SOLVING LINEAR DSGE MODELS WITH STRUCTURE PRESERVING DOUBLING METHODS

JOHANNES HUBER, ALEXANDER MEYER-GOHDE, AND JOHANNA SAECKER

ABSTRACT. This paper applies structure preserving doubling methods to solve the matrix quadratic underlying the recursive solution of linear DSGE models. We present and compare two Structure-Preserving Doubling Algorithms (SDAs) to other competing methods – the QZ method, a Newton algorithm, and an iterative Bernoulli approach – as well as the related cyclic and logarithmic reduction algorithms. Our comparison is completed using nearly 100 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 (2007) iteratively. We find that both SDAs perform very favorably relative to QZ, with generally more accurate solutions computed in less time. While we collect theoretical convergence results that promise quadratic convergence rates to a unique stable solution, the algorithms may fail to converge when there is a breakdown due to singularity of the coefficient matrices in the recursion. One of the proposed algorithms can overcome this problem by an appropriate (re)initialization. This SDA also performs particular well in refining solutions of different methods or from nearby parameterizations.

JEL classification codes: C61, C63, E17

Keywords: Numerical accuracy; DSGE; Solution methods

(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: February 12, 2024.

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".

1. Introduction

The major computational hurdle in the solution of linear DSGE models is the solution of the associated matrix quadratic equation - the current standard in the literature is to use a generalized Schur or QZ decomposition (Moler and Stewart, 1973; Golub and van Loan, 2013) to solve this equation. The applied mathematics literature has developed numerous different methods to solve quadratic matrix equations, but many of these have yet to be applied to DSGE models. We fill part of that gap, collecting and developing two versions of a Structure-Preserving Doubling Algorithm (SDA)[1] and applying them to the solution of linear DSGE models. We show that these methods cannot only be used to solve DSGE models successfully, but also that their convergence properties enable them to perform favorably relative to QZ-based methods. This is accomplished by our doubling algorithms combining the asymptotic quadratic convergence rate of, say, Meyer-Gohde and Saecker's (2022) Newton based methods and the convergence to the desired stable solution like Meyer-Gohde (2023b).

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, McGrattan, Hansen, and Sargent (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 as Anderson, McGrattan, Hansen, and Sargent (1996) apply 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, Bastani, Juillard, Mihoubi, Perendia, Ratto, and Villemot, 2011) (henceforth Dynare) includes and expands on this class of models, our approach here can be seen as an extension, specifically to Anderson, McGrattan, Hansen, and Sargent's (1996) work.[2] More closely related is the application to Riccati equations (see Poloni

[1] See Huang, Li, and Lin (2018) for a book length treatment.

[2] 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.

(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, Meini, and Poloni (2008) for example - both are quadratic equations in a matrix unknown, but with different structures. Chiang, Chu, Guo, Huang, Lin, and Xu (2009), however, presents explicit results for matrix polynomials with doubling methods - specifically structure preserving doubling methods, see Huang, Li, and Lin (2018) - and connects these algorithms to reduction algorithms (Latouche and Ramaswami's (1993) logarithmic and Bini and Meini's (1996) cyclic) that are undocumented algorithms available in Dynare.[3] 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[4], 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 (2022) Newton algorithms, and Meyer-Gohde's (2023b) Bernoulli methods following exactly the latter's experiments to ensure comparability. We begin by comparing the methods in the Smets and Wouters (2007) model of the US economy - both at the posterior mode 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 Meyer-Gohde and Saecker's (2022) Newton and Meyer-Gohde's (2023b) Bernoulli algorithms can

[3]They, as do others in the literature on doubling algorithms, link the matrix quadratic to Riccati equations in the context of quasi birth death models, whose matrices are subject to more strict assumptions than ours -e.g., nonnegative as components of a transition probability matrix, see also Poloni (2020).

[4]We use Dynare's (Adjemian, Bastani, Juillard, Mihoubi, Perendia, Ratto, and Villemot, 2011) implementation of the QZ method, documented in Villemot (2011), for comparison.

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 (2022) 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 Macroeconomic Model Data Base (MMB) (see Wieland, Cwik, Müller, Schmidt, and Wolters, 2012; Wieland, Afanasyeva, Kuete, and Yoo, 2016), both initializing with a zero matrix (or imply running 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). We find that the reduction and doubling methods provide useable alternatives to the standard QZ. 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 solution methods, one is particularly successful as a solution refinement algorithm. 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.

The remainder of the paper is organized as follows. Section 2 lays out the general DSGE model class. Section 3 presents the structure-preserving doubling algorithm. In section 4, we consider practical and theoretical aspects like the choice of initial value, solvability, accuracy and convergence. In section 5, we investigate the properties of the outlined algorithm using the suite of models from the MMB. Finally, section 6 concludes.

2. Problem Statement

The standard set of numerical solution packages for dynamic stochastic macroeconomic models[5] all analyze models that can generally be brought into the following nonlinear functional equation

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

The n_y-dimensional vector-valued 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 $y_t \in \mathbb{R}^{n_y}$ are a vector of size n_y ; and the vector of n_e exogenous shocks are contained in $\varepsilon_t \in \mathbb{R}^{n_e}$, where n_y and n_e are positive integers ($n_y, n_e \in \mathbb{N}$) and ε_t has a known mean zeros 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 then find an approximation. 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, be it analytically or numerically, and this steady state 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 the linearized model (3) is a linear solution in the form, following (2),

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

[5] E.g., Dynare (Adjemian, Bastani, Juillard, Mihoubi, Perendia, Ratto, and Villemot, 2011), Gensys (Sims, 2001), (Perturbation) AIM (Anderson and Moore, 1985; Anderson, Levin, and Swanson, 2006), Uhlig's Toolkit (Uhlig, 1999) and Solab (Klein, 2000)

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

Using our linear solution (4), it yields through substitution into (3) - and recognizing that the expectation $E_t[\varepsilon_{t+1}] = 0$ is known - the following two equations

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

With the former solution being quadratic with potentially multiple solutions, a selection criteria has to be used and generally a unique (semi) stable solution P is sought by the literature, that is a P with all its eigenvalues inside the open unit circle. 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 present two doubling algorithms to solve the UQME (5) for a unique (almost) stable solution P, which rely on the so-called First Standard Form (SF1) and Second Standard Form (SF2), respectively. Further we will show that the latter is closely related to the Cyclic and Logarithmic Reduction algorithm implemented in Dynare.

To illustrate doubling methods and build intuition, consider first the calculation of the geometric series

$$x = \sum_{j=0}^{\infty} \beta^j, \ |\beta| < 1 \tag{6}$$

The solution x can be expressed as the limit of the partial sum

$$x = \frac{1}{1-\beta} = \lim_{k\to\infty} x_k, \ x_k = \sum_{j=0}^{k} \beta^j \tag{7}$$

and x can be recovered by iterating on

$$x_k = 1 + \beta x_{k-1}, \qquad k \geq 0, \qquad x_{-1} = 0 \tag{8}$$

alternatively, we can use a doubling algorithm. Consider the $2k-1$'th partial sum

$$x_{2k-1} = \sum_{j=0}^{2k-1} \beta^j = \underbrace{1+\beta+\beta^2+\ldots+\beta^{k-1}}_{k \text{ terms}} + \underbrace{\beta^k+\ldots+\beta^{2k-1}}_{k \text{ terms}} = \sum_{j=0}^{k-1} \beta^j + \beta^k \sum_{j=0}^{k-1} \beta^j \quad (9)$$

Iterating on

$$w_{k+1} = w_k + \alpha_k w_k, \qquad \alpha_{k+1} = \alpha_k^2, \qquad k \geq 0, \qquad w_0 = 1, \qquad \alpha_0 = \beta \quad (10)$$

Gives for the first several terms

$$w_0 = \underbrace{1}_{x_0}, \qquad w_1 = w_0 + \alpha_0 w_0 = \underbrace{1+\beta}_{x_1}, \qquad \alpha_1 = \beta^2 = \beta^{2^1} \quad (11)$$

$$w_2 = w_1 + \alpha_1 w_1 = 1+\beta+\beta^2(1+\beta) = \underbrace{1+\beta+\beta^2+\beta^3}_{x_3}, \qquad \alpha_2 = \beta^4 = \beta^{2^2} \quad (12)$$

$$w_3 = w_2 + \alpha_2 w_2 = 1+\beta+\beta^2+\beta^3+\beta^4(1+\beta+\beta^2+\beta^3) = \underbrace{\sum_{j=0}^{7} \beta^j}_{x_7}, \quad \alpha_3 = \beta^8 = \beta^{2^3} \quad (13)$$

the relation $w_k = x_{2^k-1}$ and the factor 2^k gives the method its name. Clearly w_k will converge more quickly in k to x than x_k.

In terms of a vector space, we can define x_k via

$$\underbrace{\begin{pmatrix} 1 \\ x_k \end{pmatrix}}_{\mathcal{X}_k} = \underbrace{\begin{pmatrix} 1 & 0 \\ 1 & \beta \end{pmatrix}}_{S} \begin{pmatrix} 1 \\ x_{k-1} \end{pmatrix} \quad (14)$$

with $\begin{pmatrix} 1 & x_{-1} \end{pmatrix}' = \begin{pmatrix} 1 & 0 \end{pmatrix}'$. We can either iterate on the foregoing 2^k times to recover x_{2^k-1}, or look for a doubling approach

$$\mathcal{X}_{2^k-1} = S^{2^k} \mathcal{X}_{-1} = S^{2^{k-1}} S^{2^{k-1}} \mathcal{X}_{-1} \quad (15)$$

S is lower triangular and hence so is S^{2^k}. If we can find a doubling approach that preserves this structure, say,

$$S^{2^k} = \begin{pmatrix} 1 & 0 \\ e_k & f_k \end{pmatrix} \quad (16)$$

then we can reduce the difficulty of the problem significantly, as the structure enables us to define the recursion in the entries e_k and f_k instead of S^{2^k} in its

entirety. From this structure, it follows that

$$S^{2^k} = S^{2^{k-1}} S^{2^{k-1}} = \begin{pmatrix} 1 & 0 \\ e_{k-1} & f_{k-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ e_{k-1} & f_{k-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ e_{k-1} + f_{k-1} e_{k-1} & f_{k-1}^2 \end{pmatrix} \tag{17}$$

which gives the recursions for the structure preserving doubling algorithm as

$$f_k = f_{k-1}^2, \qquad e_k = e_{k-1} + f_{k-1} e_{k-1} \tag{18}$$

and again we have a doubling in $w_k = e_k + f_k w_{k-1}$ as $w_k = x_{2^k-1}$ with the initial conditions $w_{-1} = 0$, $f_0 = \beta$, and $e_0 = 1$. Notice that while the key theoretical insight is formulating a doubling approach to S^k, i.e., S^{2^k}, the key computational insight is the structure, here lower triangularity, that enables to find recursions in elements of S^{2^k}, namely e_k and f_k, instead of S^{2^k} in its entirety.

3.1. Matrix Quadratics, Pencils, QZ, and Doubling

To enable our doubling approach, we will first express the UQME in (5) as a subspace problem by forming the first companion linearization of the matrix quadratic problem (Hammarling, Munro, and Tisseur, 2013; Meyer-Gohde and Pigkou, 2023)

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

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 (19). Further note that the eigenvalues of $\mathscr{M}$ are a subset of the generalized eigenvalues of the matrix pencil $\mathscr{A} - \lambda\mathscr{B}$, i.e., $\text{eig}(\mathscr{M}) \subset \text{eig}(\mathscr{A}_0, \mathscr{B}_0)$.

Before we address solving (19), we will assume the conditions for the existence of the unique solvent P are fulfilled, i.e., Blanchard and Kahn's (1980) celebrated rank and order conditions.[6] We will make one more assumption to be able to prove

[6] Lan and Meyer-Gohde (2014) and Meyer-Gohde (2022) provide the conditions expressed in terms of the general class of multivariate models we consider here.

quadratic convergence later via the corresponding dual equation to the UQME (5) which is the quadratic equation above "in reverse"

$$0 = CP_d^2 + BP_d + A. \tag{20}$$

Throughout this paper, we assume with respect to P and P_d that the following statement is true.

Assumption 1. *There exist solvents P and P_d satisfying the UQMEs in (5) and (20), respectively, such that*

$$\rho(P) := \max_{\lambda \in \text{eig}(P)} |\lambda| \leq 1, \qquad \rho(P_d) := \max_{\lambda \in \text{eig}(P_d)} |\lambda| \leq 1, \qquad \rho(P) \cdot \rho(P_d) < 1.$$

So note that assumption 1 implies the Blanchard and Kahn (1980) *rank* and *order* conditions and is the usual assumption on P, see above. The condition on P_d in assumption 1 will provide sufficient conditions for quadratic convergence of the algorithms presented below.

