Finance and Economics Discussion Series
Divisions of Research & Statistics and Monetary Affairs
Federal Reserve Board, Washington, D.C.

Solving Linear Rational Expectations Models: A Horse Race

Gary S. Anderson

2006-26

Solving Linear Rational Expectations Models: A Horse Race

Gary S. Anderson*
Board of Governors of the Federal Reserve System

May 24, 2006

Abstract

This paper compares the functionality, accuracy, computational efficiency, and practicalities of alternative approaches to solving linear rational expectations models, including the procedures of (Sims, 1996), (Anderson and Moore, 1983), (Binder and Pesaran, 1994), (King and Watson, 1998), (Klein, 1999), and (Uhlig, 1999). While all six procedures yield similar results for models with a unique stationary solution, the AIM algorithm of (Anderson and Moore, 1983) provides the highest accuracy; furthermore, this procedure exhibits significant gains in computational efficiency for larger-scale models.

*I would like to thank Robert Tetlow, Andrew Levin and Brian Madigan for useful discussions and suggestions. I would like to thank Ed Yao for valuable help in obtaining and installing the MATLAB code. The views expressed in this document are my own and do not necessarily reflect the position of the Federal Reserve Board or the Federal Reserve System.

1 Introduction and Summary

Since (Blanchard and Kahn, 1980) a number of alternative approaches for solving linear rational expectations models have emerged. This paper describes, compares and contrasts the techniques of (Anderson, 1997; Anderson and Moore, 1983, 1985), (Binder and Pesaran, 1994), (King and Watson, 1998), (Klein, 1999), (Sims, 1996), and (Uhlig, 1999). All these authors provide MATLAB code implementing their algorithm.[1]

The paper compares the computational efficiency, functionality and accuracy of these MATLAB implementations. The paper uses numerical examples to characterize practical differences in employing the alternative procedures.

Economists use the output of these procedures for simulating models, estimating models, computing impulse response functions, calculating asymptotic covariance, solving infinite horizon linear quadratic control problems and constructing terminal constraints for nonlinear models. These applications benefit from the use of reliable, efficient and easy to use code.

A comparison of the algorithms reveals that:

- For models satisfying the Blanchard-Kahn conditions, the algorithms provide equivalent solutions.[2]
- The Anderson-Moore algorithm requires fewer floating point operations to achieve the same result. This computational advantage increases with the size of the model.
- While the Anderson-Moore, Sims and Binder-Pesaran approaches provide matrix output for accommodating arbitrary exogenous processes, the King-Watson and Uhlig implementations only provide solutions for VAR exogenous process.[3] Fortunately, there are straightforward formulae for augmenting the King-Watson Uhlig and Klein approaches with the matrices characterizing the impact of arbitrary shocks.
- The Anderson-Moore suite of programs provides a simple modeling language for developing models. In addition, the Anderson-Moore implementation requires no special treatment for models with multiple lags and leads. To use each of the other algorithms, one must cast the model in a form with at most one lead or lag. This can be a tedious and error prone task for models with more than a couple of equations.
- Using the Anderson-Moore algorithm to solve the quadratic matrix polynomial equation improves the performance of both Binder-Pesaran's and Uhlig's algorithms.

Section 2 states the problem and introduces notation. This paper divides the algorithms into three categories: eigensystem, QZ, and matrix polynomial methods. Section 3 describes the eigensystem methods. Section 4 describes applications of the QZ algorithm. Section 5 describes applications of the matrix polynomial approach. Section 6 compares the computational efficiency, functionality and accuracy of the algorithms. Section 7 concludes the paper. The appendices provide usage notes for each of the algorithms as well as information about how to compare inputs and outputs from each of the algorithms.

[1] Although (Broze, Gouriéroux, and Szafarz, 1995) and (Zadrozny, 1998) describe algorithms, I was unable to locate code implementing the algorithms.

[2] (Blanchard and Kahn, 1980) developed conditions for existence and uniqueness of linear rational expectations models. In their setup, the solution of the rational expectations model is unique if the number of unstable eigenvectors of the system is exactly equal to the number of forward-looking (control) variables.

[3] I modified Klein's MATLAB version to include this functionality by translating the approach he used in his Gauss version.

