SOLVING LINEAR DSGE MODELS WITH STRUCTURE PRESERVING DOUBLING METHODS

JOHANNES HUBER, ALEXANDER MEYER-GOHDE, AND JOHANNA SAECKER

ABSTRACT. This paper applies Structure-Preserving Doubling Algorithms (SDAs) to solve the matrix quadratic that underlies linear DSGE models. We present and compare two SDAs to other competing methods – the QZ method, a Newton algorithm, and an iterative Bernoulli approach – as well as linking them to the cyclic and logarithmic reduction algorithms included in Dynare. Our evaluation, conducted across 142 models from the Macroeconomic Model Data Base (MMB) and multiple parameterizations of the Smets and Wouters (2007) model, demonstrates that SDAs generally provide more accurate solutions in less time than QZ. We also establish their theoretical convergence properties and robustness to initialization issues. The SDAs perform particularly well in refining solutions provided by other methods and for large models.

JEL classification codes: C61, C63, E17

Keywords: Numerical accuracy; DSGE; Solution methods

1. INTRODUCTION

Solving a linear DSGE model is often the first step in analyzing a structural macroeconomic question. A key computational challenge is solving the associated matrix quadratic equation efficiently and accurately. Existing methods, such as the generalized Schur (QZ) decomposition (Moler and Stewart, 1973) are widely used but can struggle with

(Huber) UNIVERSITY OF REGENSBURG, UNIVERSITÄTSSTRASSE 31, 93053 REGENSBURG, GERMANY

(Meyer-Gohde, Saecker) GOETHE-UNIVERSITÄT FRANKFURT AND INSTITUTE FOR MONETARY AND FINANCIAL STABILITY (IMFS), THEODOR-W.-ADORNO-PLATZ 3, 60629 FRANKFURT AM MAIN, GERMANY

E-mail addresses: johannes1.huber@ur.de, meyer-gohde@econ.uni-frankfurt.de, saecker@hof.uni-frankfurt.de.

Date: May 19, 2025.

We are grateful to participants of the 30th International Conference on Computing in Economics and Finance (2024 CEF) and the Annual Conference of the International Association for Applied Econometrics (IAAE 2024). The code to replicate the analysis can be found at https://github.com/AlexMeyer-Gohde/Linear-DSGE-with-Doubling. This research was supported by the DFG through grant nr. 465469938 “Numerical diagnostics and improvements for the solution of linear dynamic macroeconomic models” and grant nr. 465135565 “Models of Imperfect Rationality and Redistribution in the Context of Retirement”.

computational efficiency and scalability for large-scale models or when iteratively exploring multiple parameterizations. We demonstrate the potential of Structure-Preserving Doubling Algorithm (SDA)[1] to overcome these challenges by applying them to two practical contexts: first, in Smets and Wouters's (2007) New Keynesian model, where we evaluate performance across varying monetary policy rules; and second, in a comprehensive comparison using 142 models from the Macroeconomic Model Data Base (MMB).[2] In both cases, SDAs outperform conventional methods in terms of speed and accuracy, particularly when refining solutions or solving large models.

Doubling algorithms are certainly not unknown to economists (see, e.g., Hansen and Sargent, 2014, Chapter 3.6). Anderson and Moore (1979) consider doubling algorithms to solve the Riccati equations occuring in optimal linear filtering exercises. Building on this, Anderson et al. (1996, Section 10, p. 224) use doubling algorithms to receive a conditional log-likelihood function for linear state space models (see also Harvey, 1990, Chapter 3, p. 119,129). Furthermore, McGrattan (1990) as well applies doubling algorithms to the Riccati and Sylvester equations in the unknown matrices of the linear solution to LQ optimal control problems in economics. As our class of models as defined by Dynare (Adjemian et al., 2011) (henceforth Dynare) includes and expands on this class of models, our approach here can be seen as an extension, specifically to Anderson et al.'s (1996) work.[3] More closely related is the application to Riccati equations (see Poloni (2020) for an accessible introduction to doubling algorithms for Riccati equations), a link between the solution of Riccati equations and the matrix quadratic we solve is noted explicitly by Higham and Kim (2000) and Bini et al. (2008) for example—both are quadratic equations in a matrix unknown, but with different structures. Chiang et al. (2009), however, present explicit results for matrix polynomials with doubling methods—specifically structure preserving doubling methods, see Huang et al. (2018)—and connect these algorithms to reduction algorithms (Latouche and Ramaswami's (1993) logarithmic and Bini and Meini's

[1]See Huang et al. (2018) for an overview on doubling methods for nonlinear problems.

[2]The MMB is a model comparison initiative at the Institute for Monetary and Financial Stability (IMFS), see `http://www.macromodelbase.com` and Wieland et al. (2012, 2016).

[3]Note that a doubling algorithm is also used in Dynare in the algorithm `disclyap_fast.m` for solving Lyapunov equations in the variance-covariance matrices of linear state space models.

(1996) cyclic) that are algorithms available in Dynare,[4] though arguably undocumented.[5] Beyond relating these algorithms using unified notation, we provide iterative capabilities (i.e., updating or refining some initialized solution) using Bini and Meini (2023) that allow us to operate on a starting value for a solution to the matrix quadratic for the first SDA, First Standard Form (SF1). We show, however, that while this may reduce the computation time, the asymptotic solution of the second SDA, Second Standard Form (SF2), is unaffected.

We engage in a number of experiments to compare the algorithms to QZ-based methods,[6] Dynare's implementation of Latouche and Ramaswami's (1993) logarithmic reduction algorithm and Bini and Meini's (1996) cyclic reduction algorithms, Meyer-Gohde and Saecker's (2024) Newton algorithms, and Meyer-Gohde's (2024) Bernoulli methods following the latter's experiments to ensure comparability. We begin by comparing the methods in the Smets and Wouters (2007) model of the US economy - at the posterior mode, a numerically problematic calibration, and in solving for different parameterizations of the Taylor rule. In the latter, we move through a grid of different values of the reaction of monetary policy to inflation and output. Whereas the QZ and reduction methods have to recalculate the entire solution at each new parameter combination, the iterative implementations of the SDA like Newton and Bernoulli algorithms can initialize using the solution from the previous, nearby parameterization. As the parameterizations get closer together, the advantage in terms of computation time increases significantly, while the accuracy (measured by Meyer-Gohde's (2023) practical forward error bounds) remains unaffected. We show that one of the doubling algorithms profits from this effect, while the other does not - this is consistent with our theoretical results that this particular doubling algorithm converges to the same solution regardless of its initialization.

We then compare the different methods using the models in the MMB, both initializing with a zero matrix (as implied by Latouche and Ramaswami's (1993) logarithmic reduction and Bini and Meini's (1996) cyclic reduction algorithms) and as solution refinement for iterative implementations (i.e., initializing the iterative methods with the QZ solution).

[4]See also Poloni (2020) for the link to Riccati equations in quasi birth death models, with, e.g., nonnegative as components of a transition probability matrix imposing stronger assumptions than in DSGE.

[5]Adjemian et al. (2011, p. 55) does note the options `dr=cycle_reduction` and `dr=logarithmic_reduction` to use these algorithms and even claims the former "is faster than the default one for large scale models", but neither this claim nor the algorithms themselves are laid out or analyzed rigorously in a DSGE context.

[6]We use Dynare's implementation of the QZ method, documented in Villemot (2011), for comparison.

We find that the reduction and doubling methods provide useable alternatives to the standard QZ.[7] The cyclic reduction algorithm suffers from unreliable convergence to the stable solution and the doubling algorithms perform more reliably than both the reduction methods, providing higher accuracy at frequently lower cost than alternatives including QZ. While each of the two doubling algorithms have their relative advantages as unconditional methods, one is particularly successful at solution refinement. This algorithm, consistent with the grid experiment in the Smets and Wouters (2007) model, reliably provides large increases in accuracy at low additional computation costs - exactly what would be demanded of such an algorithm.

2. Problem Statement

Standard numerical solution packages for DSGE models such as Dynare analyze models that can generally be collected in the following nonlinear functional equation

$$0 = E_t[f(y_{t+1}, y_t, y_{t-1}, \varepsilon_t)] \tag{1}$$

The function $f : \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_e} \to \mathbb{R}^{n_y}$ comprises the conditions (first order conditions, resource constraints, market clearing, etc.) that characterize the model; the endogenous variables $y_t \in \mathbb{R}^{n_y}$; and the exogenous shocks $\varepsilon_t \in \mathbb{R}^{n_e}$, where $n_y, n_e \in \mathbb{N}$ and ε_t has a known mean zero distribution. The solution to the model (1) is the unknown function

$$y_t = y(y_{t-1}, \varepsilon_t), \quad y : \mathbb{R}^{n_y + n_e} \to \mathbb{R}^{n_y} \tag{2}$$

that maps states, y_{t-1} and ε_t, into endogenous variables, y_t. A closed form for (2) is generally not available and we must approximate. One point in the solution, the deterministic steady state, $\overline{y} \in \mathbb{R}^{n_y}$ a vector such $\overline{y} = y(\overline{y}, 0)$ and $0 = f(\overline{y}, \overline{y}, \overline{y}, 0)$ can often be solved for and this provides a point around which local solutions can be expanded.

The linear, or first-order, approximation of (1) at the steady state gives

$$0 = AE_t[y_{t+1}] + By_t + Cy_{t-1} + D\varepsilon_t \tag{3}$$

where A, B, C, and D are the derivatives of f in (1) evaluated at the steady state and the y's in (3) now, reusing notation, are the (log) deviations of the endogenous variables from their steady states, $\overline{y}$. The solution to (3) is a linear solution of the form (2)

$$y_t = P\, y_{t-1} + Q\, \varepsilon_t \tag{4}$$

[7]See Heiberger et al. (2017) and Meyer-Gohde (2023) for accuracy measures and both practical and theoretical evidence of potential instabilities of the QZ algorithm in DSGE applications.

which expresses y_t as a function of its own past, y_{t-1}, and the shocks, ε_t.

Substituting (4) into (3) and recognizing that $E_t[\varepsilon_{t+1}] = 0$ yields the following

$$0 = AP^2 + BP + C, \quad 0 = (AP + B)Q + D \tag{5}$$

The former is quadratic with potentially multiple solutions—generally a P with all its eigenvalues inside the open unit circle is sought. As Lan and Meyer-Gohde (2014) prove, the latter can be uniquely solved for Q if such a P can be found, our focus will be the former, the unilateral quadratic matrix equations (UQME). In the following we show how to solve for P in (5) using structure-preserving doubling algorithms.