The problem in (19) is numerically an eigenvalue problem and can thus be solved using the QZ or generalized Schur decompostion of Moler and Stewart (1973). We will derive the solution by working directly with the linear algebraic problem instead of dynamic model as is usually done. This should link the more familiar QZ with the doubling algorithms we will subsequently present. The decomposition provides unitary Q and Z and upper triangular S and T with $Q^* \mathscr{B} Z = S$ and $Q^* \mathscr{A} Z = T$ where the eigenvalues of the matrix pencil $P_{\mathscr{B}\mathscr{A}}(z) = \mathscr{B}z - \mathscr{A}$, $\rho(P_{\mathscr{B}\mathscr{A}}) = \rho(P_{ST}) = \{t_{ii}/s_{ii}, \text{ if } s_{ii} \neq 0; \infty, \text{ if } s_{ii} = 0; \emptyset, \text{ if } s_{ii} = t_{ii} = 0; i = 1, \ldots, 2n_y\}$, can be ordered arbitrarily to form

$$\begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \begin{bmatrix} w^s \\ w^u \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix} \begin{bmatrix} w^s \\ w^u \end{bmatrix} P \tag{21}$$

where $Z^* \begin{bmatrix} I & P' \end{bmatrix}' = \begin{bmatrix} w^{s\prime} & w^{u\prime} \end{bmatrix}'$. We assume the decomposition is ordered so that the unstable eigenvalues are in the lower right blocks of S and T (hence S_{22} and T_{22}), wherefor the lower block equation gives

$$T_{22} w^u = S_{22} w^u P \Rightarrow w^u = T_{22}^{-1} S_{22} w^u P \tag{22}$$

as the eigenvalues of $T_{22}^{-1}S_{22}$ are inside the unit circle and P is a (semi) stable solvent we can iterate

$$w^u = T_{22}^{-1}S_{22}w^uP = \left(T_{22}^{-1}S_{22}\right)^2 w^uP^2 = \left(T_{22}^{-1}S_{22}\right)^j w^uP^j \underset{j\to\infty}{\to} 0 \tag{23}$$

for a unit root stable P. Using the definition of w^u

$$0 = w^u = Z_{21}^* + Z_{22}^*P \Rightarrow P = -{Z_{22}^*}^{-1}Z_{21}^* = Z_{21}Z_{11}^{-1} \tag{24}$$

where * indicates the complex conjugation of Z that delivers its inverse by virtue of it being a unitary matrix. The equivalence ${Z_{22}^*}^{-1}Z_{21}^* = -Z_{21}Z_{11}^{-1}$ follow from the properties of unitary matrices and $Z_{21}Z_{11}^{-1} = Q_{11}S_{11}^{-1}T_{11}Q_{11}^{-1}$ from the first block rows of $\mathscr{A}$ and $\mathscr{B}$ in (19) and upper triangularity of S and T. From $Q_{11}S_{11}^{-1}T_{11}Q_{11}^{-1}$, it follows that the recursion in P is indeed stable from the ordering of the eigenvalues above, i.e. the eigenvalues of the upper left block of the generalized Schur decomposition, $det(S_{11}\lambda - T_{11}) = 0$, are inside the unit circle.[7] So the QZ decomposition applied to our matric pencil will recover the unique (semi) stable solvent P if it exists consistent with our assumption 1. Importantly, we did not solve the problem by working directly with the pencil $P_{\mathscr{B}\mathscr{A}}(z) = \mathscr{B}z - \mathscr{A}$ but first transformed the problem unitarily with Q and Z.

Analogous to our geometric series, we would also like to find a way to approach the matrix pencil problem here via a doubling approach. Following Guo, Lin, and Xu (2006) a transformation $\widehat{\mathscr{A}} - \lambda\widehat{\mathscr{B}}$ of a pencil $\mathscr{A} - \lambda\mathscr{B}$ is called a doubling transform if

$$\widehat{\mathscr{A}} = \overline{\mathscr{A}}\mathscr{A}, \quad \widehat{\mathscr{B}} = \overline{\mathscr{B}}\mathscr{B} \tag{25}$$

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

$$\text{rank}\left(\begin{bmatrix}\overline{\mathscr{A}} & \overline{\mathscr{B}}\end{bmatrix}\right) = 2n_y, \quad \begin{bmatrix}\overline{\mathscr{A}} & \overline{\mathscr{B}}\end{bmatrix}\begin{bmatrix}\mathscr{A} \\ -\mathscr{B}\end{bmatrix} = 0 \tag{26}$$

[7] Meyer-Gohde (2023a) derives this representation for the recursive solution in y_t directly. The parallel derivation here emphasises the equivalence of solving the matrix quadratic in P or for a recursive solution in y_t.

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

Such a doubling transformation is eigenspace preserving and eigenvalue squaring ("doubling") following Guo, Lin, and Xu (2006, Theorem 2.1) or Huang, Li, and Lin (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*

$$\mathscr{A}\mathscr{X} = \mathscr{B}\mathscr{X}\mathscr{M} \tag{27}$$

from (19)

$$\widehat{\mathscr{A}}\mathscr{X} = \widehat{\mathscr{B}}\mathscr{X}\mathscr{M}^2 \tag{28}$$

Proof. Starting with (19) and multiplying with $\overline{\mathscr{A}}$ gives

$$\mathscr{A}\mathscr{X} = \mathscr{B}\mathscr{X}\mathscr{M} \quad \rightarrow \quad \overline{\mathscr{A}}\mathscr{A}\mathscr{X} = \overline{\mathscr{A}}\mathscr{B}\mathscr{X}\mathscr{M} \quad \rightarrow \quad \widehat{\mathscr{A}}\mathscr{X} = \overline{\mathscr{A}}\mathscr{B}\mathscr{X}\mathscr{M} \tag{29}$$

From $\begin{bmatrix}\overline{\mathscr{A}} & \overline{\mathscr{B}}\end{bmatrix}\begin{bmatrix}\mathscr{A} \\ -\mathscr{B}\end{bmatrix} = 0$ it follows that $\overline{\mathscr{A}}\mathscr{B} = \overline{\mathscr{B}}\mathscr{A}$

$$\widehat{\mathscr{A}}\mathscr{X} = \overline{\mathscr{A}}\mathscr{B}\mathscr{X}\mathscr{M} \qquad \rightarrow \qquad \widehat{\mathscr{A}}\mathscr{X} = \overline{\mathscr{B}}\mathscr{A}\mathscr{X}\mathscr{M} \tag{30}$$

Then substituting $\mathscr{B}\mathscr{X}\mathscr{M}$ for $\mathscr{A}\mathscr{X}$ following (19) on the right hand side of the foregoing gives

$$\widehat{\mathscr{A}}\mathscr{X} = \overline{\mathscr{B}}\mathscr{A}\mathscr{X}\mathscr{M} \quad \rightarrow \quad \widehat{\mathscr{A}}\mathscr{X} = \overline{\mathscr{B}}\mathscr{B}\mathscr{X}\mathscr{M}\mathscr{M} \quad \rightarrow \quad \widehat{\mathscr{A}}\mathscr{X} = \widehat{\mathscr{B}}\mathscr{X}\mathscr{M}^2 \tag{31}$$

where we completed the proof by recalling the definition of $\widehat{\mathscr{B}}$ □

Following this theorem, we obtain a doubling algorithm for (19) by iterating on

$$\underbrace{\widehat{\mathscr{A}_k}}_{\mathscr{A}_{k+1}} = \overline{\mathscr{A}}_k\mathscr{A}_k, \quad \underbrace{\widehat{\mathscr{B}_k}}_{\mathscr{B}_{k+1}} = \overline{\mathscr{B}}_k\mathscr{B}_k \tag{32}$$

initializing with $\mathscr{A}$ and $\mathscr{B}$ as

$$\mathscr{A}\mathscr{X} = \mathscr{B}\mathscr{X}\mathscr{M} \tag{33}$$

$$\mathscr{A}_1\mathscr{X} = \mathscr{B}_1\mathscr{X}\mathscr{M}^2 \tag{34}$$

$$\mathscr{A}_2\mathscr{X} = \mathscr{B}_2\mathscr{X}\mathscr{M}^4 \tag{35}$$

$$\mathscr{A}_k\mathscr{X} = \mathscr{B}_k\mathscr{X}\mathscr{M}^{2^k} \tag{36}$$

As above for the geometric series, we seek a structure in $\mathscr{A}$ and $\mathscr{B}$ such that we calculate $\mathscr{A}_k \to \mathscr{A}_{k+1}$ and $\mathscr{B}_k \to \mathscr{B}_{k+1}$ by recursions in elements (here we will settle for submatrices). In the following two section, we provide exactly such recursions, First Standard Form and Second Standard Form. Both require we rearrange our pencil $\mathscr{A} - \lambda\mathscr{B}$ to conform to the respective structures of the two recursions, as we now show.

3.2. First Standard Form

Assuming that B is non-singular we receive the primal problem in SF1

$$\mathscr{A}_0\mathscr{X} = \mathscr{B}_0\mathscr{X}\mathscr{M}, \tag{37}$$

with

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

multiplying (19) from the left by S. Note that (19) and (37) are equivalent in the sense that pencils $\mathscr{A} - \lambda\mathscr{B}$ and $\mathscr{A}_0 - \lambda\mathscr{B}_0$ share the same set of generalized eigenvalues, i.e., $\operatorname{eig}(\mathscr{A}, \mathscr{B}) = \operatorname{eig}(\mathscr{A}_0, \mathscr{B}_0)$.

The SDA for SF1 recursively computes sequences $\{\mathscr{A}_k\}_{k=0}^{\infty}$, $\{\mathscr{B}_k\}_{k=0}^{\infty}$ with

$$\mathscr{A}_{k+1} = \begin{pmatrix} E_{k+1} & 0 \\ -X_{k+1} & I \end{pmatrix} := \begin{pmatrix} E_k(I - Y_kX_k)^{-1} & 0 \\ -F_k(I - X_kY_k)^{-1}X_k & I \end{pmatrix} \mathscr{A}_k, \tag{38}$$

$$\mathscr{B}_{k+1} = \begin{pmatrix} I & -Y_{k+1} \\ 0 & F_{k+1} \end{pmatrix} := \begin{pmatrix} I & -E_k(I - Y_kX_k)^{-1}Y_k \\ 0 & F_k(I - X_kY_k)^{-1} \end{pmatrix} \mathscr{B}_k, \tag{39}$$

such that

$$\mathscr{A}_k\mathscr{X} = \mathscr{B}_k\mathscr{X}\mathscr{M}^{2^k}. \tag{40}$$

A key feature here is that (40) retains SF1 for all $k \in \mathbb{N}$. Algorithm 1 summarizes the SDA for the SF1. The intuition here is that under some fairly general pre-conditions, e.g., Assumption 1, the term $\mathscr{B}_k \mathscr{X} \mathscr{M}^{2^k}$ on the right-hand side of (40) converges to zero for $k \to \infty$, so that consequently X_k converges to P.

Algorithm 1: Structure-Preserving Doubling Algorithm (SF1)

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

Set X_0, Y_0, E_0, F_0 according to

$$X_0 = -B^{-1}C, \qquad Y_0 = -B^{-1}A, \qquad E_0 = -B^{-1}C, \qquad F_0 = -B^{-1}A$$

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

Set $E_{k+1} = E_k (I - Y_k X_k)^{-1} E_k$

Set $F_{k+1} = F_k (I - X_k Y_k)^{-1} F_k$

Set $X_{k+1} = X_k + F_k (I - X_k Y_k)^{-1} X_k E_k$

Set $Y_{k+1} = Y_k + E_k (I - Y_k X_k)^{-1} Y_k F_k$

Advance $k = k + 1$

end

Return: X_k

3.3. Second Standard Form

As an alternative to Algorithm 1, Chiang, Chu, Guo, Huang, Lin, and Xu (2009) show that the UQME (5) can also be solved using a doubling algorithm based on the eigenvalue problem

$$\mathscr{A}_0^\dagger \mathscr{X}^\dagger = \mathscr{B}_0^\dagger \mathscr{X}^\dagger \mathscr{M}, \tag{41}$$

with

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

which satisfies the so-called Second Standard Form (SF2). Obviously, again any P satisfying (5) is also a solution of (41). 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., $\operatorname{eig}(\mathscr{A},\mathscr{B})=\operatorname{eig}(\mathscr{A}_0^\dagger,\mathscr{B}_0^\dagger)$.