2 Problem Statement and Notation

These algorithms compute solutions for models of the form

$$\sum_{i=-\tau}^{\theta} H_i x_{t+i} = \Psi z_t, t = 0, \ldots, \infty \tag{1}$$

with initial conditions, if any, given by constraints of the form

$$x_i = x_i^{data}, i = -\tau, \ldots, -1 \tag{2}$$

where both τ and θ are non-negative, and x_t is an L dimensional vector of endogenous variables with

$$\lim_{t \to \infty} \|x_t\| < \infty \tag{3}$$

and z_t is a k dimensional vector of exogenous variables.

(4)

Solutions can be cast in the form

$$(x_t - x^*) = \sum_{i=-\tau}^{-1} B_i (x_{t+i} - x^*)$$

Given any algorithm that computes the B_i, one can easily compute other quantities useful for characterizing the impact of exogenous variables. For models with $\tau = \theta = 1$ the formulae are especially simple.

Let

$$\Phi = (H_0 + H_1 B)^{-1}$$
$$F = -\Phi H_1 B$$

We can write

$$(x_t - x^*) = B(x_{t-1} - x^*) + \sum_{s=0}^{\infty} F^s \phi \psi z_{t+s}$$

and when

$$z_{t+1} = \Upsilon z_t$$
$$vec(\vartheta) = (I - \Upsilon^T \otimes F)^{-1} vec(\Phi\Psi)$$
$$(x_t - x^*) = B(x_{t-1} - x^*) + \vartheta z_t$$

Consult (Anderson, 1997) for other useful formulae concerning rational expectations model solutions.

I downloaded the MATLAB code for each implementation in July, 1999. See the bibliography for the relevant URL's.

3 Eigensystem Methods

3.1 The Anderson-Moore Algorithm

(Anderson, 1997; Anderson and Moore, 1985) developed their algorithm in the mid 80's for solving rational expectations models that arise in large scale macro models. Appendix B.1 on page 13 provides a synopsis of the model concepts and algorithm inputs and outputs. Appendix A presents pseudocode for the algorithm.

The algorithm determines whether equation 1 has a unique solution, an infinity of solutions or no solutions at all. The algorithm produces a matrix Q codifying the linear constraints guaranteeing asymptotic convergence. The matrix Q provides a strategic point of departure for making many rational expectations computations.

The uniqueness of solutions to system 1 requires that the transition matrix characterizing the linear system have appropriate numbers of explosive and stable eigenvalues (Blanchard and Kahn, 1980), and that the asymptotic linear constraints are linearly independent of explicit and implicit initial conditions (Anderson and Moore, 1985).

The solution methodology entails

1. Manipulating equation 1 to compute a state space transition matrix.
2. Computing the eigenvalues and the invariant space associated with explosive eigenvalues
3. Combining the constraints provided by:
   (a) the initial conditions,
   (b) auxiliary initial conditions identified in the computation of the transition matrix and
   (c) the invariant space vectors

The first phase of the algorithm computes a transition matrix, A, and auxiliary initial conditions, Z. The second phase combines left invariant space vectors associated with large eigenvalues of A with the auxiliary initial conditions to produce the matrix Q characterizing the saddle point solution. Provided the right hand half of Q is invertible, the algorithm computes the matrix B, an autoregressive representation of the unique saddle point solution.

The Anderson-Moore methodology does not explicitly distinguish between predetermined and non-predetermined variables. The algorithm assumes that history fixes the values of all variables dated prior to time t and that these initial conditions, the saddle point property terminal conditions, and the model equations determine all subsequent variable values.

3.2 King & Watson's Canonical Variables/System Reduction Method

(King and Watson, 1998) describe another method for solving rational expectations models. Appendix B.2 provides a synopsis of the model concepts and algorithm inputs and outputs. The algorithm consists of two parts: system reduction for efficiency and canonical variables solution for solving the saddle point problem. Although their paper describes how to accommodate arbitrary exogenous shocks, the MATLAB function does not return the relevant matrices.

King-Watson provide a MATLAB function, resolkw, that computes solutions. The MATLAB function transforms the original system to facilitate the canonical variables calculations. The mdrkw program computes the solution assuming the exogenous variables follow a vector autoregressive process.

