Smets, F., & Wouters, R. (2007). Shocks and frictions in US business cycles: A bayesian DSGE approach. *The American Economic Review, 97*(3), 586–606. https://doi.org/10.1257/aer.97.3.586

Stewart, G. W. (1971). Error bounds for approximate invariant subspaces of closed linear operators. *SIAM Journal on Numerical Analysis, 8*(4), 796–808. https://doi.org/10.1137/0708073

Taylor, J. B., & Uhlig, H. (1990). Solving nonlinear stochastic growth models: A comparison of alternative solution methods. *Journal of Business & Economic Statistics, 8*(1), 1–17. https://doi.org/10.1080/07350015.1990.10509766

Tisseur, F., & Meerbergen, K. (2001). The quadratic eigenvalue problem. *SIAM Review, 43*(2), 235–286. https://doi.org/10.1137/S0036144500381988

Uhlig, H. (1999). A toolkit for analysing nonlinear dynamic stochastic models easily. In R. Marimon & A. Scott (Eds.), *Computational methods for the study of dynamic economies* (Vol. 3, pp. 30–61). Oxford University Press.

Villemot, S. (2011). Solving rational expectations models at first order: What dynare does. Dynare Working Papers 2. CEPREMAP.

Wieland, V., Afanasyeva, E., Kuete, M., & Yoo, J. (2016). New methods for macro-financial model comparison and policy analysis. In J. B. Taylor & H. Uhlig (Eds.), *Handbook of macroeconomics handbook of macroeconomics* (Vol. 2, pp. 1241–1319). Elsevier. https://doi.org/10.1016/bs.hesmac.2016.04.004

Wieland, V., Cwik, T., Müller, G. J., Schmidt, S., & Wolters, M. (2012). A new comparative approach to macroeconomic modeling and policy analysis. *Journal of Economic Behavior & Organization, 83*(3), 523–541. https://doi.org/10.1016/j.jebo.2012.01.006

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.