Similar to Algorithm 1 the SDA for SF2 recursively computes sequences $\{\mathscr{A}_k^\dagger\}_{k=0}^\infty$, $\{\mathscr{B}_k^\dagger\}_{k=0}^\infty$ with

$$\mathscr{A}_{k+1}^\dagger=\begin{pmatrix}E_{k+1}^\dagger & 0\\ -X_{k+1}^\dagger & I\end{pmatrix}:=\begin{pmatrix}E_k^\dagger\left(X_k^\dagger-Y_k^\dagger\right)^{-1} & 0\\ F_k^\dagger\left(X_k^\dagger-Y_k^\dagger\right)^{-1} & I\end{pmatrix}\mathscr{A}_k, \tag{42}$$

$$\mathscr{B}_{k+1}^\dagger=\begin{pmatrix}-Y_{k+1}^\dagger & I\\ -F_{k+1}^\dagger & 0\end{pmatrix}:=\begin{pmatrix}I & E_k^\dagger\left(X_k^\dagger-Y_k^\dagger\right)^{-1}\\ 0 & F_k^\dagger\left(X_k^\dagger-Y_k^\dagger\right)^{-1}\end{pmatrix}\mathscr{B}_k, \tag{43}$$

such that

$$\mathscr{A}_k^\dagger\mathscr{X}^\dagger=\mathscr{B}_k^\dagger\mathscr{X}^\dagger\mathscr{M}^{2^k}. \tag{44}$$

We summarize the SDA for the SF2 in Algorithm 2. Note that compared to Algorithm 1 the sequence $\{X_k^\dagger\}_{k=0}^\infty$ now converges to AP instead of P. However, in case of convergence we may obtain an P approximately as $-(X_k^\dagger+B)^{-1}C$.

3.4. Cyclic and Logarithmic Reduction

SDAs generate a sequence of matrix pencils, in each step squaring the corresponding eigenvalues. In contrast, Bini and Meini's (1996) Cyclic Reduction and Latouche and Ramaswami's (1993) Logarithmic Reduction, that are already implemented in Dynare, 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 illustrate the links to the SDA of SF2.[8]

[8]For a comprehensive textbook treatment see Bini, Iannazzo, and Meini (2011, Chapter 5.2) and Bini, Latouche, and Meini (2005, Chapter 7)

Algorithm 2: Structure-Preserving Doubling Algorithm (SF2)

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

Set $X_0^\dagger$, $Y_0^\dagger$, $E_0^\dagger$, $F_0^\dagger$ according to

$$X_0^\dagger = 0, \qquad Y_0^\dagger = -B, \qquad E_0^\dagger = -C, \qquad F_0^\dagger = -A$$

While *criterion*$(X_k^\dagger) > \epsilon$ **do**

Set $E_{k+1}^\dagger = E_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger$

Set $F_{k+1}^\dagger = F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} F_k^\dagger$

Set $X_{k+1}^\dagger = X_k^\dagger - F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger$

Set $Y_{k+1}^\dagger = Y_k^\dagger + E_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} F_k^\dagger$

Advance $k = k + 1$

end

Return: $-(X_k^\dagger + B)^{-1} C$

The 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

$$A_{k+1} = -A_k\, B_k^{-1}\, A_k, \qquad A_0 = A, \tag{45}$$

$$B_{k+1} = B_k - A_k\, B_k^{-1}\, C_k - C_k\, B_k^{-1}\, A_k, \qquad B_0 = B, \tag{46}$$

$$C_{k+1} = -C_k\, B_k^{-1}\, C_k, \qquad C_0 = C, \tag{47}$$

$$\widehat{B}_{k+1} = \widehat{B}_k - A_k\, B_k^{-1}\, C_k \qquad \widehat{B}_0 = B. \tag{48}$$

Using a divide-and-conquer strategy one can show that the Cyclic Reduction defines a sequence of UQMEs with

$$0 = A_k \mathscr{M}_k^2 + B_k \mathscr{M}_k + C_k. \tag{49}$$

where $\mathscr{M}_k = \mathscr{M}^{2^k}$. Moreover, we may state that

$$0 = A_k \mathscr{M}_k \mathscr{M} + \widehat{B}_k \mathscr{M} + C, \tag{50}$$

To see this note that both (49) and (50) will clearly hold for $k = 0$. Further, assuming that (49) and (50) hold for an arbitrary $k \geq 0$, we may multiply (49) from the right by $\mathcal{M}$, $\mathcal{M}_k$ and $\mathcal{M}_k^2$, respectively, and receive

$$0 = A_k \mathcal{M}_k^2 \mathcal{M} + B_k \mathcal{M}_k \mathcal{M} + C_k \mathcal{M}, \tag{51}$$

$$0 = A_k \mathcal{M}_k^3 + B_k \mathcal{M}_k^2 + C_k \mathcal{M}_k, \tag{52}$$

$$0 = A_k \mathcal{M}_k^4 + B_k \mathcal{M}_k^3 + C_k \mathcal{M}_k^2. \tag{53}$$

If we now add $C_k \mathcal{M}_k$, $A_k \mathcal{M}_k \mathcal{M}$, and $A_k \mathcal{M}_k^3$ to (49), (51), and (53) multiplied by $-C_k B_k^{-1}$, $-A_k B_k^{-1}$, and $-A_k B_k^{-1}$, respectively, we get

$$C_k \mathcal{M}_k = -C_k B_k^{-1} A_k \ \mathcal{M}_k^2 - C_k B_k^{-1} C_k, \tag{54}$$

$$A_k \mathcal{M}_k \mathcal{M} = -A_k B_k^{-1} A_k \ \mathcal{M}_k^2 \mathcal{M} - A_k B_k^{-1} C_k \ \mathcal{M}, \tag{55}$$

$$A_k \mathcal{M}_k^3 = -A_k B_k^{-1} A_k \ \mathcal{M}_k^4 - A_k B_k^{-1} C_k \ \mathcal{M}_k^2. \tag{56}$$

Finally, substituting (54) and (56) in (52) as well as (55) in (50)

$$0 = A_{k+1} \ \mathcal{M}_{k+1}^2 + B_{k+1} \ \mathcal{M}_{k+1} C_{k+1}, \tag{57}$$

$$0 = A_{k+1} \ \mathcal{M}_{k+1} \mathcal{M} + \widehat{B}_{k+1} \mathcal{M} + C. \tag{58}$$

Hence, (49) and (50) hold in $k+1$ as well. Furthermore, assuming that $\lim_{k\to\infty} \widehat{B}_k^{-1}$ exists and that $\lim_{k\to\infty} \widehat{B}_k^{-1} A_k \mathcal{M}_k \mathcal{M} = 0$ we can express the solvent P as

$$P = -\lim_{k\to\infty} \widehat{B}_k^{-1} C. \tag{59}$$

As pointed out by Bini, Iannazzo, and Meini (2011, pp. 167) and Chiang, Chu, Guo, Huang, Lin, and Xu (2009, pp. 236) the SDA for SF2 is connected to the Cyclic Reduction (Algorithm 3). In detail, we can show via induction that

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

for all $k \geq 0$. Hence, Algorithms 2 and 3 are theoretically equivalent.

Another algorithm already implemented in Dynare is the Logarithmic Reduction by Latouche and Ramaswami (1993), which uses the same divide-and-conquer strategy as the Cyclic Reduction to obtain $\{H_k\}_{k=0}^\infty$, $\{L_k\}_{k=0}^\infty$, $\{\widehat{H}_k\}_{k=0}^\infty$, and $\{\widehat{L}_k\}_{k=0}^\infty$

Algorithm 3: Cyclic Reduction

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

Set A_0, B_0, C_0, and $\widehat{B}_0$ according to

$$A_0 = A, \qquad B_0 = B, \qquad C_0 = C, \qquad \widehat{B}_0 = B$$

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

Set $A_{k+1} = -A_k B_k^{-1} A_k$

Set $B_{k+1} = B_k - A_k B_k^{-1} C_k - C_k B_k^{-1} A_k$

Set $C_{k+1} = -C_k B_k^{-1} C_k$

Set $\widehat{B}_{k+1} = \widehat{B}_k - A_k B_k^{-1} C_k$

Advance $k = k + 1$

end

Return: $-\widehat{B}_k^{-1} C_0$

with

$$H_{k+1} = (I - H_k L_k - L_k H_k)^{-1} H_k^2, \qquad H_0 = -B^{-1} A \tag{60}$$

$$L_{k+1} = (I - H_k L_k - L_k H_k)^{-1} L_k^2, \qquad L_0 = -B^{-1} C, \tag{61}$$

$$\widehat{H}_{k+1} = \widehat{H}_k H_{k+1}, \qquad \widehat{H}_0 = -B^{-1} A, \tag{62}$$

$$\widehat{L}_{k+1} = \widehat{L}_k + \widehat{H}_k L_{k+1}, \qquad \widehat{L}_0 = -B^{-1} C, \tag{63}$$

that define a sequence of UQMEs with

$$0 = H_k \mathscr{M}_k^2 - \mathscr{M}_k + L_k. \tag{64}$$

Furthermore, analogously to (50) we receive

$$0 = \widehat{H}_k \mathscr{M}_{k+1} - \mathscr{M} + \widehat{L}_k. \tag{65}$$

The Logarithmic Reduction is connected to the Cyclic Reduction. As pointed out by Bini, Latouche, and Meini (2005, Theorem 7.5), we may show via induction that

for all $k \geq 0$

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

so that (64) follows directly from multiplying (49) from the left by $-B_k^{-1}$. Consequently, there is also a link between Algorithm 2 and the Logarithmic Reduction (Algorithm 4). In detail, we get for all $k \geq 0$ that

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

Note that (65) also follows by induction, where in $k = 0$ we get (65) directly from multiplying (5) from the left by B^{-1}. Now, assuming that (65) holds for an arbitrary $k \geq 0$, we get

$$\begin{aligned} 0 &= \widehat{H}_k \mathscr{M}_{k+1} - \mathscr{M} + \widehat{L}_k, \\ &= \widehat{H}_k H_{k+1} \mathscr{M}_{k+1}^2 + \widehat{H}_k L_{k+1} - \mathscr{M} + \widehat{L}_k, \\ &= \widehat{H}_{k+1} \mathscr{M}_{k+1}^2 - \mathscr{M} + \widehat{L}_{k+1}, \end{aligned} \tag{66}$$

where we can use the fact that from (64) follows that $\mathscr{M}_{k+1} = H_{k+1}\mathscr{M}_{k+1}^2 + L_{k+1}$. Similar to the Cyclic Reduction we can express the stable solvent of the the UQME (5) as

$$P = \lim_{k \to \infty} \widehat{L}_k, \tag{67}$$

assuming $\lim_{k \to \infty} \widehat{H}_k \mathscr{M}_{k+1} = 0$.

We display the Cyclic and the Logarithmic Reduction in Algorithms 3 and 4. Summarizing, we can state that compared to the doubling algorithm, the reduction algorithms follow a similar idea by squaring the eigenvalues of matrix polynomials. Beyond that, abstracting from numerical inaccuracies, we find that for a given number of iterations SF2 in algorithm 2 and the Cyclic Reduction in algorithm 3 will deliver identical approximation to P. While the Cyclic and Logarithmic Reduction generate interchangeable sequences of UQMEs ((49) and (64)), the two algorithms differ in the way in which they recover the approximation to P ((50) and (65)). 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 approximations to P.

Algorithm 4: Logarithmic Reduction

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

Set $L_0, H_0, \widehat{L}_0, \widehat{H}_0$ according to

$$L_0 = -B^{-1}C, \qquad H_0 = -B^{-1}A, \qquad \widehat{L}_0 = -B^{-1}C, \qquad \widehat{H}_0 = -B^{-1}A$$

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

Set $U_k = (I - H_k L_k - L_k H_k)$

Set $L_{k+1} = U_k^{-1} L_k^2$

Set $H_{k+1} = U_k^{-1} H_k^2$

Set $\widehat{L}_{k+1} = \widehat{L}_k + \widehat{H}_k L_{k+1}$

Set $\widehat{H}_{k+1} = \widehat{H}_k H_{k+1}$

Advance $k = k + 1$

end

Return: $\widehat{L}_k$

4. Theoretical and Practical Considerations

In this section we present theoretical and practical considerations relating to the doubling methods. In particular we will address the convergence of the algorithms, the ability to adapt the algorithms to accept initializations for the solution P, and our measure of accuracy.

4.1. 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 2 (Convergence)**.** *Suppose P and P_d exist and satisfy Assumption 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,* $\limsup_{k\to\infty} \|X_k - P\|^{1/2^k} \leq \rho(P)\cdot\rho(P_d)$.

(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. □

Note that Theorem 2 also applies to the cyclic reduction presented in Algorithm 3, since it is equivalent to the SDA for SF2. Consequently, under Assumption 1 Algorithms 1 to 4 will all converge quadratically to the unique (semi) stable solution P.

4.2. Initial guess

A disadvantage of SDAs is that numerical inaccuracies can propagate from iteration to iteration, e.g., if 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 DAREs, one can solve the associated DAREs of the approximation error to increase the overall accuracy. Following this idea Bini and Meini (2023), show how to incorporate an initial guess to Algorithm 1 by means of an equivalence transformation of the pencil $\mathscr{A} - \lambda\mathscr{B}$. Huang, Li, and Lin (2018) discuss similar transformations for algebraic Riccati equations (AREs) in general.