Given:

$$AE(y_{t+1}) = By_t + Cx_t$$

system reduction produces an equivalent model of the form

$$0 = f_t + Kd_t + \Psi_f(\boldsymbol{F})E(x_t)$$
$$E(d_{t+1}) = Wd_t + \Psi_d(\boldsymbol{F})E(x_t)$$

Where d_t are the "dynamic" variables and f_t are the "flow" variables in the y_t vector.

The mdrkw program takes the reduced system produced by redkw and the decomposition of its dynamic subsystem computed by dynkw and computes the rational expectations solution. The computation can use either eigenvalue-eigenvector decomposition or Schur decomposition.

Appendix B.2.1 shows one way to compute the King-Watson solution using the Anderson-Moore algorithm. Appendix B.2.2 shows one way to compute the Anderson-Moore solution using the King-Watson algorithm.

4 Applications of the QZ Algorithm

Several authors exploit the properties of the Generalized Schur Form (Golub and van Loan, 1989).

Theorem 1 *The Complex Generalized Schur Form* – If A and B are in $\mathcal{C}^{n\times n}$, then there exist unitary Q and Z such that $Q^H AZ = T$ and $Q^H BZ = S$ are upper triangular. If for some k, t_{kk} and s_{kk} are both zero, then $\lambda(A,B) = \mathcal{C}$. Otherwise, $\lambda(A,B) = \{\frac{t_{ii}}{s_{ii}} : s_{ii} \neq 0\}$

The algorithm uses the QZ decomposition to recast equation 5 in a canonical form that makes it possible to solve the transformed system "forward" for endogenous variables consistent with arbitrary values of the future exogenous variables.

4.1 Sims' QZ Method

(Sims, 1996) describes the QZ Method. His algorithm solves a linear rational expectations model of the form:

$$\Gamma_0 y_t = \Gamma_1 y_{t-1} + C + \Psi z_t + \Pi\eta_t \tag{5}$$

where $t = 1, 2, 3, \cdots, \infty$ and C is a vector of constants, z_t is an exogenously evolving, possibly serially correlated, random disturbance, and η_t is an expectational error, satisfying $E_t\eta_{t+1} = 0$.

Here, as with all the algorithms except the Anderson-Moore algorithm, one must cast the model in a form with one lag and no leads. This can be problematic for models with more than a couple of equations.

Appendix B.3 summarizes the Sims' QZ method model concepts and algorithm inputs and outputs.

The Π designation of expectational errors identifies the "predetermined" variables. The Anderson-Moore technique does not explicitly require the identification of expectational errors. In applying the Anderson-Moore technique, one chooses the time subscript of the variables that appear in the equations. All predetermined variables have historical values available through time $t-1$. The evolution of the solution path can have no effect on any variables dated $(t-1)$ or earlier. Future model values may influence time t values of any variable.

Appendix B.3.1 shows one way to transform the problem from Sims' form to Anderson-Moore form and how to reconcile the solutions. For the sake of comparison, the Anderson-Moore transformation adds $L\theta$ variables and the same number of equations, setting future expectation errors to zero.

Appendix B.3.2 shows one way to transform the problem from Anderson-Moore form to Sims form.

4.2 Klein's Approach

(Klein, 1999) describes another method. Appendix B.4 summarizes the model concepts and algorithm inputs and outputs.

The algorithm uses the Generalized Schur Decomposition to decouple backward and forward variables of the transformed system.

Although the MATLAB version does not provide solutions for autoregressive exogenous variables, one can solve the autoregressive exogenous variables problem by augmenting the system. The MATLAB program does not return matrices for computing the impact of arbitrary exogenous factors.

Appendix B.4.1 describes one way to recast a model from a form suitable for Klein into a form for the Anderson-Moore algorithm. Appendix B.4.2 describes one way to recast a model from a form suitable for the Anderson-Moore methodology into a form for the Klein Algorithm.

5 Applications of the Matrix Polynomial Approach

Several algorithms rely on determining a matrix C satisfying

$$H_1 C^2 + H_0 C + H_{-1} = 0. \tag{6}$$

They employ linear algebraic techniques to solve this quadratic equation. Generally there are many solutions.