3. Doubling Methods for Linear DSGE models

In this section, we outline two structure-preserving doubling algorithms (SDAs), well known from the applied mathematics literature (e.g. Huang et al., 2018), and apply them to solve linear DSGE models. While the algorithms themselves are not new, we contribute by showing how to recast the UQME (5) into the standard forms—First Standard Form (SF1) and Second Standard Form (SF2)—required to apply these methods. We establish a link between the standard conditions used in the DSGE literature to ensure existence and uniqueness of a solution and the convergence requirements of SDAs, as discussed in Section 4. Finally, we present extensions of both algorithms to incorporate an initial guess to enable them to act as refinement tools in iterative applications.[8]

3.1. Matrix Quadratics, Pencils, and Doubling

We begin by expressing the UQME in (5) as a subspace problem via the first companion linearization of the matrix quadratic problem (Hammarling et al., 2013)

$$\mathscr{A}\mathscr{X} = \mathscr{B}\mathscr{X}\mathscr{M} \tag{6}$$

with

$$\mathscr{X} = \begin{pmatrix} I \\ P \end{pmatrix} \qquad \mathscr{A} = \begin{pmatrix} 0 & I \\ C & B \end{pmatrix}, \qquad \mathscr{B} = \begin{pmatrix} I & 0 \\ 0 & -A \end{pmatrix}, \qquad \mathscr{M} = P$$

Clearly, any P satisfying (5) is a solution of (6). Further, the eigenvalues of $\mathscr{M}$ are a subset of the generalized eigenvalues of the matrix pencil $\mathscr{A} - \lambda\mathscr{B}$, i.e., $\operatorname{eig}(\mathscr{M}) \subset \operatorname{eig}(\mathscr{A}, \mathscr{B})$.

[8] The online appendix contains a scalar presentation of the doubling approach to a geometric series for readers looking for more intuition for how a doubling approach can be applied to subspace problems.

We make the standard assumption in the literature for the existence of a unique stable solvent P, i.e., Blanchard and Kahn's (1980) celebrated rank and order conditions.

Assumption 1 (Blanchard and Kahn's (1980) Rank and Order). *There exist*

(1) $2n_y$ *latent roots of* $A\lambda^2+B\lambda+C$ *(*$n_y+\operatorname{rank}(A)$ *finite* $\lambda\in\mathbb{C}:\det(A\lambda^2+B\lambda+C)=0$ *and* $n_y-\operatorname{rank}(A)$ *"infinite"* λ*) with* n_y *inside or on and* n_y *outside the unit circle.*

(2) a $P\in\mathbb{R}^{n_y\times n_y}$ *with* $\rho(P)\le 1$ *that satisfies the primary matrix quadratic*

$$0=AP^2+BP+C$$

where $\rho(X)$ *denotes the spectral radius of a matrix* X.

The quadratic convergence of our SDAs requires the existence of a unique solvent to the reverse problem, which follows directly from assumption 1.[9]

Corollary 1 (Solvent of the Dual Matrix Equation). *Let assumption 1 hold. There exists a* $P_d\in\mathbb{R}^{n_y\times n_y}$ *with* $\rho(P_d)<1$ *that satisfies the dual matrix quadratic*

$$0=CP_d^2+BP_d+A.$$

The problem in (6) is an eigenvalue problem and can thus be solved numerically using the QZ or generalized Schur decompostion of Moler and Stewart (1973). In the online appendix, we derive the solution with QZ by working directly with the linear algebraic problem. This should link the more familiar QZ with the doubling algorithms we will subsequently present more clearly to readers looking for additional intuition.

We will approach the matrix pencil problem here via a doubling approach. In detail we will try to generate sequences $\{\mathscr{A}_k\}_{k=1}^N$ and $\{\mathscr{B}_k\}_{k=1}^N$ such that

$$\mathscr{A}_k\mathscr{X}=\mathscr{B}_k\mathscr{X}\mathscr{M}^{2^k} \tag{7}$$

squaring the matrix $\mathscr{M}$ in each iteration. At the same time, we will seek to preserve the structure of the pencil, e.g., a special block structure of $\mathscr{A}_k$ and $\mathscr{B}_k$. This will enable us to express the algorithm in terms of submatrices of $\mathscr{A}_k$ and $\mathscr{B}_k$ instead of the matrices in their entirety and makes sure that we can repeat the doubling step in the next iteration.

Following Guo et al. (2006), a transformation $\widehat{\mathscr{A}}-\lambda\widehat{\mathscr{B}}$ of a pencil $\mathscr{A}-\lambda\mathscr{B}$ is called a doubling transform if

$$\widehat{\mathscr{A}}=\overline{\mathscr{A}}\mathscr{A},\quad \widehat{\mathscr{B}}=\overline{\mathscr{B}}\mathscr{B} \tag{8}$$

[9] For more details and the proof of the following corollary, see the online appendix.

for $\overline{\mathscr{A}}$ and $\overline{\mathscr{B}}$ that satisfy

$$\operatorname{rank}\left(\begin{bmatrix}\overline{\mathscr{A}} & \overline{\mathscr{B}}\end{bmatrix}\right) = 2n_y, \quad \begin{bmatrix}\overline{\mathscr{A}} & \overline{\mathscr{B}}\end{bmatrix}\begin{bmatrix}\mathscr{B} \\ -\mathscr{A}\end{bmatrix} = 0 \tag{9}$$

That is, we will be transforming the problem here as in QZ, seeking a structure that is amenable to doubling instead of the upper triangularity sought there.

Such a doubling transformation is eigenspace preserving and eigenvalue squaring ("doubling") following Guo et al. (2006, Theorem 2.1) or Huang et al. (2018, Theorem 3.1) that we repeat here adapted to our problem

Theorem 1 (Doubling Pencil). *Suppose $\widehat{\mathscr{A}} - \lambda\widehat{\mathscr{B}}$ is a doubling transformation of the pencil $\mathscr{A} - \lambda\mathscr{B}$. Then, as $\mathscr{A}\mathscr{X} = \mathscr{B}\mathscr{X}\mathscr{M}$ it follows from (6) that*

$$\widehat{\mathscr{A}}\mathscr{X} = \widehat{\mathscr{B}}\mathscr{X}\mathscr{M}^2 \tag{10}$$

Proof. See Huang et al. (2018, Theorem 3.1) □

Following this theorem, we obtain a doubling algorithm for (6) by iterating on

$$\underbrace{\widehat{\mathscr{A}_k}}_{\mathscr{A}_{k+1}} = \overline{\mathscr{A}}_k\mathscr{A}_k, \quad \underbrace{\widehat{\mathscr{B}_k}}_{\mathscr{B}_{k+1}} = \overline{\mathscr{B}}_k\mathscr{B}_k \tag{11}$$

initializing with $\mathscr{A}$ and $\mathscr{B}$, as $\mathscr{A}\mathscr{X} = \mathscr{B}\mathscr{X}\mathscr{M}$ then gives

$$\mathscr{A}_1\mathscr{X} = \mathscr{B}_1\mathscr{X}\mathscr{M}^2 \quad \rightarrow \quad \mathscr{A}_2\mathscr{X} = \mathscr{B}_2\mathscr{X}\mathscr{M}^4 \quad \rightarrow \quad \mathscr{A}_k\mathscr{X} = \mathscr{B}_k\mathscr{X}\mathscr{M}^{2^k} \tag{12}$$

We seek a structure in $\mathscr{A}$ and $\mathscr{B}$ such that we calculate $\mathscr{A}_k \rightarrow \mathscr{A}_{k+1}$ and $\mathscr{B}_k \rightarrow \mathscr{B}_{k+1}$ by recursions in elements (here we will settle for submatrices). In the following subsections, we provide two recursions of exactly this structure. Both require that we rearrange our pencil $\mathscr{A} - \lambda\mathscr{B}$ to conform to a specific structure of each recursion, as we now show.

3.2. First Standard Form

To obtain the first SDA, we will transform the pencil $\mathscr{A} - \lambda\mathscr{B}$. Assuming that B is non-singular, we receive the primal problem in SF1

$$\mathscr{A}_0\mathscr{X} = \mathscr{B}_0\mathscr{X}\mathscr{M} \tag{13}$$

by multiplying (6) from the left by S, where

$$\mathscr{A}_0 = \begin{pmatrix} E_0 & 0 \\ -X_0 & I \end{pmatrix} := S\mathscr{A}, \qquad \mathscr{B}_0 = \begin{pmatrix} I & -Y_0 \\ 0 & F_0 \end{pmatrix} := S\mathscr{B}, \qquad S = \begin{pmatrix} I & -B^{-1} \\ 0 & B^{-1} \end{pmatrix}$$

Note that (6) and (13) are equivalent in the sense that the pencils $\mathscr{A} - \lambda\mathscr{B}$ and $\mathscr{A}_0 - \lambda\mathscr{B}_0$ share the same set of generalized eigenvalues, i.e., $\text{eig}(\mathscr{A},\mathscr{B}) = \text{eig}(\mathscr{A}_0,\mathscr{B}_0)$. We now use doubling transformations to arrive at a SDA for SF1, computing $\{\mathscr{A}_k\}_{k=0}^{\infty}$, $\{\mathscr{B}_k\}_{k=0}^{\infty}$ with

$$\mathscr{A}_{k+1} = \begin{pmatrix} E_{k+1} & 0 \\ -X_{k+1} & I \end{pmatrix} := \begin{pmatrix} E_k(I - Y_k X_k)^{-1} & 0 \\ -F_k(I - X_k Y_k)^{-1} X_k & I \end{pmatrix} \mathscr{A}_k \tag{14}$$

$$\mathscr{B}_{k+1} = \begin{pmatrix} I & -Y_{k+1} \\ 0 & F_{k+1} \end{pmatrix} := \begin{pmatrix} I & -E_k(I - Y_k X_k)^{-1} Y_k \\ 0 & F_k(I - X_k Y_k)^{-1} \end{pmatrix} \mathscr{B}_k \tag{15}$$

such that

$$\mathscr{A}_k \mathscr{X} = \mathscr{B}_k \mathscr{X} \mathscr{M}^{2^k} \tag{16}$$

A key feature here is that (16) retains SF1 for all $k \in \mathbb{N}$. Algorithm 1 with an initial guess $P_0 = 0$ summarizes this SDA for the SF1. The intuition here is that under some fairly general preconditions, e.g., assumption 1, the term $\mathscr{B}_k \mathscr{X} \mathscr{M}^{2^k}$ on the right-hand side of (16) converges to zero for $k \to \infty$, so that consequently X_k converges to P.