In detail, we can introduce an initial guess by transforming the eigenvalue problem (19) to

$$\widehat{\mathscr{A}}\,\widehat{\mathscr{X}} = \widehat{\mathscr{B}}\,\widehat{\mathscr{X}}\,\mathscr{M} \tag{68}$$

with

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

where P_0 is the initial guess such that $P = \hat{P} + P_0$. Assuming that $B + AP_0$ has full rank we then may 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.,

$$\widehat{\mathscr{A}_0}\,\widehat{\mathscr{X}} = \widehat{\mathscr{B}}_0\,\widehat{\mathscr{X}}\,\mathscr{M}, \tag{69}$$

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 5 summarizes the SDA for SF1 based on an initial guess P_0. Note that a good guess P_0 should increase the probability that $B + AP_0$ is well-conditioned, since under Assumptions 1 we know that $B + AP$ has full rank (see Lan and Meyer-Gohde, 2014). In comparison to Algorithm 1 to 3 which are only applicable for non-singular B, Algorithm 5 can handle situations where B is singular, provided we can determine a matrix P_0 such that $B + AP_0$ is non-singular.[9]

[9] Chiang, Fan, and Lin (2010) use a similar technique to solve DAREs with singular transition matrices.

Algorithm 5: Structure-Preserving Doubling Algorithm (SF1) - with initial guess

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

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

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

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

Advance $k = k + 1$

end

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

Following the idea of Bini and Meini (2023) and applying it to Chiang, Chu, Guo, Huang, Lin, and Xu's (2009) Algorithm 2, we take again P_0 as the initial guess such that $P = \widehat{P} + P_0$ and insert this into the UQME (5)

$$0 = A\left(\widehat{P} + P_0\right)^2 + B\left(\widehat{P} + P_0\right) + C = A\widehat{P}^2 + A\widehat{P}P_0 + (AP_0 + B)\widehat{P} + AP_0^2 + BP_0 + C \tag{70}$$

This can be written as (41) 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 6 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 5 to the SDA for SF2 in Algorithm 2, the algorithm will deliver the same approximation to P regardless of P_0. We summarize this in the following

Algorithm 6: Structure-Preserving Doubling Algorithm (SF2) - with initial guess

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

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

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

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

Advance $k = k + 1$

end

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

theorem following Huang, Li, and Lin (2018, Theorem 3.32) and the surrounding discussion.

Theorem 3. *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 may state that*

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

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

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

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

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

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

and therefore

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

□

So 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 both not novel in the sense that they are known for Riccati equations, what is new is the presentation and proof for the specific case of our UQME. That we can provide a version of SF1 in Algorithm 5 that can operate 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 that is being sought. We will confirm this and the result from Theorem 3 that Algorithm 6 does not possess the same potential.

4.3. Accuracy

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

$$\underbrace{\frac{\left\|P-\hat{P}\right\|_F}{\|P\|_F}}_{\text{Forward Error}} \leq \underbrace{\frac{\left\|H_{\hat{P}}^{-1}\text{vec}\left(R_{\hat{P}}\right)\right\|_2}{\left\|\hat{P}\right\|_F}}_{\text{Forward Error Bound 1}} \leq \underbrace{\left\|H_{\hat{P}}^{-1}\right\|_2 \frac{\left\|R_{\hat{P}}\right\|_F}{\left\|\hat{P}\right\|_F}}_{\text{Forward Error Bound 2}} \tag{71}$$

where the residual of the UQME is $R_{\hat{P}} = A\hat{P}^2 + B\hat{P} + C$ and $H_{\hat{P}} = I_{n_y} \otimes \left(A\hat{P} + B\right) + \hat{P}' \otimes A$. A key component in assessing the conditioning of the problem following is Stewart's (1971) separation function, see also Kågström (1994), Kågström and Poromaa (1996), and Chen and Lv (2018), given by

$$\begin{aligned}
\text{sep}\left[\left(A, A\hat{P}+B\right), \left(I, -\hat{P}\right)\right] &= \min_{\|X\|_F=1} \left\|AX\hat{P} + \left(A\hat{P}+B\right)X\right\|_F && (72)\\
&= \min_{\|\text{vec}(X)\|_2=1} \left\|H_{\hat{P}}\text{vec}(X)\right\|_2 && (73)\\
&= \sigma_{\min}\left(H_{\hat{P}}\right) \leq \min\left|\lambda\left(A, A\hat{P}+B\right) - \lambda\left(\hat{P}\right)\right| && (74)
\end{aligned}$$

where $\lambda\left(A, A\hat{P}+B\right)$ is the set of (generalized) eigenvalues or the spectrum of the pencil $\left(A, A\hat{P}+B\right)$ (hence $\lambda\left(\hat{P}\right)$ is thus the spectrum of $\hat{P}$). As emphasized by Meyer-Gohde (2023a) the separation between the two pencils - the smallest singular value of $H_{\hat{P}}$ - extends the conditioning number from standard linear equations to structured problems like our UQME. 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 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 backward error, like the condition number for a standard linear system, and Meyer-Gohde (2023a) shows that this backward error can also differ arbitrarily from the residual, another frequent measure used in the literature. The tighter bound accounts for the structure of the problem more carefully, considering the linear operator $H_{\hat{P}}$ and

the residual $R_{\hat{P}}$ jointly, but for larger models can be computationally prohibitive. Hence the looser bound, besides linking difficulties in alternative measures via the separation and backward errors directly, is also practical, albeit pessimistic, for large scale DSGE models.

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) (see Wieland, Cwik, Müller, Schmidt, and Wolters, 2012; Wieland, Afanasyeva, Kuete, and Yoo, 2016).[10] 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. Additionally, these are exactly the experiments run in Meyer-Gohde and Saecker (2022), Meyer-Gohde (2023b), and Binder and Meyer-Gohde (2024) to maximize the comparability with all the different algorithms compared there. We will focus here on comparing our algorithms above with Dynare's QZ-based method,[11] Dynare's Cyclic and Logarithmic reduction methods, the baseline Newton method from Meyer-Gohde and Saecker (2022), and the baseline Bernoulli method from Meyer-Gohde (2023b).[12] 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.

[10] A model comparison initiative at the Institute for Monetary and Financial Stability (IMFS), see http://www.macromodelbase.com.

[11] See Villemot (2011).

[12] Additionally, note that 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 and structuring the matrix quadratic according to the classification of "static", "purely forward", "purely backward looking", and "mixed" variables. The details are in the online appendix and Meyer-Gohde and Saecker (2022). We take Dynare's reduction algorithms "as is".

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 following log-linearized monetary policy rule is particularly important, as we will compare the accuracy of different solution methods when solving under alternate, but nearby parameterizations - specifically parameter values in the following Taylor rule

$$r_t = \rho r_{t-1} + (1-\rho)(r_\pi \pi_t + r_Y (y_t - y_t^p)) + r_{\Delta y}((y_t - y_t^p) - (y_{t-1} - y_{t-1}^p)) + \varepsilon_t^r, \tag{75}$$

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. The rule is completed with an AR(1) monetary policy shock, ε_t^r, which is assumed to have an iid normal innovation. The model parameters are estimated using Bayesian methods on seven macroeconomic time series from the US economy to estimate, the resulting posterior produces out-of-sample forecasting performance that is inline with reduced form (B)VAR models.

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 of Meyer-Gohde and Saecker (2022) that fails to converge to the stable solvent (there is no guarantee that this algorithm will converge to a particular solvent, see their discussion and the varieties and extensions of this baseline algorithm that also perform more favorably). The baseline Bernoulli algorithm of Meyer-Gohde (2023b) 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

Method	Relative Performance		Forward Errors		Iterations
	Run Time	Max Abs. Diff.	Bound 1	Bound 2	
Dynare (QZ)	7.2e-04	—	5.2e-14	2.3e-11	1
Dynare (Cyclic Reduction)	0.92	7.4e-13	2.9e-15	1e-11	10
Dynare (Log Reduction)	1.3	1e-12	2.3e-14	1.5e-11	9
Baseline Newton	17	1.1e+02	1.3e-14	3.2e-09	99
Baseline Bernoulli	14	7.7e-13	3.7e-14	2.6e-11	4.4e+02
Doubling SF1	1	7.8e-13	8.6e-15	4.9e-12	10
Doubling SF2	0.85	7e-13	8.1e-15	4.9e-12	10

TABLE 1. Results: Model of Smets and Wouters (2007), Posterior Mode

- For Dynare (QZ), refer to Adjemian, Bastani, Juillard, Mihoubi, Perendia, Ratto, and Villemot (2011).
- Run Time for Dynare (QZ) 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 (71).

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.

In table 2 we examine the different methods as solution refinement techniques, by parameterizing the model of Smets and Wouters (2007) to an economically relevant numerical instability following Meyer-Gohde (2023a) and initializing the different methods at the Dynare QZ solution. Note that the reduction methods implemented in Dynare 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 Meyer-Gohde and Saecker's (2022) Newton and Meyer-Gohde's (2023b) Bernoulli method. First, consistent with theorem 3, 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. The Newton algorithm provides substantially more accuracy than the Bernoulli algorithm but at a much higher additional cost, echoing a tradeoff mentioned above. The SF1 algorithm breaks through this barrier, providing roughly the same level

Method	Relative Performance		Forward Errors		Iterations
	Run Time	Variance π_t	Bound 1	Bound 2	
Dynare (QZ)	0.0019	0.28	3.5e-13	4.6	1
Baseline Newton	19	0.45	3.9e-17	0.00058	4
Baseline Bernoulli	13	0.39	1.2e-15	0.018	90
Doubling SF1	5.6	0.33	6.6e-17	0.0009	11
Doubling SF2	3.2	0.48	6.6e-14	0.86	12

TABLE 2. Results: Model of Smets and Wouters (2007), Numerically Problematic Parameterization

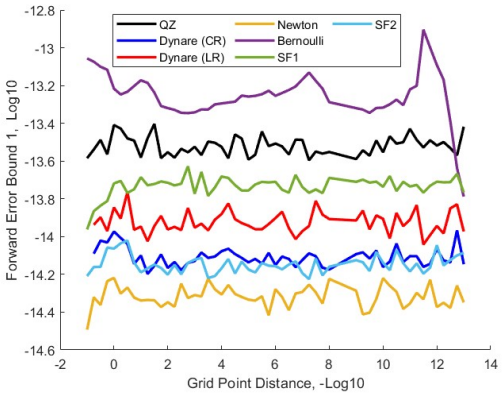
- For Dynare, refer to Adjemian, Bastani, Juillard, Mihoubi, Perendia, Ratto, and Villemot (2011).
- Run Time for Dynare in seconds, for all others, run time relative to Dynare.
- Variance π_t gives the associated value for the population or theoretical variance of inflation - note that two algorithms 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 (71).

of accuracy as the Newton algorithm at half the additional computational costs of the Bernoulli algorithm. That the Newton method is so time consuming despite the relative 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. Notice that the resulting implied variances of inflation, π_t, differ on an economically relevant scale. That is, the more accurate methods all agree that the QZ solution understates the variance of inflation by up to about one-half.[13]

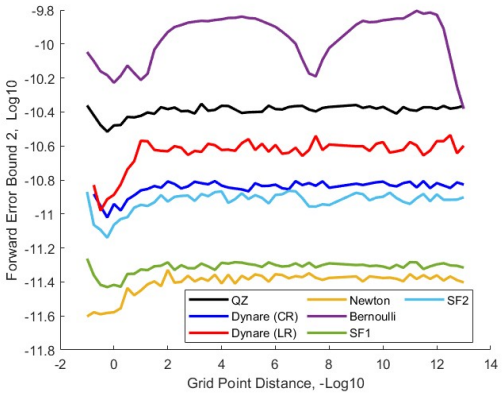
[13]Note that the different refinements still do not entirely agree on the actual level of the variance, this parametrization is very poorly conditioned - see Meyer-Gohde (2023a) - and hence very sensitive to small differences in the solutions. Additionally, the same method, Dynare's native theoretical moment calculator, is used with all methods to calculate the variance. Given the warnings of ill condition, a researcher would be well advised to calculate the moment with each moment more carefully - we do not do so as we want to proceed uniformly with the results from each method and, hence, follow the choice of method by QZ in Dynare.

We now take this refinement perspective and apply the different algorithms to solve iteratively for different parameterizations of the Taylor rule in the final experiment with 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 experiment iterates through a grid of 10×10 for the response in the Taylor rule to inflation and the output gap by varying the size of the interval in the grid - 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 and Wouters (2007) calibrate them to $r_\pi = 2.0443$ and $r_Y = 0.0882$). The algorithm iterates through the two-dimensional grid taking the solution from 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 initialization.