When the homogeneous linear system has a unique saddle-path solution, the Anderson-Moore algorithm constructs the unique matrix $C_{AM} = B$ that satisfies the quadratic matrix equation and has all roots inside the unit circle.

$$H_1 C_{AM}^2 + H_0 C_{AM} + H_{-1} = 0$$

5.1 Binder & Pesaran's Method

(Binder and Pesaran, 1994) describe another method.

According to Binder & Pesaran(1994), under certain conditions, the unique stable solution, if it exists, is given by:

$$x_t = C x_{t-1} + \sum_{i=0}^{\infty} F^i E(w_{t+1})$$

where

$$F = (I - BC)^{-1} B$$

and C satisfies a quadratic equation like equation 6.

Their algorithm consists of a "recursive" application of the linear equations defining the relationships between C, H and F.

Appendix B.5.1 describes one way to recast a model from a form suitable for Binder-Pesaran into a form for the Anderson-Moore algorithm. Appendix B.5.2 describes one way to recast a model from a form suitable for the Anderson-Moore methodology into a form for the Binder-Pesaran Algorithm.

5.2 Uhlig's Technique

(Uhlig, 1999) describes another method. The algorithm uses generalized eigenvalue calculations to obtain a solution for the matrix polynomial equation.

One can view the Uhlig technique as preprocessing of the input matrices to reduce the dimension of the quadratic matrix polynomial. It turns out that once the simplification has been done, the Anderson-Moore algorithm computes the solution to the matrix polynomial more efficiently than the approach adopted in Uhlig's algorithm.

Uhlig's algorithm operates on matrices of the form:

$$\begin{bmatrix} B & 0 & A & C & 0 & 0 \\ H & 0 & G & K & F & J \end{bmatrix}$$

Uhlig in effect pre-multiplies the equations by the matrix

$$\begin{bmatrix} C^0 & 0 \\ C^+ & \\ -KC^+ & I \end{bmatrix}$$

where

$$C^0C = 0, C^+ = (C^TC)^{-1}C^T$$

to get

$$\begin{bmatrix} C^0B & 0 & C^0A & 0 & 0 & 0 \\ C^+B & 0 & C^+A & I & 0 & 0 \\ H - KC^+B & 0 & G - KC^+A & 0 & F & J \end{bmatrix}$$

one can imagine leading the second block of equations by one period and using them to annihilate J to get

$$\begin{bmatrix} 0 & 0 & C^0B & 0 & C^0A & 0 \\ H - KC^+B & 0 & G - KC^+A - JC^+B & 0 & F - JC^+A & 0 \\ 0 & 0 & C^+B & 0 & C^+A & I \end{bmatrix}$$

This step in effect decouples the second set of equations making it possible to investigate the asymptotic properties by focusing on a smaller system.

$$\begin{bmatrix} 0 & C^0B & C^0A \\ H - KC^+B & G - KC^+A - JC^+B & F - JC^+A \end{bmatrix}$$

Uhlig's algorithm undertakes the solution of a quadratic equation like equation 6 with

$$H_1 = \begin{bmatrix} C^0A \\ F - JC^+A \end{bmatrix}, H_0 = \begin{bmatrix} C^0B \\ G - KC^+A - JC^+B \end{bmatrix}, H_{-1} = \begin{bmatrix} 0 \\ H - KC^+B \end{bmatrix}$$

Appendix B.6.2 describes one way to recast a model from a form suitable for Uhlig into a form for the Anderson-Moore algorithm. Appendix B.6.1 describes one way to recast a model from a form suitable for the Anderson-Moore methodology into a form for the Uhlig Algorithm.

6 Comparisons

Section 2 identified B, ϑ, ϕ and F as potential outputs of a linear rational expectation algorithm. Most of the implementations do not compute each of the potential outputs. Only Anderson-Moore and Binder-Pesaran provide all four outputs (See Table 6).

Generally, the implementations make restrictions on the form of the input. Most require the user to specify models with at most one lag or one lead. Only Anderson-Moore explicitly allows multiple lags and leads.

Each of the authors provides small illustrative models along with their MATLAB code. The next two sections present results from applying all the algorithms to each of the example models.