A disadvantage of SDAs is that numerical inaccuracies can propagate through iteration if, say, the matrices to be inverted are not well-conditioned. In the context of discrete algebraic Riccati equations (DAREs), Mehrmann and Tan (1988) show that such a defection of the approximate solution again satisfies a DARE. As a result, after solving a DARE, one can solve the associated DAREs of the approximation error to increase the overall accuracy. Following this, Bini and Meini (2023) show how to incorporate an initial guess to SDA for the SF1 by means of an equivalence transformation of the pencil $\mathscr{A} - \lambda\mathscr{B}$. Huang et al. (2018) discuss similar transformations for algebraic Riccati equations (AREs) in general. We introduce such an initial guess by transforming the eigenvalue problem (6) to

$$\widehat{\mathscr{A}}\, \widehat{\mathscr{X}} = \widehat{\mathscr{B}}\, \widehat{\mathscr{X}}\, \mathscr{M} \tag{17}$$

with P_0 as the initial guess such that $P = \widehat{P} + P_0$ and

$$\widehat{\mathscr{X}} := \begin{pmatrix} I \\ \hat{P} \end{pmatrix} \qquad \widehat{\mathscr{A}} := \mathscr{A} \begin{pmatrix} I & 0 \\ P_0 & I \end{pmatrix}, \qquad \widehat{\mathscr{B}} := \mathscr{B} \begin{pmatrix} I & 0 \\ P_0 & I \end{pmatrix}$$

Assuming that $B + AP_0$ has full rank, we multiply $\widehat{\mathscr{A}}$ and $\widehat{\mathscr{B}}$ by

$$\widehat{S} = \begin{pmatrix} (B + AP_0)^{-1} B & -(B + AP_0)^{-1} \\ I - (B + AP_0)^{-1} B & (B + AP_0)^{-1} \end{pmatrix}$$

to receive the corresponding problem in SF1, i.e.,

$$\widehat{\mathscr{A}_0}\,\widehat{\mathscr{X}} = \widehat{\mathscr{B}}_0\,\widehat{\mathscr{X}}\,\mathscr{M} \tag{18}$$

with

$$\widehat{\mathscr{A}_0} = \begin{pmatrix} \widehat{E}_0 & 0 \\ -\widehat{X}_0 & I \end{pmatrix} := \widehat{S}\widehat{\mathscr{A}}, \qquad \widehat{\mathscr{B}}_0 = \begin{pmatrix} I & -\widehat{Y}_0 \\ 0 & \widehat{F}_0 \end{pmatrix} := \widehat{S}\widehat{\mathscr{B}}$$

Algorithm 1 summarizes the SDA for SF1 based on an initial P_0. Note that a good P_0 should increase the probability that $B + AP_0$ is well-conditioned, since under assumption 1 we know that $B + AP$ has full rank (see Lan and Meyer-Gohde, 2014). In comparison to $P_0 = 0$ above which is only applicable for non-singular B, Algorithm 1 can handle a singular B, provided we have a P_0 such that $B + AP_0$ is non-singular.[10]

Algorithm 1: Structure-Preserving Doubling Algorithm (SF1)

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

Set $\widehat{X}_0, \widehat{Y}_0, \widehat{E}_0, \widehat{F}_0$ according to

$$\widehat{X}_0 = -P_0 - (B + AP_0)^{-1}C, \qquad \widehat{Y}_0 = -(B + AP_0)^{-1}A,$$
$$\widehat{E}_0 = -(B + AP_0)^{-1}C, \qquad \widehat{F}_0 = -(B + AP_0)^{-1}A$$

While *criterion*($\widehat{X}_k$) $> \epsilon$ **do**

  Set $\widehat{E}_{k+1} = \widehat{E}_k\left(I - \widehat{Y}_k\widehat{X}_k\right)^{-1}\widehat{E}_k$ and $\widehat{F}_{k+1} = \widehat{F}_k\left(I - \widehat{X}_k\widehat{Y}_k\right)^{-1}\widehat{F}_k$

  Set $\widehat{X}_{k+1} = \widehat{X}_k + \widehat{F}_k\left(I - \widehat{X}_k\widehat{Y}_k\right)^{-1}\widehat{X}_k\widehat{E}_k$ and $\widehat{Y}_{k+1} = \widehat{Y}_k + \widehat{E}_k\left(I - \widehat{Y}_k\widehat{X}_k\right)^{-1}\widehat{Y}_k\widehat{F}_k$

  Advance $k = k + 1$

end

Return: $\widehat{X}_k + P_0$

3.3. Second Standard Form

Alternatively, Chiang et al. (2009) use a doubling algorithm on the eigenvalue problem

$$\mathscr{A}_0^{\dagger}\mathscr{X}^{\dagger} = \mathscr{B}_0^{\dagger}\mathscr{X}^{\dagger}\mathscr{M} \tag{19}$$

with

$$\mathscr{X}^{\dagger} = \begin{pmatrix} I \\ AP \end{pmatrix}, \quad \mathscr{A}_0^{\dagger} = \begin{pmatrix} E_0^{\dagger} & 0 \\ -X_0^{\dagger} & I \end{pmatrix} := \begin{pmatrix} -C & 0 \\ 0 & I \end{pmatrix}, \quad \mathscr{B}_0^{\dagger} = \begin{pmatrix} -Y_0^{\dagger} & I \\ -F_0^{\dagger} & 0 \end{pmatrix} := \begin{pmatrix} B & I \\ A & 0 \end{pmatrix}, \mathscr{M} = P$$

[10] Chiang et al. (2010) use a similar technique to solve DAREs with singular transition matrices.

which satisfies the so-called Second Standard Form (SF2). Again any P satisfying (5) is also a solution of (19). Note that $\mathscr{A} - \lambda\mathscr{B}$ and $\mathscr{A}_0^\dagger - \lambda\mathscr{B}_0^\dagger$ again share the same set of generalized eigenvalues, i.e., $\mathrm{eig}(\mathscr{A},\mathscr{B}) = \mathrm{eig}(\mathscr{A}_0^\dagger,\mathscr{B}_0^\dagger)$.

Similarly to SF1 above, the SDA for SF2 computes sequences $\{\mathscr{A}_k^\dagger\}_{k=0}^\infty$, $\{\mathscr{B}_k^\dagger\}_{k=0}^\infty$ with

$$\mathscr{A}_{k+1}^\dagger = \begin{pmatrix} E_{k+1}^\dagger & 0 \\ -X_{k+1}^\dagger & I \end{pmatrix} := \begin{pmatrix} E_k^\dagger\left(X_k^\dagger - Y_k^\dagger\right)^{-1} & 0 \\ F_k^\dagger\left(X_k^\dagger - Y_k^\dagger\right)^{-1} & I \end{pmatrix} \mathscr{A}_k \tag{20}$$

$$\mathscr{B}_{k+1}^\dagger = \begin{pmatrix} -Y_{k+1}^\dagger & I \\ -F_{k+1}^\dagger & 0 \end{pmatrix} := \begin{pmatrix} I & E_k^\dagger\left(X_k^\dagger - Y_k^\dagger\right)^{-1} \\ 0 & F_k^\dagger\left(X_k^\dagger - Y_k^\dagger\right)^{-1} \end{pmatrix} \mathscr{B}_k \tag{21}$$

such that

$$\mathscr{A}_k^\dagger \mathscr{X}^\dagger = \mathscr{B}_k^\dagger \mathscr{X}^\dagger \mathscr{M}^{2^k} \tag{22}$$

Note that compared to the SDA for SF1, the sequence $\{X_k^\dagger\}_{k=0}^\infty$ now converges to AP instead of P. However, in case of convergence we may obtain a P as $-(X_k^\dagger + B)^{-1}C$.

Following the idea of Bini and Meini (2023) and applying it to the SDA for SF2, we take again P_0 as the initial guess such that $P = \widehat{P} + P_0$ and insert this into the UQME (5)

$$0 = A\left(\widehat{P} + P_0\right)^2 + B\left(\widehat{P} + P_0\right) + C = A\widehat{P}^2 + A\widehat{P}P_0 + (AP_0 + B)\widehat{P} + AP_0^2 + BP_0 + C \tag{23}$$

This can be rewritten as (19) with

$$\widehat{\mathscr{X}}^\dagger = \begin{pmatrix} I \\ A\widehat{P} \end{pmatrix}, \quad \widehat{\mathscr{A}}_0^\dagger = \begin{pmatrix} \widehat{E}_0^\dagger & 0 \\ -\widehat{X}_0^\dagger & I \end{pmatrix} := \begin{pmatrix} -C & 0 \\ AP_0 & I \end{pmatrix}, \quad \widehat{\mathscr{B}}_0^\dagger = \begin{pmatrix} -\widehat{Y}_0^\dagger & I \\ -\widehat{F}_0^\dagger & 0 \end{pmatrix} := \begin{pmatrix} AP_0 + B & I \\ A & 0 \end{pmatrix}, \mathscr{M} = P$$

Algorithm 2 summarizes the SDA for SF2 based on an initial guess P_0.

While this is a straightforward application of the initial guess approach of Algorithm 1 to the SDA for SF2, the algorithm will deliver the same approximation to P regardless of P_0. We summarize this in the following (Huang et al., 2018, Theorem 3.32).

Theorem 2. *Let $X_t^\dagger$, $Y_t^\dagger$, $E_t^\dagger$, and $F_t^\dagger$ denote the quantities of the Structure-Preserving Doubling Algorithm (SF2) and let $\widehat{X}_t^\dagger$, $\widehat{Y}_t^\dagger$, $\widehat{E}_t^\dagger$, and $\widehat{F}_t^\dagger$ be the quantities of the Structure-Preserving Doubling Algorithm (SF2) with an initial guess P_0. Then we have that*

$$\widehat{X}_t^\dagger = X_t^\dagger - AP_0, \qquad \widehat{Y}_t^\dagger = Y_t^\dagger - AP_0$$
$$\widehat{E}_t^\dagger = E_t^\dagger, \qquad \widehat{F}_t^\dagger = F_t^\dagger$$

Hence, both algorithms will eventually return the same approximation for P.

Proof. See the appendix. □

Algorithm 2: Structure-Preserving Doubling Algorithm (SF2)

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

Set $\widehat{X}_0^\dagger$, $\widehat{Y}_0^\dagger$, $\widehat{E}_0^\dagger$, $\widehat{F}_0^\dagger$ according to

$$\widehat{X}_0^\dagger = -AP_0, \qquad \widehat{Y}_0^\dagger = -(AP_0 + B), \qquad \widehat{E}_0^\dagger = -C, \qquad \widehat{F}_0^\dagger = -A$$

While *criterion*($\widehat{X}_k^\dagger$) $> \epsilon$ **do**

Set $\widehat{E}_{k+1}^\dagger = \widehat{E}_k^\dagger \left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right)^{-1} \widehat{E}_k^\dagger$ and $\widehat{F}_{k+1}^\dagger = \widehat{F}_k^\dagger \left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right)^{-1} \widehat{F}_k^\dagger$

Set $\widehat{X}_{k+1}^\dagger = \widehat{X}_k^\dagger - \widehat{F}_k^\dagger \left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right)^{-1} \widehat{E}_k^\dagger$ and $\widehat{Y}_{k+1}^\dagger = \widehat{Y}_k^\dagger + \widehat{E}_k^\dagger \left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right)^{-1} \widehat{F}_k^\dagger$

Advance $k = k + 1$

end

Return: $-(AP_0 + \widehat{X}_k^\dagger + B)^{-1} C$

We see that it is not trivial to generate versions of these algorithms that enable refinement of arbitrary initializations of the solution. The results above are known for Riccati equations, we have extended them here to the specific case of our UQME. That we can provide a version of SF1 in Algorithm 1 which operates on arbitrary initializations is all the more interesting as it can potentially profit in terms of increased accuracy and reduced computation time from initializations that are close to the stable solvent P being sought. We will confirm this and, following theorem 2, that the algorithm 2 does not.