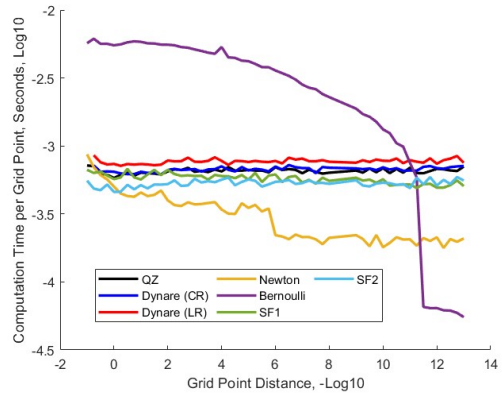
Figure 1 summarizes the experiment graphically. Firstly, the two top panels show that the accuracy of the algorithms is independent of the grid spacing (and, hence, how close the parameter steps are from each other), with the exception being the baseline Bernoulli method that displays a significant drop in forward errors that coincides with the drop in computation time in the lower half - at a close enough parameterization, the Bernoulli algorithm starts with a guess from the previous parameterization that is accurate beyond 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 closely spaced grids) the least accurate. All of the algorithms here, the doubling and reduction algorithms, are more accurate than QZ, with the cyclic reduction and SF2 algorithms roughly the same and close to Newton, then the log reduction follows, with the SF1 doubling algorithm between QZ and the log reduction algorithm. As to be expected from theorem 3, the SF2 doubling algorithm does not systematically profit in terms of reduced computation time as the parameter iterations get closer and closer as the Newton and Bernoulli algorithms do. The SF1 doubling algorithm, in contrast, does profit with a clear downward trend in the computation time as can be seen in figure 1d. This downward trend is hardly recognizable in figure 1d



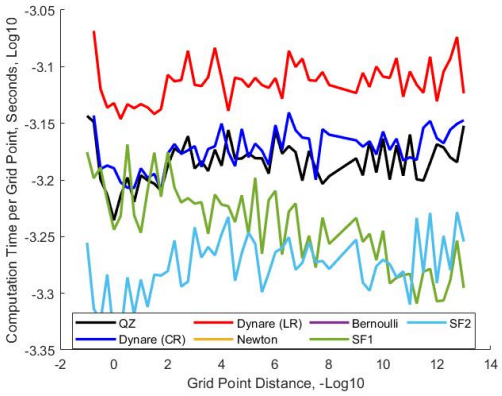
(A) Forward Error 1



(B) Forward Error 2



(C) Computation Time per Grid Point, All methods



(D) Computation Time per Grid Point, Subset

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

Figures 1a and 1b plot the upper forward error bounds 1 and 2 against the grid size, log10 scale on both axes. Figures 1c and 1d plot the computation per grid point against the number of grid points, log10 scale on both axes.

when the Newton and Bernoulli algorithms are also plotted highlighting that this effect is far less significant for the SF1 doubling algorithm than for the Newton and Bernoulli algorithms.

5.2. MMB Suite Comparison

We use models from the Macroeconomic Model Data Base (MMB) to investigate the properties of the structure-preserving doubling algorithms. This gives us the advantage to investigate the properties of the algorithms in a model-robust fashion.

We use version 3.1 which contains 151 different models, ranging from small scale to large scale models. The latter include estimated models of the US, EU, and multi-country economies. We use the algorithm on a subset of models appropriate for reproduction[14] for which their differing sizes are visible in figure 2.

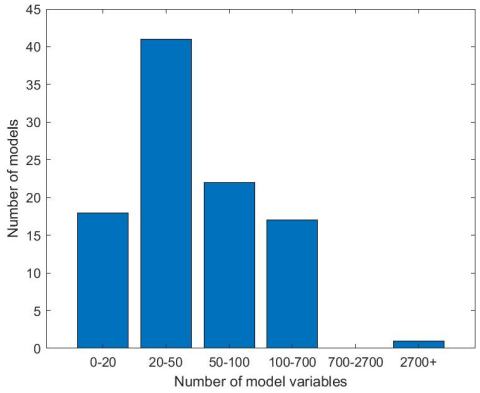


FIGURE 2. Histogram over the number of variables for the 99 MMB models

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

Examining the models of the MMB by comparing the doubling methods to the QZ and various alternatives, we solve each applicable model in the MMB 100 times, initializing the methods with a zero matrix and present the results as the average of the middle three quintiles across runs to minimize the effects of outliers.

Table 3 summarizes the results. As noted in Meyer-Gohde and Saecker (2022) and Meyer-Gohde (2023b), the Newton method is somewhat slower but far 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 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 almost all of the models, and doing so at about the same speed and accuracy as the cyclic reduction

[14]Currently, this is 99 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 here.

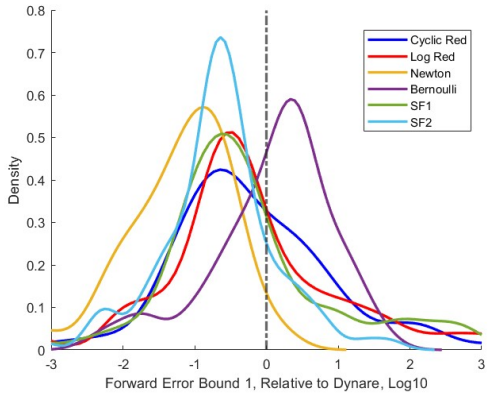
Method	Convergence	Run Time			Forward Error 1			Forward Error 2			Iterations
		Median	Min	Max	Median	Min	Max	Median	Min	Max	
Dynare (QZ)	99	1	1	1	1	1	1	1	1	1	1
Dynare (Cyclic Reduction)	76	1.3	0.42	13	0.37	0.00093	1.7e+03	0.7	0.0031	2e+06	12
Dynare (Log Reduction)	91	1.3	0.16	3.9	0.44	0.00051	4.9e+03	0.57	0.0023	6e+03	10
Baseline Newton	54	2.8	0.34	2.9e+02	0.078	0.00055	1.4	0.099	0.0014	0.94	10
Baseline Bernoulli	99	20	1.3	5.8e+04	1.8	0.0043	41	1.3	0.0023	6.8e+04	7.2e+02
Doubling SF1	93	1.1	0.14	3.6	0.35	0.00028	7.7e+02	0.4	0.00013	2.7e+03	11
Doubling SF2	92	0.84	0.072	47	0.23	0.00065	52	0.38	0.00021	5.5e+02	11

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

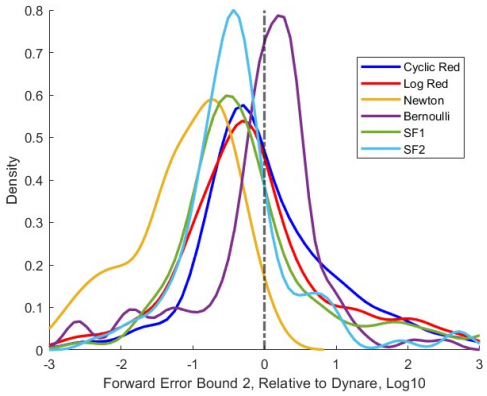
- For Dynare, refer to Adjemian, Bastani, Juillard, Mihoubi, Perendia, Ratto, and Villemot (2011).
- Run time and forward errors relative to Dynare, number of models converging to the stable solution and median of number of iterations in absolute terms.
- Forward error 1 and 2 are the upper bounds for the true forward error, see (71).

method. In terms of the doubling algorithms, they converge more frequently than even the logarithmic reduction methods, but still not for all models. The methods fail to converge when there is a breakdown of nonsingularity of the coefficient matrices in the recursion that need to be inverted. 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. 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 7 of the 99 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 usually has the lower worst case error upper bound, although SF1 has the most accurate best case model. In sum, the doubling algorithms (less so the reduction algorithms) perform very favorably relative to QZ.

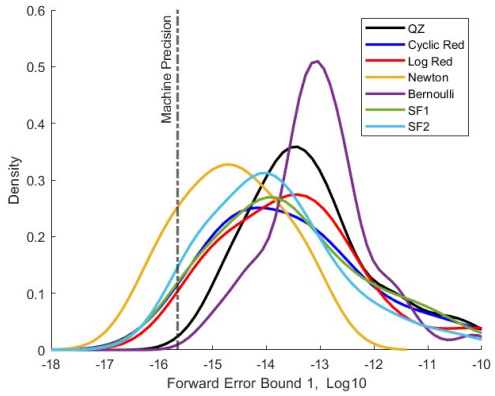
A graphical overview of the entire distribution of forward errors is plotted in figure 3, the upper row relative to QZ and the lower in absolute terms. Meyer-Gohde and Saecker's (2022) Newton algorithm is the most accurate with a clearly left shifted distribution relative to Dynare, with a mode improvement of about one order of magnitude for both forward error measures, and Meyer-Gohde's (2023b) Bernoulli method the least with a clearly right shifted distribution relative to QZ for both error measures. 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 lower panels plot the absolute values and the tighter bound, forward error 1 on the left, 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 upper bounds on forward error inside machine precision and all mass essentially below e-12. The lower right panel shows how much the weaker bound



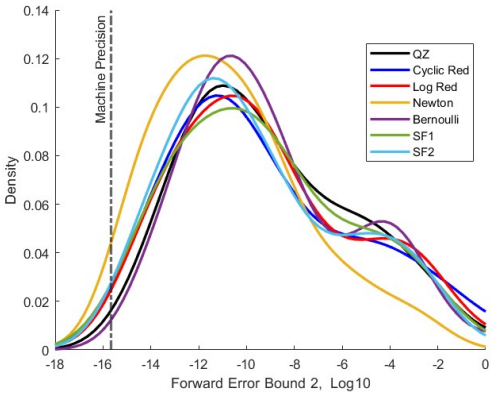
(A) Forward Error 1, Relative to Dynare



(B) Forward Error 2, Relative to Dynare



(C) Forward Error 1



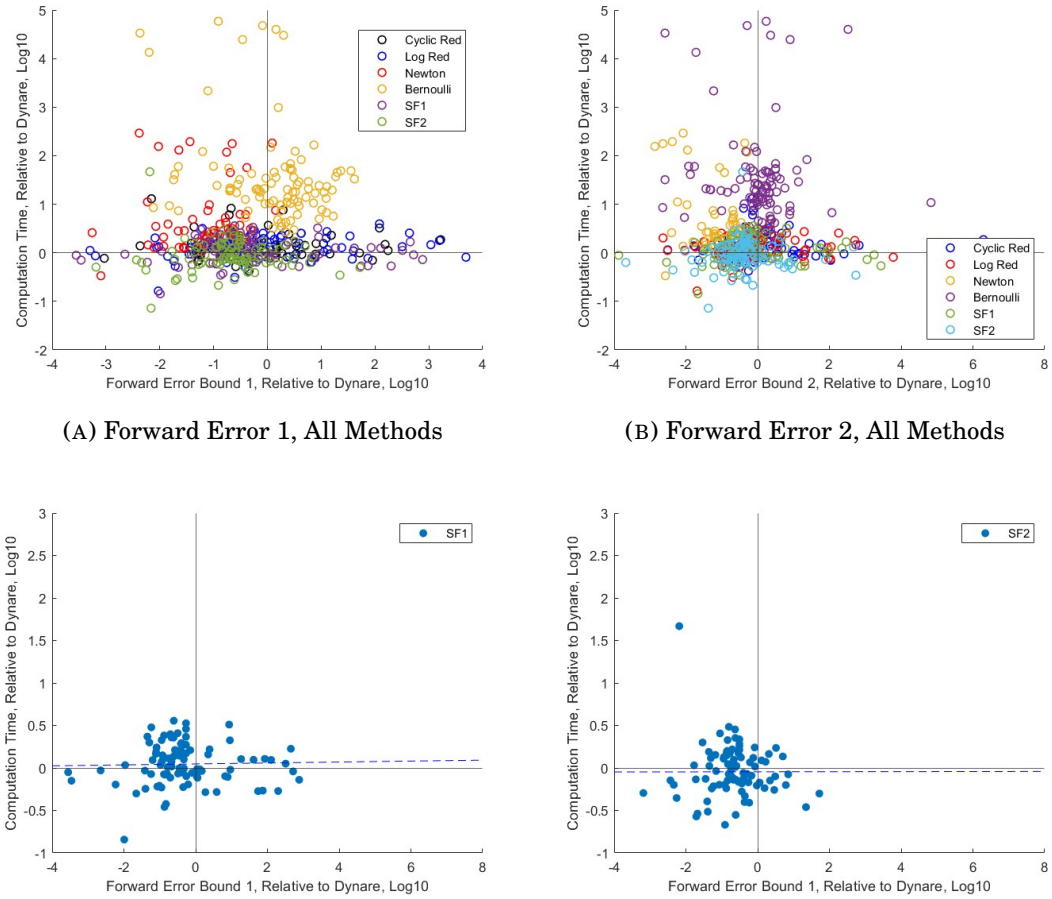
(D) Forward Error 2

FIGURE 3. Distribution of forward error bounds relative to Dynare for the Macroeconomic Model Data Base (MMB)

Figures 3a, 3b 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).

can differ which must be weighed against its lower computational intensity, see Meyer-Gohde (2023a). Nonetheless, it appears that all models in the MMB were solved with acceptable accuracy and the SF2 doubling algorithm is to be preferred among the algorithms here.