Table 1: Modeling Features

Technique	B	ϑ	ϕ, F	Usage Notes
Anderson-Moore	✔	✔	✔	Allows multiple lags and leads. Has modeling language.
King & Watson	✔	✔		one lead, no lags
Sims	✔		✔	one lag, no leads
Klein	✔	✔		one lead, no lags
Binder-Peseran	✔	✔	✔	one lag, one lead; $\hat{C}$ must be non-singular
Uhlig	✔	✔		one lag, one lead; constraint involving choice of "jump" variables and rank condition on C.

Note:

- The Klein and Uhlig procedures compute ϑ by augmenting linear system
- For the Uhlig procedure one must choose "jump" variables to guarantee that the C matrix has full rank.

6.1 Computational Efficiency

Nearly all the algorithms successfully computed solutions for all the examples. Each of the algorithms, except Binder-Pesaran's, successfully computed solutions for all of Uhlig's examples. Uhlig's algorithm failed to provide a solution for the given parametrization of one of King's examples. However, Binder-Pesaran's and Uhlig's routines would likely solve alternative parametrization of the models that had convergence problems.

Tables 2-4 present the MATLAB-reported floating point operations (flops) counts for each of the algorithms applied to the example models.

The first column of each table identifies the example model. The second column provides the flops required by the Anderson-Moore algorithm to compute B followed by the flops required to compute B, ϑ, ϕ, and F. Columns three through seven report the flops required by each algorithm divided by the flops required by the Anderson-Moore algorithm for a given example model.

Note that the Anderson-Moore algorithm typically required a fraction of the number of flops required by the other algorithms. For example, King-Watson's algorithm required more than three times the flops required by the Anderson-Moore algorithm for the first Uhlig example. In the first row, one observes that Uhlig's algorithm required only 92% of the number of flops required by the Anderson-Moore algorithm, but this is the only instance where an alternative to the Anderson-Moore algorithm required fewer flops.

In general, Anderson-Moore provides solutions with the least computational effort. There were only a few cases where some alternative had approximately the same number of floating point operations. The efficiency advantage was especially pronounced for larger models. King-Watson generally used twice to three times the number of floating point operations. Sims generally used thirty times the number of floating point operations – never fewer than Anderson-Moore, King-Watson or Uhlig. It had about the same performance as Klein. Klein generally used thirty times the number of floating point operations. It never used fewer than Anderson-Moore, King-Watson or Uhlig. Binder-Pesaran was consistently the most computationally expensive algorithm. It generally used hundreds of times more floating point operations. In one case, it took as many as 100,000 times the number of floating point operations. Uhlig generally used about twice the flops of Anderson-Moore even for small models and many more flops for larger models.

Table 5 presents a comparison of the original Uhlig algorithm to a version using Anderson-Moore to solve the quadratic polynomial equation. Employing the Anderson-Moore algorithm speeds the computation. The difference was most dramatic for larger models.

6.2 Numerical Accuracy

Tables 6-11 presents the MATLAB relative errors. I have employed a symbolic algebra version of the Anderson-Moore algorithm to compute solutions to high precision[4]. Although ϑ and F are relatively simple linear transformations of B, each of the authors uses radically different methods to compute these quantities. I then compare the matrices computed in MATLAB by each algorithm to the high precision solution.

Anderson-Moore always computed the correct solution and in almost every case produced the most accurate solution. Relative errors were on the order of 10^{-16}. King-Watson always computed the correct solution, but produced solutions with relative errors generally 3 times the size of the Anderson-Moore algorithm. Sims always computed correct solutions but produced solutions with relative errors generally 5 times the size of the Anderson-Moore algorithm. [5] Sim's F calculation produced errors that were 20 times the size of the Anderson-Moore relative errors. Klein always computed the correct solution but produced solutions with relative errors generally 5 times the size of the Anderson-Moore algorithm.

Uhlig provides accurate solutions with relative errors about twice the size of the Anderson-Moore algorithm for each case for which it converges. It cannot provide a solution for King's example 3 for the particular parametrization I employed. I did not explore alternative parametrizations. For the ϑ computation, the results were similar. The algorithm was unable to compute ϑ for King example 3. Errors were generally 10 times the size of Anderson-Moore relative errors.