4. Theoretical and Practical Considerations

We now turn to theoretical and practical considerations relating to the doubling methods. While the applied mathematics literature establishes quadratic convergence of these algorithms under general conditions, our contribution lies in linking these convergence results to the standard solution criteria familiar to DSGE practitioners—specifically, the Blanchard and Kahn (1980) conditions formalized in assumption 1. We also show that, due to structural similarities, the cyclic and logarithmic reduction algorithms—already available in Dynare but only lightly documented—converge quadratically under the same conditions. From a practitioner's perspective, it is crucial to know that these alternative methods are theoretically sound when applied to DSGE models. Finally, we discuss the measures of solution accuracy we will employ in the applications.

4.1. Cyclic and Logarithmic Reduction

SDAs generate a sequence of matrix pencils, in each step squaring the corresponding eigenvalues. Similarly, Bini and Meini's (1996) cyclic reduction and Latouche and Ramaswami's (1993) logarithmic reduction, generate sequences of matrix polynomials whose eigenvalues are squared in each step. In the following we outline the idea of the cyclic and logarithmic reduction and link them to the SDA of SF2.[11]

Cyclic reduction computes the sequences $\{A_k\}_{k=0}^{\infty}$, $\{B_k\}_{k=0}^{\infty}$, $\{C_k\}_{k=0}^{\infty}$, and $\{\widehat{B}_k\}_{k=0}^{\infty}$ with

$$A_{k+1} = -A_k\, B_k^{-1}\, A_k, \qquad A_0 = A \tag{24}$$

$$B_{k+1} = B_k - A_k\, B_k^{-1}\, C_k - C_k\, B_k^{-1}\, A_k, \qquad B_0 = B \tag{25}$$

$$C_{k+1} = -C_k\, B_k^{-1}\, C_k, \qquad C_0 = C \tag{26}$$

$$\widehat{B}_{k+1} = \widehat{B}_k - A_k\, B_k^{-1}\, C_k \qquad \widehat{B}_0 = B \tag{27}$$

Using divide-and-conquer, cyclic reduction defines a sequence of UQMEs with

$$0 = A_k \mathscr{M}_k^2 + B_k \mathscr{M}_k + C_k, \qquad 0 = A_k \mathscr{M}_k \mathscr{M} + \widehat{B}_k \mathscr{M} + C, \tag{28}$$

where $\mathscr{M}_k = \mathscr{M}^{2^k}$. Furthermore, assuming that $\lim_{k\to\infty} \widehat{B}_k^{-1}$ exists and that $\lim_{k\to\infty} \widehat{B}_k^{-1} A_k \mathscr{M}_k \mathscr{M} = 0$ we can express the solvent P as $P = -\lim_{k\to\infty} \widehat{B}_k^{-1} C$.

As pointed out by Bini et al. (2011, pp. 167) and Chiang et al. (2009, pp. 236) the SDA for SF2 is connected to cyclic reduction. In detail, it follows via induction that

$$\widehat{B}_k = B + X_k^{\dagger}, \qquad B_k = X_k^{\dagger} - Y_k^{\dagger}, \qquad C_k = -E_k^{\dagger}, \qquad A_k = -F_k^{\dagger}, \qquad \forall k \geq 0$$

Hence, the SDA for SF2 with $P_0 = 0$ and cyclic reduction are theoretically equivalent.

This equivalence with a special case ($P_0 = 0$) of an SDA is significant as this cyclic reduction is included yet only mildly documented in Dynare. The same is true for the logarithmic reduction by Latouche and Ramaswami (1993), which uses the same divide-and-conquer strategy to obtain $\{H_k\}_{k=0}^{\infty}$, $\{L_k\}_{k=0}^{\infty}$, $\{\widehat{H}_k\}_{k=0}^{\infty}$, and $\{\widehat{L}_k\}_{k=0}^{\infty}$ with

$$H_{k+1} = (I - H_k L_k - L_k H_k)^{-1} H_k^2, \qquad H_0 = -B^{-1} A \tag{29}$$

$$L_{k+1} = (I - H_k L_k - L_k H_k)^{-1} L_k^2, \qquad L_0 = -B^{-1} C, \tag{30}$$

$$\widehat{H}_{k+1} = \widehat{H}_k H_{k+1}, \qquad \widehat{H}_0 = -B^{-1} A, \tag{31}$$

$$\widehat{L}_{k+1} = \widehat{L}_k + \widehat{H}_k L_{k+1}, \qquad \widehat{L}_0 = -B^{-1} C, \tag{32}$$

[11] For a comprehensive treatment see Bini et al. (2011, Chapter 5.2) and Bini et al. (2005, Chapter 7)

that define a sequence of UQMEs similarly to the cyclic reduction above with

$$0 = H_k \mathcal{M}_k^2 - \mathcal{M}_k + L_k. \qquad\qquad 0 = \widehat{H}_k \mathcal{M}_{k+1} - \mathcal{M} + \widehat{L}_k \tag{33}$$

As pointed out by Bini et al. (2005, Theorem 7.5), it follows via induction that for all $k \geq 0$

$$H_k = -B_k^{-1} A_k, \qquad\qquad L_k = -B_k^{-1} C_k.$$

Consequently, there is also a link between the SDA for SF2 with $P_0 = 0$ and the logarithmic reduction. In detail, we get for all $k \geq 0$ that

$$H_k = \left(X_k^\dagger - Y_k^\dagger\right)^{-1} F_k^\dagger \qquad\qquad L_k = \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger$$

Similar to the cyclic reduction assuming here $\lim\limits_{k\to\infty} \widehat{H}_k \mathcal{M}_{k+1} = 0$ the stable solvent is

$$P = \lim_{k\to\infty} \widehat{L}_k, \tag{34}$$

Pseudocode of both reductions in our notation are in the online appendix. Compared to the doubling algorithm, the reductions follow a similar idea by squaring the eigenvalues of matrix polynomials. Beyond that, abstracting from numerical inaccuracies, we find that the SDA for SF2 with $P_0 = 0$ and the cyclic reduction will deliver identical approximation to P for a given number of iterations. While the two reduction generate interchangeable sequences of UQMEs (the left equations in (28) and (33)), the two algorithms differ in the way in which they recover the approximation to P (the right equations in (28) and (33)). Hence, although we can link some quantities computed by the logarithmic reduction to the quantities of the cyclic reduction / SF2, we will in general receive distinct solutions P.

4.2. Convergence

A major advantage of SDAs—regardless of a particular starting value—is that they provide quadratic convergence at relatively low computational cost per iteration. We establish sufficient conditions for quadratic convergence in the following theorem.

Theorem 3 (Convergence)**.** *Suppose P and P_d exist and satisfy assumption 1 (and hence corollary 1). Then following statements are true.*

(1) If the sequences $\{X_k\}_{k=0}^\infty$, $\{Y_k\}_{k=0}^\infty$, $\{E_k\}_{k=0}^\infty$, and $\{F_k\}_{k=0}^\infty$ are well defined, i.e., all the inverses exist during the doubling iteration process, X_k converges to P quadratically, and moreover, $\limsup\limits_{k\to\infty} \|X_k - P\|^{1/2^k} \leq \rho(P)\cdot\rho(P_d)$.

(2) *If the sequences* $\{X_k^\dagger\}_{k=0}^\infty$, $\{Y_k^\dagger\}_{k=0}^\infty$, $\{E_k^\dagger\}_{k=0}^\infty$, *and* $\{F_k^\dagger\}_{k=0}^\infty$ *are well defined, i.e., all the inverses exist during the doubling iteration process,* $X_k^\dagger$ *converges to AP quadratically, and moreover,* $\limsup_{k\to\infty} \|X_k^\dagger - AP\|^{1/2^k} \leq \rho(P)\cdot\rho(P_d)$.

(3) *If the sequences* $\{L_k\}_{k=0}^\infty$, $\{H_k\}_{k=0}^\infty$, $\{\widehat{L}_k\}_{k=0}^\infty$, *and* $\{\widehat{H}_k\}_{k=0}^\infty$ *are well defined, i.e., all the inverses during the Logarithmic Reduction exist,* $\widehat{L}_k$ *converges to P quadratically, and moreover,* $\limsup_{k\to\infty} \|\widehat{L}_k - P\|^{1/2^k} \leq \rho(P)\cdot\rho(P_d)$.

Proof. See the appendix. □

Theorem 3 also applies to the cyclic reduction due to its equivalence to the SDA for SF2. Consequently, under assumption 1, all algorithms presented above, from SDA for SF1 to logarithmic reduction, will converge quadratically to the unique (semi) stable solution P.

4.3. Accuracy

To compare the different algorithms we need a measure of accuracy, that is, how far the solution produced by a particular method, $\hat{P}$, is from the true solution, P. As the latter is not known we will appeal to two measures, one from numerical stability analysis and another by comparing to arbitrary precision. For the numerical measure, we use the tighter forward error bound of Meyer-Gohde (2023)

$$\underbrace{\frac{\|P-\hat{P}\|_F}{\|P\|_F}}_{\text{Forward Error}} \leq \underbrace{\frac{\left\|H_{\hat{P}}^{-1}\mathrm{vec}\left(R_{\hat{P}}\right)\right\|_2}{\|\hat{P}\|_F}}_{\text{Forward Error Bound}} \tag{35}$$

where the residual of the UQME is $R_{\hat{P}} = A\hat{P}^2 + B\hat{P} + C$ and $H_{\hat{P}} = I_{n_y} \otimes \left(A\hat{P} + B\right) + \hat{P}' \otimes A$, and the forward error relative to a high precision solution of the problem, $\hat{P}_{HP}$,

$$\underbrace{\frac{\|\hat{P}_{HP}-\hat{P}\|_F}{\|\hat{P}_{HP}\|_F}}_{\text{High Precision Forward Error}} \tag{36}$$

The righthand side of (35) can be rearranged so the numerator solves a Sylvester equation and we use the Advanpix Multiprecision Toolbox (Advanpix LLC., 2025) for $\hat{P}_{HP}$.[12] We will see in the applications that the two measures provide essentially the same answer, which favors (35) that operates directly with the numerical solution, $\hat{P}$. Nonetheless, having both measures should support belay skepticism in either particular measure .

[12]We solve the Sylvester with ACM Algorithm 705 from Gardiner et al. (1992) and `mepack_gsylv` from Köhler (2024) and, for the high precision solution, 20 digits is sufficient in our experiments.

5. Applications

We run through two different sets of experiments to assess the SDAs—in the model of Smets and Wouters (2007) and on the suite of models in the Macroeconomic Model Data Base (MMB). These two sets enable us to assess the different methods firstly in a specific, policy relevant model and then also in non-model specific manner, to give us insight on how robust our results are across models. We focus here on comparing our algorithms above with Dynare's QZ-based method,[13] Dynare's cyclic and logarithmic reduction methods,[14] the baseline Newton method from Meyer-Gohde and Saecker (2024), and the baseline Bernoulli method from Meyer-Gohde (2024), who also run these experiments.[15] We assess the performance with respect to the accuracy, computational time, and convergence to the stable solvent. We examine the consequences of initializing both from zero matrix (an uninformed initialization of a stable solvent or the standard initialization for the reduction and structure preserving doubling algorithms) and Dynare's QZ solution.