Figure 4 provides a model-by-model comparison of the different algorithms' performance relative to QZ. 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 panels all of the methods are plotted using the two different measures of



(A) Forward Error 1, All Methods

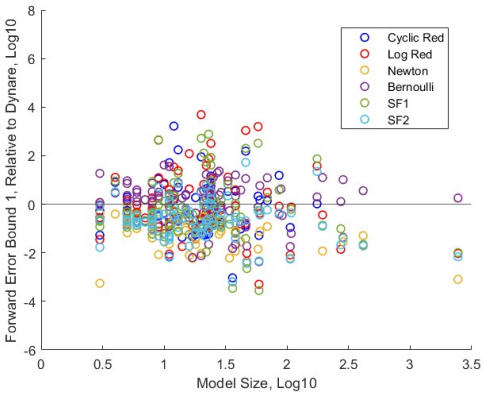
(B) Forward Error 2, All Methods

(C) SDA SF1

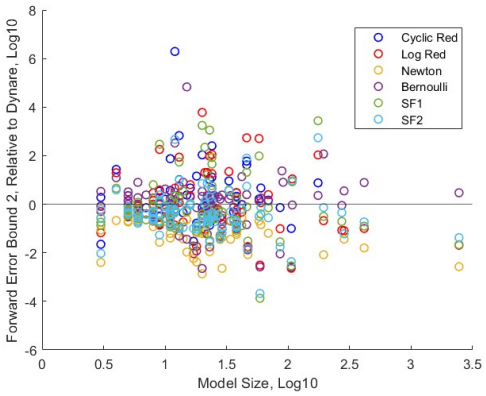
(D) SDA SF2

FIGURE 4. Forward Errors and Computation Time Relative to Dynare, log10 scales, for the Macroeconomic Model Data Base (MMB)

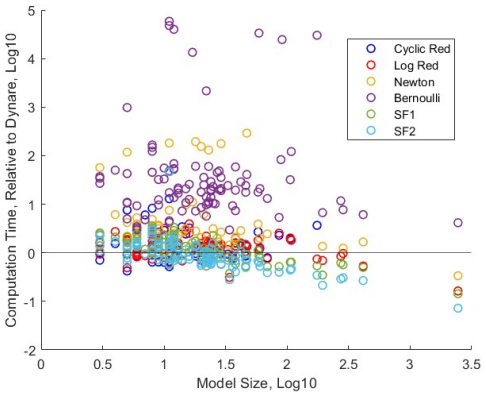
the forward error. First, recognize that the Bernoulli method is sometimes more and sometimes less accurate than QZ but usually slower (higher computation time), whereas the baseline Newton method is generally more accurate but usually slightly slower than QZ. The doubling and reduction algorithms require 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.



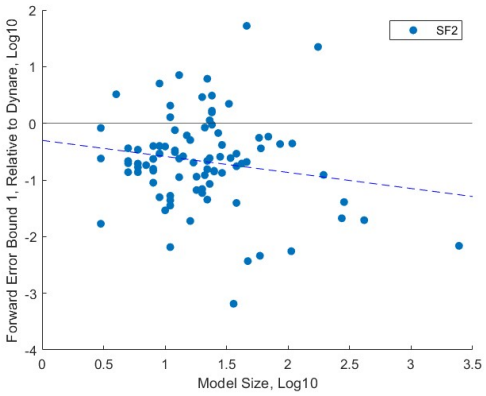
(A) Forward Error 1, Relative to Dynare



(B) Forward Error 2, Relative to Dynare



(C) Computation Time, Relative to Dynare



(D) Forward Error 1, Relative to Dynare

FIGURE 5. Forward Errors, Computation Time and Number of State Variables for the Macroeconomic Model Data Base (MMB)

Figures 5a, 5b plot the upper bounds of the forward error 1 and 2 against model size (number of state or backward looking variables) for all methods, log10 scale on both axes.

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, on the algorithms. The top tow panels give 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 lower panels plot 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 table 4. Note that in contrast to table 3, now the SF1 doubling algorithm, along with the baseline Newton method, converges successfully to the unique stable solution in all models. Consistent with theorem 3, the SF2 doubling algorithm, however, does not and continues to converge or not for the same models as under the initialization with the zero matrix.

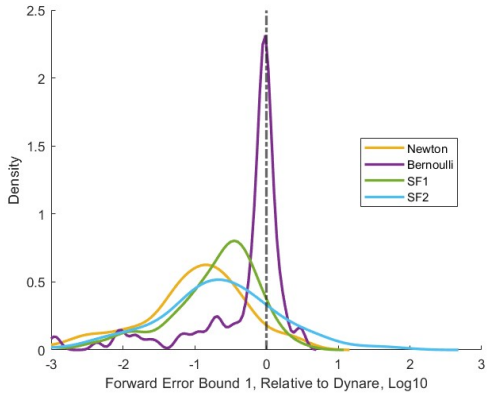
Both the Bernoulli and Newton methods run one iteration, and the later generally achieves a greater increase in accuracy albeit at a higher computation cost due to the computational intensity of the Newton step. The SF1 doubling algorithm runs through multiple iterations, ending up being slightly faster but slightly less 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 1.9 times QZ computation cost and 3.1 times the forward error. The worst case Newton computation costs are an additional 330 times the QZ initial time and the SF2 doubling algorithm has at worst a forward error bound 53 times that of QZ. 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 upper row relative to those from Dynare's QZ method and the lower in absolute terms. 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 one Bernoulli iteration was performed, the same is true for the Newton algorithm. The doubling algorithms perform favorably, with both displaying left shifted distributions of error bounds relative to QZ. Taken together with the results from table 4, the SF1 doubling algorithm can be considered an ideal solution refinement algorithm across a wide variety of models.

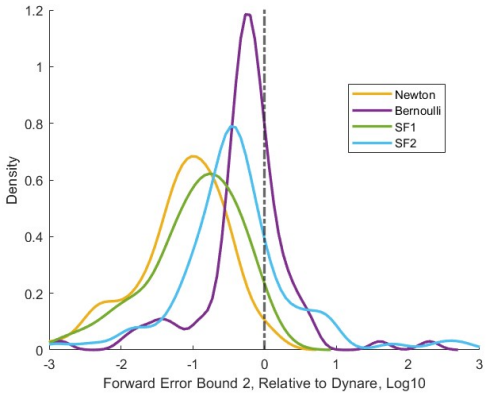
Method	Convergence	Run Time			Forward Error 1			Forward Error 2			Iterations
		Median	Min	Max	Median	Min	Max	Median	Min	Max	
Dynare (QZ)	99	1	1	1	1	1	1	1	1	1	1
Baseline Newton	99	0.63	0.032	3.3e+02	0.12	0.00038	2.9	0.098	1.1e-05	1.1	1
Baseline Bernoulli	99	0.16	0.0062	3.9e+03	0.87	0.001	3	0.6	0.00038	2e+02	1
Doubling SF1	99	0.59	0.092	1.9	0.27	0.00089	3.1	0.14	1.8e-05	1.4	8
Doubling SF2	92	0.72	0.054	18	0.2	0.00066	53	0.37	0.0012	5.6e+02	11

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

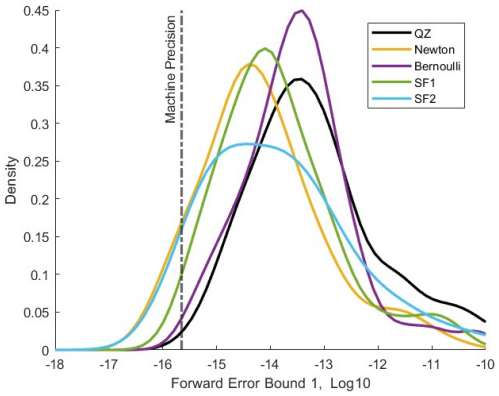
- For Dynare, refer to Adjemian, Bastani, Juillard, Mihoubi, Perendia, Ratto, and Villemot (2011).
- Run Time and foward errors relative to Dynare.
- Forward error 1 and 2 are the upper bounds for the true forward error, see (71).



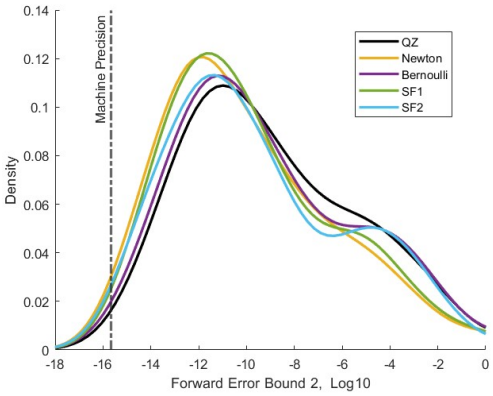
(A) Forward Error 1, Relative to Dynare



(B) Forward Error 2, Relative to Dynare



(C) Forward Error 1



(D) Forward Error 2

FIGURE 6. Distribution of forward error bounds relative to Dynare for the Macroeconomic Model Data Base (MMB)

Figures 6a, 6b 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)).

6. CONCLUSION

We have introduced and developed doubling algorithms for solving linear DSGE models as alternatives to QZ methods (Moler and Stewart, 1973; Golub and van Loan, 2013). We connect these to the related reduction algorithms implemented, albeit silently, in Dynare. The doubling algorithms have theoretical convergence results that promise quadratic convergence rates like the Newton based methods of Meyer-Gohde and Saecker (2022) as well as convergence to the stable solution as with the Bernoulli methods of Meyer-Gohde (2023b).

In a set of experiments using the Smets and Wouters (2007) model and the suite of models in the Macroeconomic Model Data Base (MMB), we find that both the doubling algorithms perform very favorably relative to QZ, with generally more accurate solutions produced using less computational time. The results are not entirely clear cut, as there are outliers in terms of accuracy, computational time, and convergence. We extended the doubling algorithms from the literature to operate off of user defined initializations and provide convincing evidence that the SF1 doubling algorithm can reliably provide low cost, high accuracy refinements of existing solutions. That is, in the absence of any specific considerations, the SF1 doubling algorithm should be the algorithm of choice of researchers looking to improve the accuracy of a solution produced by another method.

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.

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.

ANDERSON, G. S., A. LEVIN, AND E. SWANSON (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., AND G. MOORE (1985): "A Linear Algebraic Procedure for Solving Linear Perfect Foresight Models," *Economics Letters*, 17(3), 247–252.

BINDER, M., AND A. MEYER-GOHDE (2024): "Revisiting the Fully Recursive Computation of Multivariate Linear Rational Expectations Models," mimeo, Goethe University Frankfurt, Institute for Monetary and Financial Stability (IMFS).

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.

CHEN, X. S., AND P. LV (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.

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.

GOLUB, G. H., AND C. F. VAN LOAN (2013): *Matrix Computations*. The Johns Hopkins University Press, fourth edn.

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.

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.

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.

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.

KÅGSTRÖM, B., AND P. POROMAA (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.

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 βEconomic 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. (2022): "Backward Error and Condition Number Analysis of Linear DSGE Solutions," mimeo, Goethe University Frankfurt, Institute for Monetary and Financial Stability (IMFS).

——— (2023a): "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 (IMFS).

——— (2023b): "Solving Linear DSGE Models with Bernoulli Methods," IMFS Working Paper Series 182, Goethe University Frankfurt, Institute for Monetary and Financial Stability (IMFS).

MEYER-GOHDE, A., AND I. PIGKOU (2023): "Pencils, Scaling, and Deflating: Improving the Accuracy of DSGE Perturbations," mimeo, Goethe University Frankfurt, Institute for Monetary and Financial Stability (IMFS).

MEYER-GOHDE, A., AND J. SAECKER (2022): "Solving Linear DSGE Models with Newton Methods," IMFS Working Paper Series 174, Goethe University Frankfurt, Institute for Monetary and Financial Stability (IMFS).

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).

SIMS, C. A. (2001): "Solving Linear Rational Expectations Models," *Computational Economics*, 20(1-2), 1–20.

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.

STEWART, G. W. (1971): "Error Bounds for Approximate Invariant Subspaces of Closed Linear Operators," *SIAM Journal on Numerical Analysis*, 8(4), 796–808.

UHLIG, H. (1999): "A Toolkit for Analysing Nonlinear Dynamic Stochastic Models Easily," in *Computational Methods for the Study of Dynamic Economies*, ed. by R. Marimon, and A. Scott, chap. 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., 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. Elsevier.

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

6.1. Detailed Dynare Topology

Here we summarize the details in the matrix quadratic that follows from the typology of variables from Dynare as laid out in Villemot (2011). See Meyer-Gohde and Saecker (2022) for details.