Binder-Pesaran converges to an incorrect value for three of the Uhlig examples: example 3, 6 and example 7. In each case, the resulting matrix solves the quadratic matrix polynomial, but the particular solution has an eigenvalue greater than one in magnitude even though an alternative matrix solution exists with eigenvalues less than unity. For Uhlig's example 3, the algorithm diverges and produces a matrix with NaN's. Even when the algorithm converges to approximate the correct solution, the errors are much larger than the other algorithms. One could tighten the convergence criterion at the expense of increasing computational time, but the algorithm is already the slowest of the algorithms evaluated. Binder-Pesaran's algorithm does not converge for either of Sims' examples. The algorithm provides accurate answers for King & Watson's examples. Although the convergence depends on the particular parametrization, I did not explore alternative parametrization when the algorithm's did not converge. The ϑ and F results were similar to the B results. The algorithm was unable to compute H for Uhlig 3 in addition to Uhlig 7. It computed the wrong value for Uhlig 6. It was unable to compute values for either of Sims's examples.

7 Conclusions

A comparison of the algorithms reveals that:

- For models satisfying the Blanchard-Kahn conditions, the algorithms provide equivalent solutions.
- The Anderson-Moore algorithm proved to be the most accurate.
- Using the Anderson-Moore algorithm to solve the quadratic matrix polynomial equation improves the performance of both Binder-Pesaran's and Uhlig's algorithms.
- While the Anderson-Moore, Sims and Binder-Pesaran approaches provide matrix output for accommodating arbitrary exogenous processes, the King-Watson and Uhlig implementations only provide solutions for VAR exogenous process.[6] Fortunately, there are straightforward formulae for augmenting

[4] I computed exact solutions when this took less than 5 minutes and solutions correct to 30 decimal places in all other cases.

[5] To compare F for Sims note that

$$\Psi = I_L$$
$$F = (\Theta_y.\Theta_f)\Phi_{exact}^{-1}$$

[6] I modified Klein's MATLAB version to include this functionality by translating the approach he used in his Gauss version.

the King-Watson, Uhlig and Klein approaches with the matrices characterizing the impact of arbitrary shocks.

- The Anderson-Moore algorithm requires fewer floating point operations to achieve the same result. This computational advantage increases with the size of the model.
- The Anderson-Moore suite of programs provides a simple modeling language for developing models. In addition, the Anderson-Moore algorithm requires no special treatment for models with multiple lags and leads. To use each of the other algorithms, one must cast the model in a form with at most one lead or lag. This can be tedious and error prone task for models with more than a couple of equations.

8 Bibliography

Gary Anderson. A reliable and computationally efficient algorithm for imposing the saddle point property in dynamic models. Unpublished Manuscript, Board of Governors of the Federal Reserve System. Downloadable copies of this and other related papers at http://www.federalreserve.gov/pubs/oss/oss4/aimindex.html, 1997.

Gary Anderson and George Moore. An efficient procedure for solving linear perfect foresight models. 1983.

Gary Anderson and George Moore. A linear algebraic procedure for solving linear perfect foresight models. *Economics Letters*, (3), 1985. URL `http://www.federalreserve.gov/pubs/oss/oss4/aimindex.html`.

Michael Binder and M. Hashem Pesaran. Multivariate rational expectations models and macroeconometric modelling: A review and some new results. URL `http://www.inform.umd.edu/EdRes/Colleges/BSOS/Depts/Economics/mbinder/research/matlabresparse.html`. Seminar Paper, May 1994.

Olivier Jean Blanchard and C. Kahn. The solution of linear difference models under rational expectations. *Econometrica*, 48, 1980.

Laurence Broze, Christian Gouriéroux, and Ariane Szafarz. Solutions of multivariate rational expectations models. *Econometric Theory*, 11:229–257, 1995.

Gene H. Golub and Charles F. van Loan. *Matrix Computations*. Johns Hopkins, 1989.

Robert G. King and Mark W. Watson. The solution of singular linear difference systems under rational expectations. *International Economic Review*, 39(4):1015–1026, November 1998. URL `http://www.people.virginia.edu/~rgk4m/kwre/kwre.html`.

Paul Klein. Using the generalized schur form to solve a multivariate linear rational expectations model. *Journal of Economic Dynamics and Control*, 1999. URL `http://www.iies.su.se/data/home/kleinp/homepage.htm`.