5.1. Smets and Wouters's (2007) Model

We start with Smets and Wouters's (2007) medium scale, estimated DSGE model that is a benchmark for policy analysis. They estimate and analyze a New Keynesian 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. For our purposes, the monetary policy rule is particularly important, as we will compare the accuracy of different solution methods when solving under alternate, but nearby parameterizations of the following Taylor rule

$$r_t = \rho r_{t-1} + (1-\rho)(r_\pi \pi_t + r_Y(y_t - y_t^p)) + r_{\Delta y}((y_t - y_t^p) - (y_{t-1} - y_{t-1}^p)) + \varepsilon_t^r, \tag{37}$$

where r_t is the interest rate, π_t inflation, and $(y_t - y_t^p)$ the current output gap. The parameters r_π, r_Y and $r_{\Delta y}$ describing the sensitivity of the interest rate to each of these variables, and also the change in the output gap, and ρ measures interest rate smoothing.

[13]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`.

[14]We used `Dynare 5.x`. The recent release of `Dynare 6.x` has compiled `.mex` versions of the cyclic and logarithmic reduction methods. We continue to use the Matlab implementations to maintain comparability, but note that this means we understate the potential competitiveness with Dynare's compiled QZ.

[15]We follow Dynare's QZ and reduce the dimensionality of the problem for our implementations of the doubling algorithms SF1 and SF2 by grouping variables. The details are in the online appendix.

Method	Relative Performance		Forward Errors		Iterations
	Run Time	Max Abs. Diff.	Bound	High Precision	
Dynare (QZ)	0.00081	—	5.2e-14	5.1e-14	1
Dynare (Cyclic Reduction)	0.97	7.4e-13	2.9e-15	7.6e-15	10
Dynare (Log Reduction)	1.5	1e-12	9.1e-16	3.1e-14	9
Baseline Newton	479	1.1e+02	7.2e-11	8.6	1e+03
Baseline Bernoulli	12	7.7e-13	3.7e-14	3.9e-14	436
Doubling SF1	2.1	7.8e-13	8.6e-15	1.3e-14	10
Doubling SF2	1.2	7e-13	8.1e-15	6.8e-15	10

TABLE 1. Smets and Wouters (2007), Posterior Mode

- Run Time for Dynare (QZ) in seconds, for all others, run time relative to Dynare.
- Max Abs. Diff. measures the largest abs. difference from the P produced by Dynare.

The results for the posterior mode of Smets and Wouters (2007) can be found in table 1. Compared with the QZ algorithm, both the doubling algorithms, SF1 and SF2, along with the reduction algorithms already implemented in Dynare, perform very comparably. This is in contrast to the baseline Newton algorithm that fails to converge to the stable solvent (there is no guarantee that this algorithm will converge to a particular solvent though extensions of this baseline algorithm perform more favorably). The baseline Bernoulli algorithm gives a tradeoff repeated throughout that analysis: generally more accurate, but often about an order of magnitude slower, than QZ. This is not the case with our doubling algorithms, they are as fast or faster than QZ and provide an order of magnitude more of accuracy. All four of the algorithms explored here, doubling and reduction, require about 10 iterations to converge and provide solutions that do not differ in an economic sense from the solution provided by QZ. Note that both of the measures of accuracy, the forward error bound in (35) and the error relative to a high precision solution (36), give similar results with the exception being the baseline Newton algorithm that solves the matrix quadratic, the forward error bound of (35) is small, but with a different solution than the rest of the algorithms—i.e., it converged to a solution containing unstable eigenvalues.

In table 2 we examine the different methods as solution refinement techniques, by parameterizing the model of Smets and Wouters (2007) to a numerical instability following Meyer-Gohde (2023), see the online appendix for this parameterization, and initializing the different methods at the Dynare QZ solution. Note that the reduction methods cannot work with an initial value for the solution, so we compare the solution provided by Dynare's QZ with the doubling refinement versions of SF1 and SF2 as well as the

Method	Relative Performance		Forward Errors		Iterations
	Run Time	Variance π_t	Bound	High Precision	
Dynare (QZ)	0.0014	0.28	6.4e-13	1.0e-11	1
Baseline Newton	2.3	0.32	7.6e-17	5.5e-15	4
Baseline Bernoulli	2	0.35	1.2e-15	3.6e-13	90
Doubling SF1	1.9	0.33	2.9e-16	3.3e-14	11
Doubling SF2	1.4	0.43	9.1e-14	8.9e-13	12
High Precision	–	0.33	1.5e-17	–	–

TABLE 2. Smets and Wouters (2007), Numerically Problematic Parameterization

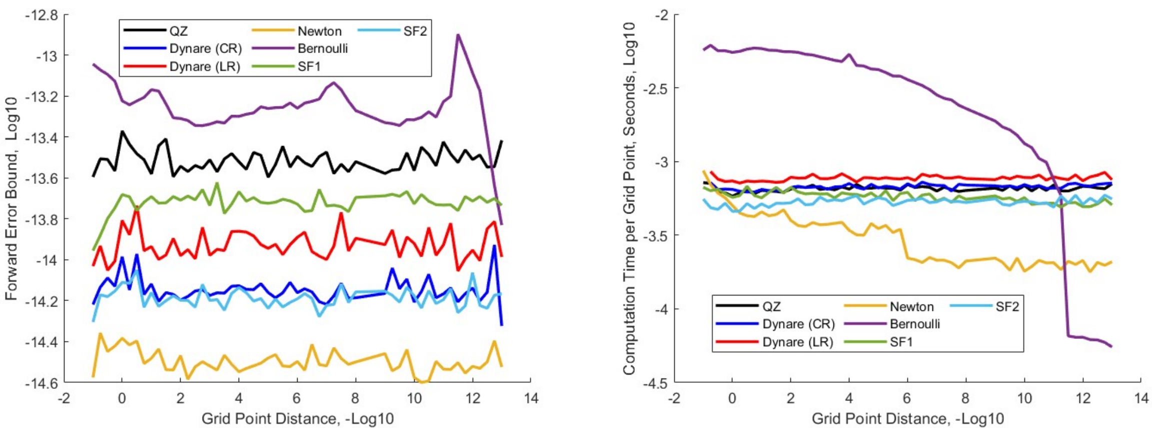
- Run Time for Dynare in seconds, for all others, run time relative to Dynare.
- Variance π_t gives the associated value for the population or theoretical variance of inflation.

Newton and the Bernoulli method. First, consistent with theorem 2 and, hence, the irrelevance of initial conditions for that algorithm, the SF2 algorithm does not appear to be comparable with the Newton or SF1 algorithm, converging to a solution that has a much less substantial improvement in accuracy. As above, the two different measures of accuracy we examine agree on the relative ordering of the different algorithms. The Newton algorithm provides substantially more accuracy than the Bernoulli algorithm but at a higher additional computational cost, echoing a tradeoff mentioned above. The SF1 algorithm breaks through this barrier, providing roughly the same level of accuracy as the Newton algorithm at less computational costs. That the Newton method is so time consuming despite the relatively few number of iterations performed emphasizes the computational intensity of the Newton step that is not shared by the doubling algorithms—the former involves solving structured linear (Sylvester) equations whereas the latter solve standard systems of linear equations. The resulting implied variances of inflation, π_t, differ on an economically relevant scale. The more accurate methods all agree that the QZ solution understates the variance of inflation.[16]

We now take this refinement perspective and apply the different algorithms to solve iteratively for different parameterizations of the Taylor rule in the model of Smets and Wouters (2007). We would like to establish whether solutions from previous, nearby parameterizations can be used to efficiently initialize the doubling methods similarly to the experiment above with the QZ solution as the initial guess. To that end, the

[16] The different refinements still do not entirely agree on the actual level of the variance as this parametrization is very poorly conditioned—see Meyer-Gohde (2023)—and hence subsequent calculations as very sensitive to small differences in the solutions. We proceed uniformly with Dynare's native theoretical moment calculator for each method to calculate the variance and it warns of this ill condition.

experiment iterates through an evenly spaced 10×10 grid for r_π and r_y, the responses to inflation and the output gap in the Taylor rule (37), holding all other parameters at their posterior mode values. We iterate through the grid taking the solution from the previous parameterization as the initialization for the next iteration, that is $P_0|_{r_\pi=r_{\pi,j},r_y=r_{y,1}} = \hat{P}|_{r_\pi=r_{\pi,j-1},r_y=r_{y,1}}$ initializing with $P_0|_{r_\pi=r_{\pi,1},r_y=r_{y,1}} = \hat{P}^{QZ}|_{r_\pi=r_{\pi,1},r_y=r_{y,1}}$ for the first row of the grid in the different values of r_π and then $P_0|_{r_\pi=r_{\pi,j},r_y=r_{y,j}} = \hat{P}|_{r_\pi=r_{\pi,j},r_y=r_{y,j-1}}$ to fill out the grid in the values of r_y. Thus we initialize all algorithms at the QZ solution and, as we move to nearby parameterizations, we measure the time and accuracy of the different methods. We calculate the median over the whole grid and then repeat the experiment with the grid points closer together by varying the size of the interval in the grid—setting $r_\pi \in [1.5, 1.5 \times (1+10^{-x})]$ and $r_Y \in [0.125, 0.125 \times (1+10^{-x})]$, where $x \in [-1, 8]$. An increase in x is a decrease in the spacing between the 100 grid points that thus increases the precision of the initialization.



(A) Forward Error Bound (B) Computation Time per Grid Point

FIGURE 1. Forward Errors and Computation Time per Grid Point for different parameterizations of the model by Smets and Wouters (2007), log10 scale all axes.

Figure 1 summarizes the experiment graphically. Firstly, the left panel shows that the accuracy of the algorithms is independent of the grid spacing (and, hence, how close the parameter steps are from each other), this is expected and confirms the consistency of the algorithms and our measures of their accuracy.[17] The Bernoulli method is the exception and displays a significant drop in forward errors that coincides with the drop in computation time in the right figure—at a close enough parameterization, the Bernoulli algorithm starts with a guess from the previous parameterization that is accurate beyond

[17]We depict only the bound, the forward error relative to the high precision solution is essentially redundant.

the convergence threshold and the single iteration that is performed provides substantial (relative to more widely spaced grids) accuracy gains. In terms of their relative accuracies, the Newton is the most and the Bernoulli is generally (that is, except at the very closely spaced grids) the least accurate. All of the doubling and reduction algorithms are more accurate than QZ, with the cyclic reduction and SF2 algorithms roughly the same (as we would expect from section 4.1) and close to Newton, followed by the log reduction, and finally SF1 doubling. As to be expected from theorem 2, the SF2 doubling algorithm does not systematically profit in terms of reduced computation time as the parameter initializations improve (due to closer grid spacing) as the Newton and Bernoulli algorithms do. The SF1 doubling algorithm, in contrast, does profit with a downward trend in the computation time, although the Newton and Bernoulli algorithms clearly profit much more substantially with more significant downward trends.