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

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

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

$$n^d = n^{--} + n^m + n^{++}, \quad n^+ = n^m + n^{++}, \quad n^- = n^{--} + n^m, \quad n = n^s + n^d = n^s + n^{--} + n^m + n^{++}$$

The transition matrix, P, from (4) that solves the matrix equation (5) can be subdivided in accordance to Dynare's typology as

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

The matrix quadratic can be expressed as

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

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

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

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

Following Meyer-Gohde and Saecker (2022), 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 $\rightarrow \mathbf{P}_{\bullet,s} = \underset{n \times n^s}{\mathbf{0}}$, $\mathbf{P}_{\bullet,++} = \underset{n \times n^{++}}{\mathbf{0}}$, whence $\mathbf{P}$ is $\mathbf{P} = \underset{n}{\overset{\begin{matrix} n^s & n^{--} & n^m & n^{++} \end{matrix}}{\begin{bmatrix} \mathbf{0} & \mathbf{P}_{\bullet,--} & \mathbf{P}_{\bullet,m} & \mathbf{0} \end{bmatrix}}}$ and $\mathbf{M}(\mathbf{P}) = \begin{bmatrix} \underset{n \times n^s}{\mathbf{0}} & \underset{n \times n^{--}}{\mathbf{M}(\mathbf{P})^{--}} & \underset{n \times n^m}{\mathbf{M}(\mathbf{P})^m} & \underset{n \times n^{++}}{\mathbf{0}} \end{bmatrix}$. Consequentially, the first n^s rows of the matrix quadratic, taking $\underset{n}{\overset{}{\begin{matrix} n^{--} \\ n^m \\ n^{++} \end{matrix}\begin{bmatrix} \mathbf{P}_{--,\bullet} \\ \mathbf{P}_{m,\bullet} \\ \mathbf{P}_{++,\bullet} \end{bmatrix}}}$ as given, yield $\underset{}{\overset{\begin{matrix} n^{--} & n^m \end{matrix}}{n^s \begin{bmatrix} \mathbf{P}_{s,--} & \mathbf{P}_{s,m} \end{bmatrix}}}$ as

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

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

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

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

$$\underset{n^d \times n^d}{\tilde{\mathbf{P}}}^2 \underset{n^d \times n^+}{\tilde{\mathbf{A}}^+} \underbrace{\overset{\begin{matrix} n^{--} & n^m & n^{++} \end{matrix}}{\begin{bmatrix} n^{--} \\ n^m \\ n^{++} \end{bmatrix}\begin{bmatrix} \mathbf{P}_{--,--} & \mathbf{P}_{--,m} & \mathbf{P}_{--,++} \\ \mathbf{P}_{m,--} & \mathbf{P}_{m,m} & \mathbf{P}_{m,++} \\ \mathbf{P}_{++,--} & \mathbf{P}_{++,m} & \mathbf{P}_{++,++} \end{bmatrix}}}_{\underset{n^d \times n^d}{\tilde{\mathbf{P}}}} \cdot \underset{n^d \times n^d}{\tilde{\mathbf{P}}} + \underset{n^d \times n^d}{\tilde{\mathbf{A}}^0} \underset{n^d \times n^d}{\tilde{\mathbf{P}}}$$

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

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

$$
\begin{array}{l}
\tilde{\mathbf{P}} = \begin{array}{c} \\ n^{--} \\ n^{m} \\ n^{++} \end{array}
\begin{array}{c} \begin{array}{ccc} n^{--} & n^{m} & n^{++} \end{array} \\
\left[\begin{array}{c|c|c} \mathbf{P}_{--,--} & \mathbf{P}_{--,m} & \mathbf{P}_{--,++} \\ \hline \mathbf{P}_{m,--} & \mathbf{P}_{m,m} & \mathbf{P}_{m,++} \\ \hline \mathbf{P}_{++,--} & \mathbf{P}_{++,m} & \mathbf{P}_{++,++} \end{array}\right] \end{array}
= \begin{array}{c} \\ n^{--} \\ n^{m} \\ n^{++} \end{array}
\begin{array}{c} \begin{array}{ccc} n^{--} & n^{m} & n^{++} \end{array} \\
\left[\begin{array}{c|c|c} \mathbf{P}_{--,--} & \mathbf{P}_{--,m} & \mathbf{0} \\ \hline \mathbf{P}_{m,--} & \mathbf{P}_{m,m} & \mathbf{0} \\ \hline \mathbf{P}_{++,--} & \mathbf{P}_{++,m} & \mathbf{0} \end{array}\right] \end{array}
\equiv \begin{array}{c} \\ n^{--} \\ n^{m} \\ n^{++} \end{array}
\begin{array}{c} \begin{array}{ccc} n^{--} & n^{m} & n^{++} \end{array} \\
\left[\begin{array}{c|c|c} \underset{n^{--} \times n^-}{\overline{\mathbf{P}}} & & \mathbf{0} \\ \hline & \underset{n^+ \times n^-}{\widehat{\mathbf{P}}} & \mathbf{0} \\ \hline & & \mathbf{0} \end{array}\right] \end{array}
\end{array}
$$

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

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

6.2. Detailed Dynare Topology - SDA for SF1

Beginning with Algorithm 1, the initial values

$$X_0 = -B^{-1}C, \qquad Y_0 = -B^{-1}A, \qquad E_0 = -B^{-1}C, \qquad F_0 = -B^{-1}A$$

can be written as

$$X_0 = -B^{-1}C = \left(\underset{n^d \times n^d}{\tilde{\mathbf{A}}^{0}}\right)^{-1} n^d \begin{array}{c} \begin{array}{cc} n^{-} & n^{++} \end{array} \\ \left[\begin{array}{c|c} \tilde{\mathbf{A}}^{-}_{n^d \times n^-} & \mathbf{0} \end{array}\right] \end{array} = \left[\underset{n^d \times n^-}{(\tilde{\mathbf{A}}^{0})^{-1}\tilde{\mathbf{A}}^{-}} \;\middle|\; \underset{n^d \times n^{++}}{\mathbf{0}}\right] = \left[\underset{n^d \times n^-}{\overline{X_0}} \;\middle|\; \underset{n^d \times n^{++}}{\mathbf{0}}\right] = \left[\underset{n^d \times n^-}{\overline{E_0}} \;\middle|\; \underset{n^d \times n^{++}}{\mathbf{0}}\right] = E_0$$

and

$$Y_0 = -B^{-1}A = \left(\underset{n^d \times n^d}{\tilde{\mathbf{A}}^{0}}\right)^{-1} n^d \begin{array}{c} \begin{array}{cc} n^{--} & n^{+} \end{array} \\ \left[\begin{array}{c|c} \mathbf{0} & \tilde{\mathbf{A}}^{+}_{n^d \times n^+} \end{array}\right] \end{array} = \left[\underset{n^d \times n^{--}}{\mathbf{0}} \;\middle|\; \underset{n^d \times n^+}{(\tilde{\mathbf{A}}^{0})^{-1}\tilde{\mathbf{A}}^{+}}\right] = \left[\underset{n^d \times n^{--}}{\mathbf{0}} \;\middle|\; \underset{n^d \times n^+}{\overline{Y_0}}\right] = \left[\underset{n^d \times n^{--}}{\mathbf{0}} \;\middle|\; \underset{n^d \times n^+}{\overline{F_0}}\right] = F_0$$

Now proceeding by induction and assuming that X_k, Y_k, E_k, and F_k have the same dimensions (i.e., zero and non zero), we will show this holds for X_{k+1}, Y_{k+1}, E_{k+1}, and F_{k+1}. Beginning with E_{k+1},

$$E_{k+1} = E_k (I - Y_k X_k)^{-1} E_k = E_k (I - Y_k X_k)^{-1} \begin{bmatrix} \underset{n^d \times n^-}{\overline{E_k}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} = \begin{bmatrix} \underset{n^d \times n^-}{E_k (I - Y_k X_k)^{-1} \overline{E_k}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} = \begin{bmatrix} \underset{n^d \times n^-}{\overline{E_{k+1}}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix}$$

which has the same zero and non zero structure as E_k. By direct extension this holds equivalently for X_k

$$\begin{aligned} X_{k+1} = X_k + F_k (I - X_k Y_k)^{-1} X_k E_k &= \begin{bmatrix} \underset{n^d \times n^-}{\overline{X_k}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} + F_k (I - X_k Y_k)^{-1} X_k \begin{bmatrix} \underset{n^d \times n^-}{\overline{E_k}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} \\ &= \begin{bmatrix} \underset{n^d \times n^-}{\overline{X_k}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} + \begin{bmatrix} \underset{n^d \times n^-}{F_k (I - X_k Y_k)^{-1} X_k \overline{E_k}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} \\ &= \begin{bmatrix} \underset{n^d \times n^-}{\overline{X_k} + F_k (I - X_k Y_k)^{-1} X_k \overline{E_k}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} = \begin{bmatrix} \underset{n^d \times n^-}{\overline{X_{k+1}}} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} \end{aligned}$$

Now for F_{k+1}

$$F_{k+1} = F_k (I - X_k Y_k)^{-1} F_k = F_k (I - X_k Y_k)^{-1} \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{\overline{F_k}} \end{bmatrix} = \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{F_k (I - X_k Y_k)^{-1} \overline{F_k}} \end{bmatrix} = \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{\overline{F_{k+1}}} \end{bmatrix}$$

By direct extension this holds equivalently for Y_k

$$\begin{aligned} Y_{k+1} = Y_k + E_k (I - Y_k X_k)^{-1} Y_k F_k &= \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{\overline{Y_k}} \end{bmatrix} + E_k (I - Y_k X_k)^{-1} Y_k \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{\overline{F_k}} \end{bmatrix} \\ &= \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{\overline{Y_k}} \end{bmatrix} + \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{E_k (I - Y_k X_k)^{-1} Y_k \overline{F_k}} \end{bmatrix} \\ &= \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{\overline{Y_k} + E_k (I - Y_k X_k)^{-1} Y_k \overline{F_k}} \end{bmatrix} = \begin{bmatrix} \underset{n^d \times n^{--}}{\mathbf{0}} & \underset{n^d \times n^+}{\overline{Y_{k+1}}} \end{bmatrix} \end{aligned}$$

This gives recursions in the generically non zero matrices $\overline{X_k}$, $\overline{Y_k}$, $\overline{E_k}$, and $\overline{F_k}$.

Noting that X_k and Y_k can be written out blockwise as

$$X_k = \begin{array}{c} \\ n^{--} \\ n^m \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} X_{k;--,--} & X_{k;--,m} & \mathbf{0} \\ X_{k;m,--} & X_{k;m,m} & \mathbf{0} \\ X_{k;++,--} & X_{k;++,m} & \mathbf{0} \end{bmatrix} \end{array} = \begin{array}{c} \\ n^{--} \\ n^m \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} & \underset{n^d \times n^-}{\overline{X_k}} & \\ & & \end{bmatrix} \end{array} \begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{array} = \begin{array}{c} \\ n^{--} \\ n^m \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} \underset{n^{--} \times n^-}{\overline{\overline{X_k}}} \\ \underset{n^+ \times n^-}{\overline{X_k}} \end{bmatrix} \end{array} \begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{array}$$

and

$$Y_k = \begin{array}{c} \\ n^{--} \\ n^m \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} \mathbf{0} & Y_{k;--,m} & Y_{k;--,++} \\ \mathbf{0} & Y_{k;m,m} & Y_{k;m,++} \\ \mathbf{0} & Y_{k;++,m} & Y_{k;++,++} \end{bmatrix} \end{array} = \begin{array}{c} \\ n^{--} \\ n^m \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \end{array} \begin{array}{c} \\ \\ \underset{n^d \times n^+}{\overline{Y_k}} \end{array} = \begin{array}{c} \\ n^{--} \\ n^m \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^m & n^{++} \end{array} \\ \begin{bmatrix} \mathbf{0} & \underset{n^- \times n^+}{\overline{\overline{Y_k}}} \\ \mathbf{0} & \\ \mathbf{0} & \underset{n^{++} \times n^+}{\overline{Y_k}} \end{bmatrix} \end{array}$$

and hence their products can be calculated as

$$X_k Y_k = \begin{array}{c} \\ n^{--} \\ n^{m} \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^{m} & n^{++} \end{array} \\ \begin{bmatrix} \mathbf{0} & & \\ \mathbf{0} & & \overline{\overline{X_k}}\,\overline{\overline{Y_k}} \\ \mathbf{0} & & \end{bmatrix} \end{array} \quad \text{and} \quad Y_k X_k = \begin{array}{c} \\ n^{--} \\ n^{m} \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^{m} & n^{++} \end{array} \\ \begin{bmatrix} & & \mathbf{0} \\ \overline{Y_k}\,\overline{X_k} & & \mathbf{0} \\ & & \mathbf{0} \end{bmatrix} \end{array}$$

Similar calculations apply to $Y_k \overline{F_k}$ and $X_k \overline{E_k}$.

For Algorithm 5, note that if $\widehat{X}_0$, $\widehat{Y}_0$, $\widehat{E}_0$, $\widehat{F}_0$ have (zero and non zero) dimensions that correspond to those of X_0, Y_0, E_0, and F_0, the same approach can be taken. Comparing, the only requirement is that P_0 takes the form

$$P_0 = \begin{array}{c} \\ n^{--} \\ n^{m} \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^{m} & n^{++} \end{array} \\ \begin{bmatrix} & & \mathbf{0} \\ \overline{P_0} & & \mathbf{0} \\ & & \mathbf{0} \end{bmatrix} \end{array}$$

which conforms to the structure of the matrix quadratic as shown in the subsection above.