Christopher A. Sims. Solving linear rational expectations models. URL `http://www.econ.yale.edu/~sims/#gensys`. Seminar paper, 1996.

Harald Uhlig. A toolkit for analyzing nonlinear dynamic stochastic models easily. URL `http://cwis.kub.nl/~few5/center/STAFF/uhlig/toolkit.dir/toolkit.htm`. User's Guide, 1999.

Peter A. Zadrozny. An eigenvalue method of undetermined coefficients for solving linear rational expectations models. *Journal of Economic Dynamics and Control*, 22:1353–1373, 1998.

Appendix A The Anderson-Moore Algorithm(s)

Algorithm 1

Given H, compute the unconstrained autoregression.
funct $\mathcal{F}_1(H)$ $\equiv$
  $k := 0$
  $\mathcal{Z}^0 := \emptyset$
  $\mathcal{H}^0 := H$
  $\Gamma := \emptyset$
  while $\mathcal{H}_\theta^k$ is singular $\cap\, rows(\mathcal{Z}^k) < L(\tau+\theta)$
    do
    $U^k = \begin{bmatrix} U_Z^k \\ U_N^k \end{bmatrix} :=$ rowAnnihilator$(\mathcal{H}_\theta^k)$
    $\mathcal{H}^{k+1} := \begin{bmatrix} 0 & U_Z^k\mathcal{H}_\tau^k & \dots & U_Z^k\mathcal{H}_{\theta-1}^k \\ U_N^k\mathcal{H}_\tau^k & & \dots & U_N^k\mathcal{H}_\theta^k \end{bmatrix}$
    $\mathcal{Z}^{k+1} := \begin{bmatrix} & \mathcal{Q}^k & \\ U_Z^k\mathcal{H}_\tau^k & \dots & U_Z^k\mathcal{H}_{\theta-1}^k \end{bmatrix}$
    $k := k+1$
  od
  $\Gamma = -H_\theta^{-1}\begin{bmatrix} H_{-\tau} & \dots & H_{\theta-1} \end{bmatrix}$
  $A = \begin{bmatrix} 0 & I \\ \Gamma & \end{bmatrix}$
  return$\{\begin{bmatrix} \mathcal{H}_{-}^k\tau & \dots & \mathcal{H}_\theta^k \end{bmatrix}, A, \mathcal{Z}^k\}$
.

Algorithm 2

Given $V, Z^{\sharp,*}$,
funct $\mathcal{F}_2(A, Z^{\sharp,*})$
  Compute V, the vectors spanning the left
   invariant space associated with eigenvalues
   greater than one in magnitude
  $Q := \begin{bmatrix} Z^{\sharp,*} \\ V \end{bmatrix}$
  return$\{Q\}$
.

Algorithm 3

Given Q,
funct $\mathcal{F}_3(Q)$
  cnt $= noRows(Q)$
  return $\begin{cases} \{Q, \infty\} & \text{cnt} < L\theta \\ \{Q, 0\} & \text{cnt} > L\theta \\ \{Q, \infty\} & (Q_R singular) \\ \{B = -Q_R^{-1}Q_L, 1\} & otherwise \end{cases}$
.

Appendix B Model Concepts

The following sections present the inputs and outputs for each of the algorithms for the following simple example:

$$V_{t+1} = (1+R)V_t - D_{t+1}$$
$$D = (1-\delta)D_{t-1}$$

Appendix B.1 Anderson-Moore

Inputs		
$$\sum_{i=-\tau}^{\theta} H_i x_{t+i} = \Psi z_t$$		
Model Variable	Description	Dimensions
x_t	State Variables	$L(\tau+\theta)\times 1$
z_t	Exogenous Variables	$M \times 1$
θ	Longest Lead	1×1
τ	Longest Lag	1×1
H_i	Structural Coefficients Matrix	$(L \times L)(\tau+\theta+1)$
Ψ	Exogenous Shock Coefficients Matrix	$L \times M$
Υ	Optional Exogenous VAR Coefficients Matrix($z_{t+1} = \Upsilon z_t$)	$M \times M$