5.2. MMB Suite Comparison

To guard against our comparison of methods from being specific to a particular model, we also adopt the model-robust approach inherit in the Macroeconomic Model Data Base (MMB). By abstracting from individual model characteristics we can gain insights into the algorithms that are robust to the particular features of a specific model. Besides drawing then general conclusions, we show an advantage for the SDAs with increasing model size. We implement version 3.3 of the MMB with the comparison repository Rep-MMB[18], which contains over 160 different models, ranging from small scale to large scale models and including estimated models of the US, EU, and multi-country economies. We test the algorithm on a subset of models appropriate for linear, rational expectations DSGE soution[19]—a histogram of the differing sizes can be found in figure 2. With this large set of models, we feel confident in asserting that our assessment of the algorithms is robust to a representative sample of models relevant to policy settings.

We solve each model in the MMB first initializing with the zero matrix and then with the QZ solution provided by Dynare. To get a reliable measure of the computation times, we solve each model 100 times and take the average computation time of the middle three quintiles to minimize the effects of outliers. [20]

[18]See the Github Repository and Website.

[19]Currently, this is 142 models. 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.

[20]Due to uncontrollable demands on, e.g., working memory, run times can differ between runs.

Initialization: Zero Matrix

Method	Convergence	Run Time				Forward Error Bound				Forward Error High Precision				Iterations
		Mean	Med.	Min	Max	Mean	Med.	Min	Max	Mean	Med.	Min	Max	Med.
Dynare (QZ)	—	1	1	1	1	1	1	1	1	1	1	1	1	1
Dynare (Cyclic Red.)	109	1.4	1	0.28	15	55	0.53	0.00093	2e+03	54	0.45	0.0011	2.1e+03	11
Dynare (Log Red.)	129	1.5	1.3	0.16	4.1	2.2e+02	0.096	7.1e-05	1.4e+04	3.4e+03	0.36	0.00089	3.1e+05	10
Baseline Newton	72	1.7e+02	8.8	0.78	1e+03	2.1e+07	0.22	0.00055	7.4e+08	2e+07	0.15	0.0035	7.3e+08	14
Baseline Bernoulli	140	2.6e+03	14	0.12	4.4e+04	3.8	1.3	0.00063	89	3.5	1.2	0.00021	82	5.9e+02
Doubling SF1	131	2.6	2.1	0.13	8.9	2.5e+03	0.24	0.00028	1.6e+05	2.2e+03	0.21	0.00032	1.4e+05	11
Doubling SF2	132	1.6	1.2	0.067	30	4.3	0.21	0.00065	3e+02	4.2	0.15	0.00032	2.8e+02	11

Initialization: Dynare (QZ)Solution

Method	Convergence	Run Time				Forward Error Bound				Forward Error High Precision				Iterations
		Mean	Med.	Min	Max	Mean	Med.	Min	Max	Mean	Med.	Min	Max	Med.
Dynare (QZ)	—	1	1	1	1	1	1	1	1	1	1	1	1	1
Baseline Newton	128	1.7e+02	3.4	0.081	8.4e+02	2.5e+07	0.21	0.0004	1.6e+09	2.4e+07	0.18	0.00024	1.5e+09	5
Baseline Bernoulli	142	2.5e+03	0.28	0.0064	5.1e+04	0.68	0.75	0.00078	3	0.64	0.77	0.0003	2.7	2
Doubling SF1	142	2.5	1.9	0.13	7.5	0.33	0.2	0.00028	4.9	0.33	0.18	0.00039	7.2	11
Doubling SF2	133	1.7	1.3	0.072	32	4.4	0.2	0.00054	3e+02	4.2	0.14	0.00036	2.8e+02	12

TABLE 3. MMB Results

- Run time and forward errors relative to Dynare, number of models that converge to the stable solution and number of iterations in absolute terms.

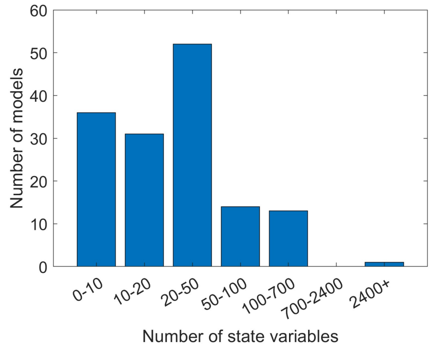
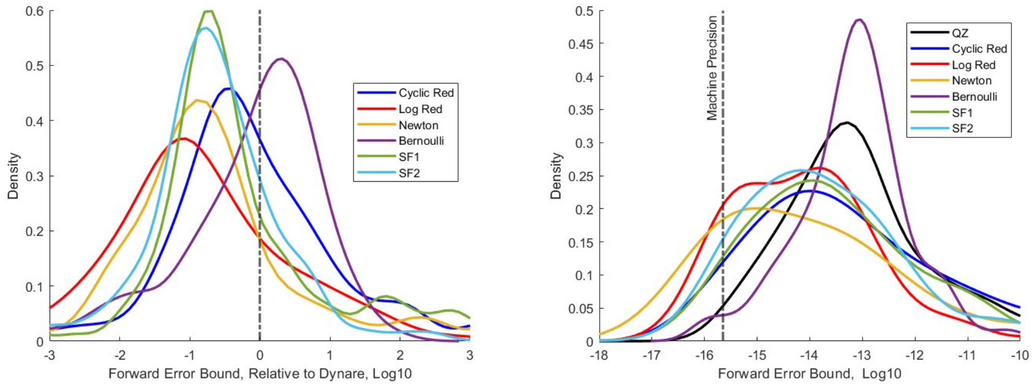


FIGURE 2. Histogram over the number of state variables for the 142 MMB models

We begin with the initialization at the zero matrix and the top half of table 3 summarizes the results. As noted in our previous studies, the Newton method is somewhat slower but more accurate than QZ when it converges to the stable solution—yet it does this in its baseline form only slightly more than half the time—and the baseline Bernoulli algorithm (almost) always converges to the unique stable solution—but does so more slowly than QZ at about the same level of accuracy. The cyclic reduction method is an intermediate case, converging to the stable solution for about 3/4 of the models, at generally comparable computation time, and somewhat higher accuracy than QZ. The logarithmic reduction is an improvement, converging for more models, and doing so at about the same speed and somewhat more accurately than the cyclic reduction. In terms of our doubling algorithms, they converge more frequently than even the logarithmic reduction (but still not for all and when, this failure is due to a breakdown of nonsingularity of the coefficient matrices in their recursions). As we will see in the next experiment, this can be overcome at least for the SF1 doubling algorithm by an appropriate (re)initialization of the algorithm.[21] The SF2 doubling algorithm on average outperforms QZ both in terms of speed and accuracy, although not for every model as the max or worst case shows (and again with the caveat that it does not successfully converge for 10 of the 142 models). The SF1 algorithm is not quite as fast at the median but has a far lower worst case computation time. This average performance of SF2 being faster than SF1 is consistent with the former inverting only one matrix, $X_k^\dagger - Y_k^\dagger$—see Algorithm 2, whereas the latter needs to invert two, $I - Y_k X_k$ and $I - X_k Y_k$—see Algorithm 1. Comparing the errors of the two doubling algorithms, SF2 has the lower worst case and mean forward error of the two and SF1 has the most accurate

[21]This will be apparent in the next experiment when we initialize at the QZ solution and examine the algorithms' potential as solution refinement methods. The online appendix contains an alternative, a middle ground between the zero matrix and the QZ initializations that provides an (imperfectly) informative initialization without relying on a solution from another algorithm like QZ. This does improve the convergence of SF1 but does not resolve it entirely and comes with its own additional computational costs.

best case model and second best worst case model. Again, the information provided by the two error measures roughly coincides. In sum, the doubling algorithms (less so the reduction algorithms) perform favorably relative to QZ.

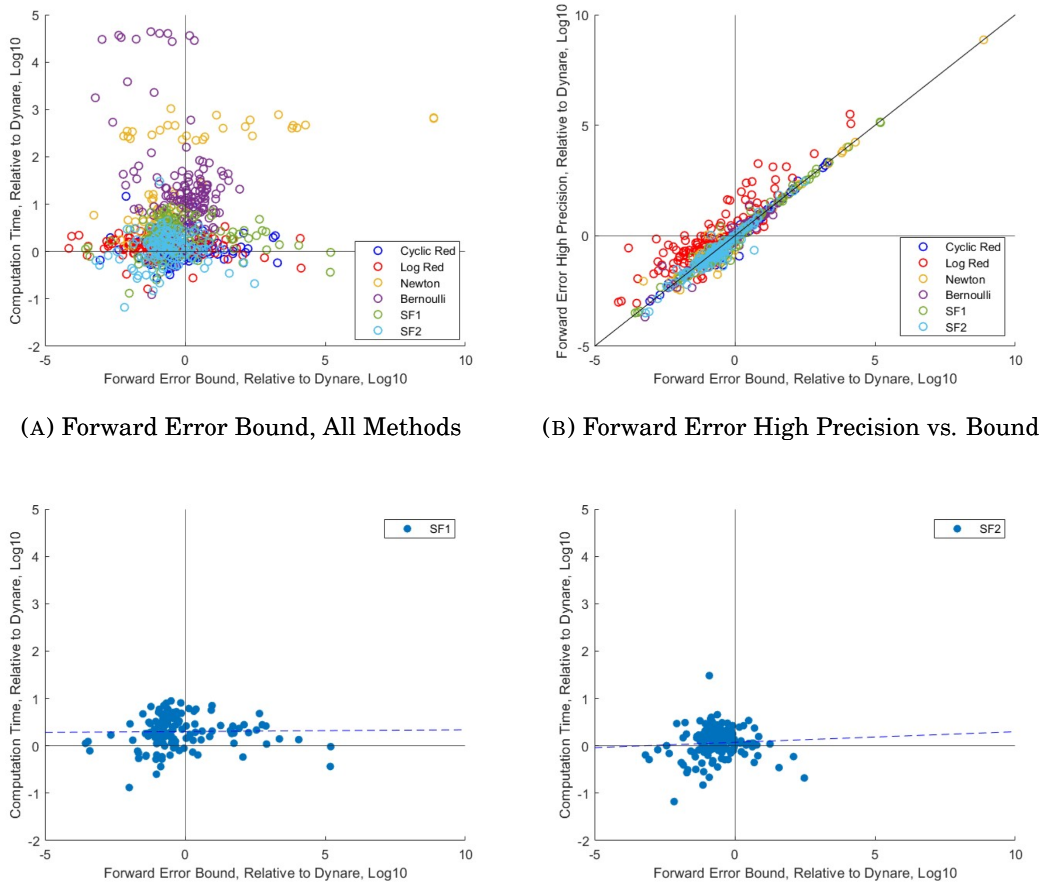


(A) Relative to Dynare (B) Forward Error Bound

FIGURE 3. Distribution of forward error bound for the MMB

A graphical overview of the entire distribution of forward errors is plotted in figure 3, the left relative to QZ and the right in absolute terms. The Newton algorithm is the most accurate with a clearly left shifted distribution relative to Dynare—with a mode improvement of about one order of magnitude—and the Bernoulli method the least with a clearly right shifted distribution relative to QZ. All of the methods here, reduction and doubling, lie in between but with modes and medians all to the left of QZ. While they are all very comparable, the SF2 algorithm has almost its entire mass to the left of Dynare and the cyclic reduction algorithm is right skewed with a substantial mass to the right of Dynare. The absolute values show that almost all of the models have errors less than 1e-10 and the accuracy of the Newton method is also clear with a substantial mass of forward error bounds inside machine precision and all mass essentially below e-12. The distributions of forward errors relative to the high precision solution (36) give virtually the same picture, as we show in figure 4c. In any case, it appears that the models in the MMB were solved with acceptable accuracy and the SF2 doubling or logarithmic reduction is to be preferred among the algorithms here.