6.3. Detailed Dynare Topology - SDA for SF2

Beginning with Algorithm 2, the initial values

$$X_0^\dagger = 0, \qquad Y_0^\dagger = -B, \qquad E_0^\dagger = -C, \qquad F_0^\dagger = -A$$

can be written as

$$X_0^\dagger = 0, \qquad Y_0^\dagger = -\underset{n^d \times n^d}{\tilde{\mathbf{A}}^0}, \qquad E_0^\dagger = n^d \begin{bmatrix} \underset{n^d \times n^-}{-\tilde{\mathbf{A}}^-} & \mathbf{0} \end{bmatrix}, \qquad F_0^\dagger = n^d \begin{bmatrix} \mathbf{0} & \underset{n^d \times n^+}{-\tilde{\mathbf{A}}^+} \end{bmatrix}$$

For the calculations, we will work with:

$$E_{k+1}^\dagger = E_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger$$
$$F_{k+1}^\dagger = F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} F_k^\dagger$$
$$X_{k+1}^\dagger = X_k^\dagger - F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger$$
$$X_{k+1}^\dagger - Y_{k+1}^\dagger = X_k^\dagger - Y_k^\dagger - F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger - E_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} F_k^\dagger$$

As above, $E_{k+1}^\dagger$ and $F_{k+1}^\dagger$ will maintain the (zero and non zero) dimensions that correspond to $E_0^\dagger$ and $F_0^\dagger$ via the post multiplication of $E_k^\dagger$ and $F_k^\dagger$ and induction. The same post multiplication gives $X_{k+1}^\dagger$ the (zero and non zero) dimensions that correspond to $E_0^\dagger$

$$\begin{aligned} X_{k+1}^\dagger = X_k^\dagger - F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} E_k^\dagger &= \begin{bmatrix} \underset{n^d \times n^-}{\overline{X_k}^\dagger} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} - F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} \begin{bmatrix} \underset{n^d \times n^-}{\overline{E_k}^\dagger} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} \\ &= \begin{bmatrix} \underset{n^d \times n^-}{\overline{X_k}^\dagger} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} - \begin{bmatrix} \underset{n^d \times n^-}{F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} \overline{E_k}^\dagger} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} \\ &= \begin{bmatrix} \underset{n^d \times n^-}{\overline{X_k}^\dagger - F_k^\dagger \left(X_k^\dagger - Y_k^\dagger\right)^{-1} \overline{E_k}^\dagger} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} = \begin{bmatrix} \underset{n^d \times n^-}{\overline{X_{k+1}}^\dagger} & \underset{n^d \times n^{++}}{\mathbf{0}} \end{bmatrix} \end{aligned}$$

For Algorithm 6 with an initial guess P_0, $E_0^\dagger$ and $F_0^\dagger$ are unchanged, so the only requirement is that P_0 takes the form

$$P_0 = \begin{array}{c} \\ n^{--} \\ n^{m} \\ n^{++} \end{array} \begin{array}{c} \begin{array}{ccc} n^{--} & n^{m} & n^{++} \end{array} \\ \begin{bmatrix} & & \mathbf{0} \\ & \overline{P_0} & \mathbf{0} \\ & & \mathbf{0} \end{bmatrix} \end{array}$$

to have (zero and non zero) dimensions that correspond to $E_0^\dagger$ - this conforms to the structure of the matrix quadratic as shown in the subsection above.

Proof of Theorem 2

For the proofs of (1) and (2) see Theorems 3.18 and 3.19 by Huang, Li, and Lin (2018, pp. 35,37). To proof (3) first note that Assumption 1 implies either $\rho(P) \leq 1 \wedge \rho(P_d) < 1$ and / or $\rho(P) < 1 \wedge \rho(P_d) \leq 1$.

In the following we will first deal with the scenario where $\rho(P) \leq 1 \wedge \rho(P_d) < 1$ and show that in this case H_k converges quadratically to zero. To see this, note that using (64) of the primal UQME (5) and the dual UQME (20), respectively, we receive

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

$$H_k = \mathscr{N}_k - L_k \mathscr{N}_k^2. \tag{A2}$$

where $\mathscr{N}_k = \mathscr{N}^{2^k} = P_d^{2^k}$. Using (A1) to substitute L_k in (A2) yields

$$H_k = \mathscr{N}_k - \mathscr{M}_k \mathscr{N}_k^2 + H_k \mathscr{M}_k^2 \mathscr{N}_k^2. \tag{A3}$$

Thus, for any sub-multiplicative matrix norm $\|.\|$ we receive

$$\|H_k\| \leq \|\mathscr{N}_k\| + \|\mathscr{M}_k\| \|\mathscr{N}_k\|^2 + \|H_k\| \|\mathscr{M}_k\|^2 \|\mathscr{N}_k\|^2. \tag{A4}$$

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

$$\|H_k\| \leq \frac{1+\epsilon_k}{1-\epsilon_k^2} \|\mathscr{N}_k\| \quad \Rightarrow \quad \lim_{k \to \infty} \|H_k\| = 0. \tag{A5}$$

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

$$\|H_k\|^{1/2^k} \leq \left(\frac{1+\epsilon_k}{1-\epsilon_k^2} \right)^{1/2^k} \|\mathscr{N}_k\|^{1/2^k} \quad \Rightarrow \quad \lim_{k \to \infty} \|H_k\|^{1/2^k} = \rho(\mathscr{N}) \leq \rho(P_d). \tag{A6}$$

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.,

$$\lim_{k\to\infty} \|\widehat{H}_k\|^{1/2^k} \leq \rho(\mathscr{N}) = \rho(P_d). \tag{A7}$$

Now we may rewrite (65) to

$$P - \widehat{L}_k = \widehat{H}_k \mathscr{M}_{k+1}, \tag{A8}$$

and receive

$$\|P - \widehat{L}_k\| \leq \|\widehat{H}_k\|\|\mathscr{M}_{k+1}\|. \tag{A9}$$

The statement then follows from the Gelfand's formula / the spectral radius theorem as

$$\|P - \widehat{L}_k\|^{1/2^k} \leq \|\widehat{H}_k\|^{1/2^k} \|\mathscr{M}_{k+1}\|^{1/2^k} \quad \Rightarrow \quad \lim_{k\to\infty} \|P - \widehat{L}_k\|^{1/2^k} \leq \rho(P)\rho(P_d). \tag{A10}$$

Now let us consider the scenario where $\rho(P) < 1 \wedge \rho(P_d) \leq 1$. In this case, we may define $\alpha := \rho(P)$ and rewrite (5) and (20) to

$$A_\alpha P_\alpha^2 + BP_\alpha + C_\alpha = 0 \tag{A11}$$

$$C_\alpha P_{\alpha,d}^2 + BP_{\alpha,d} + A_\alpha = 0, \tag{A12}$$

where

$$A_\alpha = \alpha A, \qquad C_\alpha = \alpha^{-1} C, \qquad P_\alpha = \alpha^{-1} P, \qquad P_{\alpha,d} = \alpha P_d. \tag{A13}$$

By definition the solvents P_α and $P_{\alpha,d}$ to (A11) and (A12), respectively, satisfy $\rho(P_\alpha) \leq 1 \wedge \rho(P_{\alpha,d}) < 1$. Hence, if $L_{k,\alpha}$, $H_{k,\alpha}$, $\widehat{L}_{k,\alpha}$, and $\widehat{H}_{k,\alpha}$ denote the quantities of the Logarithmic Reduction with respect to (A11), we get

$$\lim_{k\to\infty} \|P_\alpha - \widehat{L}_{k,\alpha}\|^{1/2^k} \leq \rho(P_\alpha)\rho(P_{\alpha,d}) = \rho(P)\rho(P_d), \tag{A14}$$

from the first part of the proof. From (60), (61), and (62) we receive (via induction) that

$$H_{k,\alpha} = \alpha^{2^k} H_k, \qquad L_{k,\alpha} = \alpha^{-2^k} L_k, \qquad \widehat{H}_{k,\alpha} = \alpha^{\sum_{j=0}^k 2^j} \widehat{H}_k. \tag{A15}$$

Moreover, since $\mathscr{M}_{k,\alpha} = P_\alpha^{2^k} = \alpha^{-2^k} P^{2^k} = \alpha^{-2^k} \mathscr{M}_k$ we may use (65) to show that

$$\|P_\alpha - \widehat{L}_{k,\alpha}\| = \|\widehat{H}_{k,\alpha}\mathscr{M}_{k+1,\alpha}\| = \underbrace{\alpha^{-2^{k+1}+\sum_{j=0}^k 2^j}}_{>1} \|\widehat{H}_k \mathscr{M}_{k+1}\| > \|P - \widehat{L}_k\|. \tag{A16}$$

The statement then follows from (A14) as

$$\|P - \widehat{L}_k\|^{1/2^k} < \|P_\alpha - \widehat{L}_{k,\alpha}\|^{1/2^k} \quad \Rightarrow \quad \lim_{k\to\infty} \|P - \widehat{L}_k\|^{1/2^k} \leq \rho(P)\rho(P_d). \tag{A17}$$

INITIAL GUESS P_0 FOR $AP^2 + BP + C = 0$

Suppose we want to minimize the squared Frobenius norm $\|R(P_0)\|_F^2$ of the residuals

$$R(P_0) = AP_0^2 + BP_0 + C,$$

where we restrict P_0 to

$$P_0 = \text{diag}(p_1, \ldots, p_n), \quad \text{with} \quad p1, \ldots, p_n \in [-\rho, \rho],\ \rho \in [0, 1).$$

Consequently, we have

$$\begin{aligned}\|R(P_0)\|_F^2 &= \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij}p_j^2 + b_{ij}p_j + c_{ij}\right)^2 \\ &= \sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}^2\, p_j^4 + 2a_{ij}b_{ij}\, p_j^3 + \left(b_{ij}^2 + 2a_{ij}c_{ij}\right)\, p_j^2 + 2b_{ij}c_{ij}\, p_j + c_{ij}^2.\end{aligned}$$

Further, denoting the j-th column of R, A, B, and C as r_j, a_j, b_j, and c_j, respectively, we may write

$$r_j(p_j) = \tilde{a}\, p_j^4 + \tilde{b}\, p_j^3 + \tilde{c}\, p_j^2 + \tilde{d}\, p_j + \tilde{e} \tag{A18}$$

with

$$\begin{aligned}\tilde{a} &= a_j^T\, a_j \\ \tilde{b} &= 2\, a_j^T\, b_j, \\ \tilde{c} &= b_j^T\, b_j + 2\, a_j^T\, c_j, \\ \tilde{d} &= 2\, b_j^T\, c_j, \\ \tilde{e} &= c_j^T\, c_j.\end{aligned}$$

Differentiating r_j with respect to p_j yields

$$r_j'(p_j) = 4\,\tilde{a}\, p_j^3 + 3\,\tilde{b}\, p_j^2 + 2\,\tilde{c}\, p_j + \tilde{d}$$

Hence, an interior solution for p_j must satisfy $r_j'(p_j) = 0$, so that we can obtain the $p_j \in [-\rho, \rho]$ that minimizes $r_j(p_j)$ as

$$p_j^* = \underset{p_j \in g_j \cap \{-\rho, \rho\}}{\text{argmin}}\ r_j(p_j)$$

where g_j is the set of all real roots of $r_j'(p_j)$ in $[-\rho, \rho]$, i.e.,

$$g_j = \left\{p_j \in [-\rho, \rho] \subset \mathbb{R} \ :\ r_j'(p_j) = 0\right\}, \quad |g_j| \le 3.$$

Algorithm 7 summarizes the proceeding to obtain the diagonal elements of P_0.

Algorithm 7: Initial Guess – j-th diagonal element of P_0

Given: a_j, b_j, c_j, and ρ

Set $\tilde{a} = a_j^T \; a_j$

Set $\tilde{b} = 2 \; a_j^T \; b_j$

Set $\tilde{c} = b_j^T \; b_j + 2 \; a_j^T \; c_j$

Set $\tilde{d} = 2 \; b_j^T \; c_j$

Set $\tilde{e} = c_j^T \; c_j$

Define $r(p_j) := \tilde{a} \; p_j^4 + \tilde{b} \; p_j^3 + \tilde{c} \; p_j^2 + \tilde{d} \; p_j + \tilde{e}$

Define $r'(p_j) := 4 \; \tilde{a} \; p_j^3 + 3 \; \tilde{b} \; p_j^2 + 2 \; \tilde{c} \; p_j + \tilde{d}$

Obtain all $p_j \in g = \left\{ x \in [-\rho, \rho] \subset \mathbb{R} \; : \; r'(x) = 0 \right\}$

Return: $\underset{p_j \in g_j \cap \{-\rho, \rho\}}{\operatorname{argmin}} \quad r(p_j)$