Figure 4 provides a model-by-model comparison of the different algorithms' performance relative to QZ with each dot being the result of a model. All four panels express computation times and forward errors relative to Dynare on a log scale—hence a negative value in a dimension means the algorithm is more accurate or computationally efficient than QZ. In the top left panel, all of the methods are plotted using the forward error

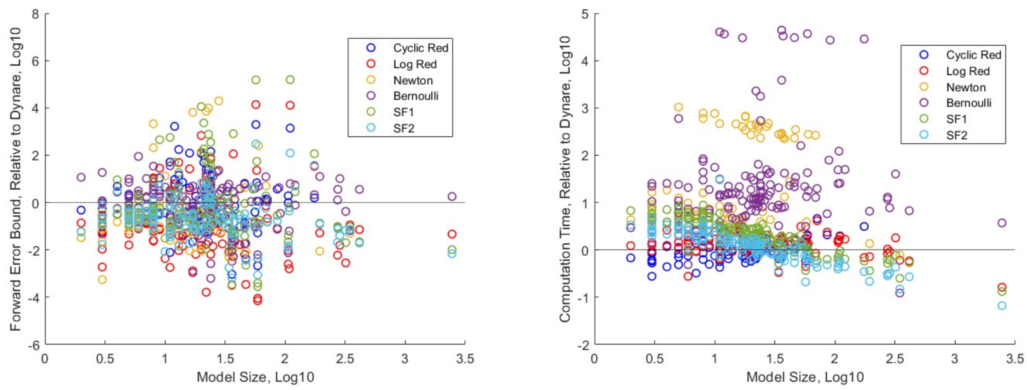


(A) Forward Error Bound, All Methods (B) Forward Error High Precision vs. Bound

(C) SDA SF1 (D) SDA SF2

FIGURE 4. Forward Errors and Computation Time, log10 scales, for the MMB

bound. First, recognize that the Bernoulli method is sometimes more and sometimes less accurate than QZ but usually slower (higher computation time), whereas the baseline Newton method is generally more accurate but usually slightly slower than QZ. The doubling and reduction algorithms require similar computation times as QZ (almost always within one half an order of magnitude slower/faster) but are generally more accurate. The doubling algorithms are more accurate than the reduction algorithms (notice the outlier along the x axis of the reduction algorithms with more than three orders of magnitude higher forward errors than QZ) and the SF2 doubling algorithm is the most accurate of the doubling algorithms as can be seen by comparing the lower two panels—visually, this algorithm is on average slightly faster than QZ and provides an order of magnitude more accuracy. The upper right panel plots the forward error relative to a high precision solution to the numerical forward error bound—as all points, hence the solutions from all algorithms for all models, lie along the 45 degree line, we conclude these two measures provide the same information. This provides strong evidence in favor of Meyer-Gohde's (2023) practical forward error bound (35) that calculates this bound without having to calculate an additional, computationally expensive higher precision solution, but rather operates through the numerical stability analysis of the solution provided by the method in question only.

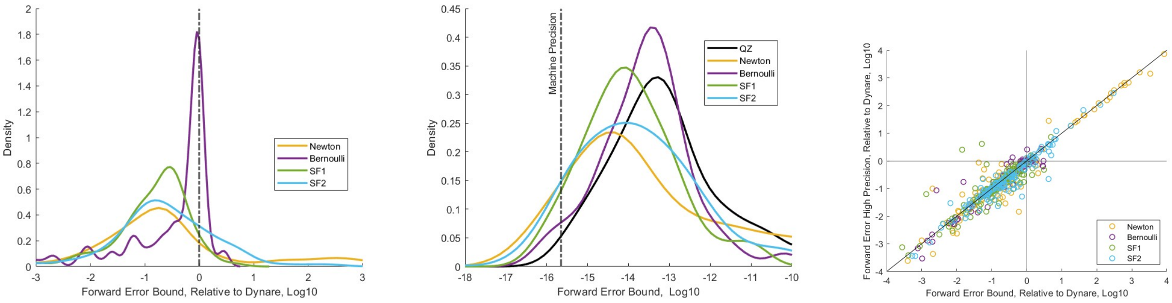


(A) FE Bound, Relative to Dynare (B) Computation Time, Relative to Dynare

FIGURE 5. Forward Errors, Computation Time and State Variables for MMB

Figure 5 continues the model-by-model comparison of the different algorithms' performance relative to QZ, but now with a focus on the effect of model size, measured by the number of state or backward-looking variables. The top left panel gives the accuracy of the different methods for the different models plotted against the dimension of the endogenous state. There is not a visually compelling correlation—although for very large models, the reduction and doubling algorithms like the Newton algorithm appear to perform better than QZ. For computational times the story is different: the right panel plots the computational time against the number of state variables and a clear downward trend or negative correlation in particular for the doubling algorithms is obvious. In sum, for the largest models in the MMB, our doubling algorithms provide about two orders of magnitude more accuracy at about one tenth the computation time.

To assess the potential for improving on solutions, we repeat the exercise, but now initialize with the solution provided by QZ, see the bottom half of table 3 for the results. The cyclic and logarithmic reductions are no longer included as they cannot operate with an arbitrary initialization. The baseline Newton method significantly improves the number of models it converges for. Note especially that, in contrast to the top half of the table when initialized at the zero matrix, now the SF1 doubling algorithm alongside the baseline Bernoulli method converges successfully to the unique stable solution in all models. Consistent with theorem 2, the SF2 doubling algorithm, however, does not and continues to converge or not for all but one of the same models as under the initialization with the zero matrix.



(A) Relative to Dynare (B) Forward Error Bound (C) High Precision v. Bound

FIGURE 6. Distribution of FE bound for the MMB, initialized at Dynare (QZ)

Both the Bernoulli and Newton methods run just a few iterations, and the later generally achieves a greater increase in accuracy albeit at a higher additional computation cost due to the computational intensity of the Newton step. The SF1 and SF2 doubling algorithms run through multiple iterations, ending up being faster and slightly more accurate than the Newton method at the median. Far more impressive here are the max or worst-case outcomes, with SF1 doubling at worst adding on an additional 7.5 times QZ computation cost and 4.9 times the forward error. The worst case Newton computation costs are an additional 840 times and Bernoulli 50+ thousand times the QZ initial time and the SF2 doubling algorithm has at worst a forward error bound about 300 times and Newton 1e9 times that of QZ. The results also highlight the effects of finite precision: numerical calculations generally introduce nonzero numerical errors, e.g. rounding errors, and if QZ's solution is already accurate up to machine precision, additional calculations with any algorithm are unlikely to increase and more likely to decrease accuracy. We conclude that the SF1 doubling algorithm ought to be preferred as a solution refinement method, usually providing significant accuracy gains at low additional computation costs that is robust even in the worst case relative to alternatives.

Figure 6, like figure 3 but now initialized at the QZ solution, provides an overview of the entire distribution of forward errors, the left relative to those from Dynare's QZ method and the middle in absolute terms. We display the results from the forward error bound and, again, the forward error relative to a high precision solution gives the same picture, as confirmed by the figure on the right. It is apparent that the baseline Bernoulli algorithm provides only a marginal improvement on the QZ solution. While this should be tempered with the observation that only two Bernoulli iterations were performed at the median, the Newton algorithm also only runs a few iterations. The doubling algorithms perform favorably, with both displaying left shifted distributions of error bounds relative to QZ. Taken together with the results from table 3, the SF1 doubling algorithm can be considered an ideal solution refinement algorithm across a wide variety of models.

6. Conclusion

We have introduced and developed doubling algorithms for solving linear DSGE models as alternatives to QZ methods, connecting them to the related reduction algorithms implemented, albeit silently, in Dynare. The doubling algorithms possess quadratic convergence rates, combining the advantages of Newton-based methods with the stable solution properties of Bernoulli methods. This paper demonstrates their practical value through extensive testing, particularly in comparison to the literature-standard QZ algorithm and in their ability to refine existing solutions for improved accuracy.

Our experiments, including applications to the Smets and Wouters (2007) model and the suite of models in the Macroeconomic Model Data Base (MMB), reveal that both doubling algorithms outperform QZ in accuracy and computational efficiency across a wide range of models. For the largest model in the MMB with thousands of state variables, doubling algorithms achieve solutions roughly an order of magnitude faster than QZ, highlighting their potential in computationally intensive settings, such as heterogeneous agent models like HANK. These findings underscore the relevance of doubling algorithms for large-scale macroeconomic modeling, particularly when speed and precision are critical.

While results across the MMB are generally favorable, outliers remain in terms of accuracy, computational time, and convergence. To address such variability, we extended the doubling algorithms to accept user-defined initializations, providing convincing evidence that the SF1 doubling algorithm in particular can reliably deliver low-cost, high-accuracy refinements of existing solutions. In scenarios where improving the accuracy of an initial solution is paramount, the SF1 doubling algorithm is the algorithm of choice, combining robustness, speed, and precision. The methods provide promising alternatives, particularly when working with large-scale or computationally intensive DSGE models.

References

ADJEMIAN, S., H. BASTANI, M. JUILLARD, F. MIHOUBI, G. PERENDIA, M. RATTO, AND S. VILLEMOT (2011): "Dynare: Reference Manual, Version 4," Dynare Working Papers 1, CEPREMAP.

ADVANPIX LLC. (2025): "Multiprecision Computing Toolbox for MATLAB," .

ANDERSON, B., AND J. MOORE (1979): *Optimal Filtering*. Prentice-Hall.

ANDERSON, E. W., E. R. MCGRATTAN, L. P. HANSEN, AND T. J. SARGENT (1996): "Mechanics of forming and estimating dynamic linear economies," vol. 1 of *Handbook of Computational Economics*, chap. 4, pp. 171–252. Elsevier.

BINI, D., AND B. MEINI (1996): "On the Solution of a Nonlinear Matrix Equation Arising in Queueing Problems," *SIAM Journal on Matrix Analysis and Applications*, 17(4), 906–926.

BINI, D. A., B. IANNAZZO, AND B. MEINI (2011): *Numerical solution of algebraic Riccati equations*. SIAM.

BINI, D. A., G. LATOUCHE, AND B. MEINI (2005): *Numerical Methods for Structured Markov Chains*. Oxford University Press.

BINI, D. A., AND B. MEINI (2023): "A defect-correction algorithm for quadratic matrix equations, with applications to quasi-Toeplitz matrices," *Linear and Multilinear Algebra*, pp. 1–16.

BINI, D. A., B. MEINI, AND F. POLONI (2008): "From Algebraic Riccati equations to unilateral quadratic matrix equations: old and new algorithms," in *Numerical Methods for Structured Markov Chains*, ed. by D. Bini, B. Meini, V. Ramaswami, M.-A. Remiche, and P. Taylor, vol. 7461 of *Dagstuhl Seminar Proceedings (DagSemProc)*, pp. 1–28. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

BLANCHARD, O. J., AND C. M. KAHN (1980): "The Solution of Linear Difference Models under Rational Expectations," *Econometrica*, 48(5), 1305–1311.

CHIANG, C.-Y., E. K.-W. CHU, C.-H. GUO, T.-M. HUANG, W.-W. LIN, AND S.-F. XU (2009): "Convergence analysis of the doubling algorithm for several nonlinear matrix equations in the critical case," *SIAM Journal on Matrix Analysis and Applications*, 31(2), 227–247.

CHIANG, C.-Y., H.-Y. FAN, AND W.-W. LIN (2010): "A structured doubling algorithm for discrete-time algebraic Riccati equations with singular control weighting matrices," *Taiwanese Journal of Mathematics*, 14(3A), 933–954.

GARDINER, J. D., M. R. WETTE, A. J. LAUB, J. J. AMATO, AND C. B. MOLER (1992): "Algorithm 705; a FORTRAN-77 Software Package for Solving the Sylvester Matrix Equation AXBT + CXDT = E," *ACM Trans. Math. Softw.*, 18(2), 232–238.

GUO, X.-X., W.-W. LIN, AND S.-F. XU (2006): "A structure-preserving doubling algorithm for nonsymmetric algebraic Riccati equation," *Numerische Mathematik*, 103(3), 393–412.

HAMMARLING, S., C. J. MUNRO, AND F. TISSEUR (2013): "An Algorithm for the Complete Solution of Quadratic Eigenvalue Problems," *ACM Transactions On Mathematical Software*, 39(3), 18:1–18:19.

HANSEN, L. P., AND T. J. SARGENT (2014): *Recursive Models of Dynamic Linear Economies*. Princeton University Press, Princeton.

HARVEY, A. C. (1990): *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

HEIBERGER, C., T. KLARL, AND A. MAUSSNER (2017): "On The Numerical Accuracy Of First-Order Approximate Solutions To DSGE Models," *Macroeconomic Dynamics*, 21(7), 1811–1826.

HIGHAM, N. J., AND H.-M. KIM (2000): "Numerical Analysis of a Quadratic Matrix Equation," *IMA Journal of Numerical Analysis*, 20, 499–519.

HUANG, T.-M., R.-C. LI, AND W.-W. LIN (2018): *Structure-Preserving Doubling Algorithms for Nonlinear Matrix Equations*. Society for Industrial and Applied Mathematics.

KÖHLER, M. (2024): "Matrix Equations PACKage – A Fortran library for the solution of Sylvester-like Matrix equations," Doi: 10.5281/zenodo.10016456, Zendo.

LAN, H., AND A. MEYER-GOHDE (2014): "Solvability of Perturbation Solutions in DSGE Models," *Journal of Economic Dynamics and Control*, 45, 366–388.

LATOUCHE, G., AND V. RAMASWAMI (1993): "A Logarithmic Reduction Algorithm for Quasi-Birth-Death Processes," *Journal of Applied Probability*, 30(3), 650–674.

MCGRATTAN, E. R. (1990): "Solving the Stochastic Growth Model by Linear-Quadratic Approximation," *Journal of Business ßEconomic Statistics*, 8(1), 41–44.

MEHRMANN, V., AND E. TAN (1988): "Defect correction method for the solution of algebraic Riccati equations," *IEEE transactions on automatic control*, 33(7), 695–698.

MEYER-GOHDE, A. (2023): "Numerical Stability Analysis of Linear DSGE Models - Backward Errors, Forward Errors and Condition Numbers," IMFS Working Paper Series 193, Goethe University Frankfurt, Institute for Monetary and Financial Stability.

——— (2024): "Solving Linear DSGE Models with Bernoulli Methods," *Computational Economics*, pp. 1–51.

MEYER-GOHDE, A., AND J. SAECKER (2024): "Solving linear DSGE models with Newton methods," *Economic Modelling*, 133, 106670.

MOLER, C. B., AND G. W. STEWART (1973): "An Algorithm for Generalized Matrix Eigenvalue Problems," *SIAM Journal on Numerical Analysis*, 10(2), 241–256.

POLONI, F. (2020): "Iterative and doubling algorithms for Riccati-type matrix equations: A comparative introduction," *GAMM-Mitteilungen*, 43(4).

SMETS, F., AND R. WOUTERS (2007): "Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach," *The American Economic Review*, 97(3), 586–606.

VILLEMOT, S. (2011): "Solving Rational Expectations Models at First Order: What Dynare Does," Dynare Working Papers 2, CEPREMAP.

WIELAND, V., E. AFANASYEVA, M. KUETE, AND J. YOO (2016): "New Methods for Macro-Financial Model Comparison and Policy Analysis," in *Handbook of Macroeconomics*, ed. by J. B. Taylor, and H. Uhlig, vol. 2 of *Handbook of Macroeconomics*, pp. 1241–1319.

WIELAND, V., T. CWIK, G. J. MÜLLER, S. SCHMIDT, AND M. WOLTERS (2012): "A new comparative approach to macroeconomic modeling and policy analysis," *Journal of Economic Behavior & Organization*, 83(3), 523–541.

Appendix: Proof of Theorem 2

We can show the statement by induction. For $k=0$ we have

$$\widehat{X}_0^\dagger = -AP_0 = X_0^\dagger - AP_0, \qquad \widehat{Y}_0^\dagger = -AP_0 - B = Y_0^\dagger - AP_0, \qquad \widehat{E}_0^\dagger = -C = E_0^\dagger, \qquad \widehat{F}_0^\dagger = -A = F_0^\dagger.$$

Further, assuming that the claim holds for an arbitrary $k \geq 0$ we have

$$\left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right) = \left(X_k^\dagger - AP_0 - Y_k^\dagger + AP_0\right) = \left(X_k^\dagger - Y_k^\dagger\right),$$

and therefore

$$\begin{aligned}
\widehat{E}_{k+1}^\dagger &= \widehat{E}_k^\dagger \left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right)^{-1} \widehat{E}_k^\dagger = E_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger = E_{k+1}^\dagger \\
\widehat{F}_{k+1}^\dagger &= \widehat{F}_k^\dagger \left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right)^{-1} \widehat{F}_k^\dagger = F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} F_k^\dagger = F_{k+1}^\dagger \\
\widehat{X}_{k+1}^\dagger &= \widehat{X}_k^\dagger - \widehat{F}_k^\dagger \left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right)^{-1} \widehat{E}_k^\dagger = X_k^\dagger - AP_0 - F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger = X_{k+1}^\dagger - AP_0 \\
\widehat{Y}_{k+1}^\dagger &= \widehat{Y}_k^\dagger + \widehat{E}_k^\dagger \left(\widehat{X}_k^\dagger - \widehat{Y}_k^\dagger\right)^{-1} \widehat{F}_k^\dagger = Y_k^\dagger - AP_0 + E_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} F_k^\dagger = Y_{k+1}^\dagger - AP_0.
\end{aligned}$$

Appendix: Proof of Theorem 3

For the proofs of parts (1) and (2) see Theorems 3.18 and 3.19 by Huang et al. (2018, pp. 35,37). To prove (3) first note that assumption 1 (and hence corollary 1) implies $\rho(P) \leq 1 \wedge \rho(P_d) < 1$.

We will show that in this case H_k converges quadratically to zero. To see this, note that using (33) of UQME (5) and it's dual UQME $0 = CP_d^2 + BP_d + A$, respectively, we receive

$$L_k = \mathscr{M}_k - H_k \mathscr{M}_k^2, \qquad\qquad H_k = \mathscr{N}_k - L_k \mathscr{N}_k^2 \tag{A1}$$

where $\mathscr{N}_k = \mathscr{N}^{2^k} = P_d^{2^k}$. Using (A1) to substitute L_k in the right equation of (A1) yields

$$H_k = \mathscr{N}_k - \mathscr{M}_k \mathscr{N}_k^2 + H_k \mathscr{M}_k^2 \mathscr{N}_k^2 \tag{A2}$$

Thus, for any sub-multiplicative matrix norm $\|.\|$ we obtain $\|H_k\| \leq \|\mathscr{N}_k\| + \|\mathscr{M}_k\| \|\mathscr{N}_k\|^2 + \|H_k\| \|\mathscr{M}_k\|^2 \|\mathscr{N}_k\|^2$. Since $\rho(P) \cdot \rho(P_d) = \rho(\mathscr{M}) \cdot \rho(\mathscr{N}) < 1$ defining $\epsilon_k = \|\mathscr{N}_k\| \|\mathscr{M}_k\|$ we know that $\lim_{k\to\infty} \epsilon_k = 0$. Hence, there is some sufficiently large k so that $\epsilon_k < 1$ and consequently $\|H_k\| \leq \frac{1+\epsilon_k}{1-\epsilon_k^2} \|\mathscr{N}_k\| \Rightarrow \lim_{k\to\infty} \|H_k\| = 0$. From the Gelfand's formula / the spectral radius theorem we also know $\lim_{k\to\infty} \|\mathscr{N}_k\|^{1/2^k} = \rho(\mathscr{N})$ so that

$$\|H_k\|^{1/2^k} \leq \left(\frac{1+\epsilon_k}{1-\epsilon_k^2}\right)^{1/2^k} \|\mathscr{N}_k\|^{1/2^k} \Rightarrow \lim_{k\to\infty} \|H_k\|^{1/2^k} = \rho(\mathscr{N}) \leq \rho(P_d).$$

Since $\rho(P_d) < 1$ we know that H_k converges quadratically to zero. Hence, we also know that there must be some sufficiently large k such that $\|\widehat{H}_{k-1}\| = \|H_0 \cdot \ldots \cdot H_{k-1}\| < 1$ and consequently $\|\widehat{H}_k\| \leq \|\widehat{H}_{k-1}\| \|H_k\| \leq \|H_k\|$. This means that $\widehat{H}_k$ also converges quadratically to zero, i.e., $\lim_{k\to\infty} \|\widehat{H}_k\|^{1/2^k} \leq \rho(\mathscr{N}) = \rho(P_d)$. Now rewrite the right equation of (33) to $P - \widehat{L}_k = \widehat{H}_k \mathscr{M}_{k+1}$ to yield $\|P - \widehat{L}_k\| \leq \|\widehat{H}_k\| \|\mathscr{M}_{k+1}\|$. The statement then follows from the Gelfand's formula / the spectral radius theorem as

$$\|P - \widehat{L}_k\|^{1/2^k} \leq \|\widehat{H}_k\|^{1/2^k} \|\mathscr{M}_{k+1}\|^{1/2^k} \Rightarrow \lim_{k\to\infty} \|P - \widehat{L}_k\|^{1/2^k} \leq \rho(P)\rho(P_d). \tag{A3}$$