Outputs		
$$x_t = B \begin{bmatrix} x_{t-\tau} \\ \vdots \\ x_{t-1} \end{bmatrix} + \begin{bmatrix} 0 & \dots 0 & I \end{bmatrix} \sum_{s=0}^{\infty} (F^s \begin{bmatrix} 0 \\ \Phi\Psi z_{t+s} \end{bmatrix})$$		
Model Variable	Description	Dimensions
B	reduced form coefficients matrix	$L \times L(\tau+\theta)$
Φ	exogenous shock scaling matrix	$L \times L$
F	exogenous shock transfer matrix	$L\theta \times L\theta$
ϑ	autoregressive shock transfer matrix when $z_{t+1} = \Upsilon z_t$ the infinite sum simplifies to give $x_t = B \begin{bmatrix} x_{t-\tau} \\ \vdots \\ x_{t-1} \end{bmatrix} + \vartheta z_t$	$L \times M$

Anderson-Moore input:

AIM Modeling Language Input

```
1   MODEL> FIRMVALUE
2   ENDOG>
3   V
4   DIV
5   EQUATION> VALUE
6   EQ> LEAD(V,1) = (1+R)*V - LEAD(DIV,1)
7   EQUATION> DIVIDEND
8   EQ> DIV = (1-DELTA)*LAG(DIV,1)
9   END
```

Parameter File Input

```
1   DELTA=0.3;
2   R=0.1;
3   psi=[4. 1.;3. -2.];
4   upsilon=[0.9 0.1;0.05 0.2];
```

$$H = \begin{bmatrix} 0 & 0 & -1.1 & 0 & 1 & 1. \\ 0 & -0.7 & 0 & 1 & 0 & 0 \end{bmatrix}, \Psi = \begin{bmatrix} 4. & 1. \\ 3. & -2. \end{bmatrix}, \Upsilon \begin{bmatrix} 0.9 & 0.1 \\ 0.05 & 0.2 \end{bmatrix}$$

produces output:

$$B = \begin{bmatrix} 0. & 1.225 \\ 0. & 0.7 \end{bmatrix} F = \begin{bmatrix} 0.909091 & 0.909091 \\ 0. & 0. \end{bmatrix} \Phi = \begin{bmatrix} -0.909091 & 1.75 \\ 0. & 1. \end{bmatrix}$$

$$\Phi\Psi = \begin{bmatrix} 1.61364 & -4.40909 \\ 3. & -2. \end{bmatrix} \vartheta = \begin{bmatrix} 21.0857 & -3.15714 \\ 3. & -2. \end{bmatrix}$$

Usage Notes for Anderson-Moore Algorithm

1. "Align" model variables so that the data history (without applying model equations), completely determines all of x_{t-1}, but none of x_t.
2. Develop a "model file" containing the model equations written in the "AIM modeling language"
3. Apply the model pre-processor to create MATLAB programs for initializing the algorithm's input matrix,(H). Create Ψ and, optionally, Υ matrices.
4. Execute the MATLAB programs to generate B, ϕ, F and optionally ϑ

Users can obtain code for the algorithm and the preprocessor from the author[7]

[7] http://www.bog.frb.fed.us/pubs/oss/oss4/aimindex.html July, 1999.

Appendix B.2 King-Watson

Inputs		
$$AE_t y_{t+1} = By_t + \sum_{i=0}^{\theta} C_i E_t(x_{t+i})$$ $$x_t = Q\delta_t$$ $$\delta_t = \rho\delta_{t-1} + G\epsilon_t$$ $$y_t = \begin{bmatrix}\Lambda_t \\ k_t\end{bmatrix}$$		
Model Variable	Description	Dimensions
θ	longest lead	1×1
y_t	Endogenous Variables	$m \times 1$
Λ_t	Non-predetermined Endogenous Variables	$(m-p) \times 1$
k_t	Predetermined Endogenous Variables	$p \times 1$
x_t	Exogenous Variables	$n_x \times 1$
δ_t	Exogenous Variables	$n_\delta \times 1$
A	Structural Coefficients Matrix associated with lead endogenous variables, y_{t+1}	$m \times m$
B	Structural Coefficients Matrix associated with contemporaneous endogenous variables, y_t	$m \times m$
C_i	Structural Coefficients Matrix associated with contemporaneous and lead exogenous variables, x_t	$m \times n$
Q	Structural Coefficients Matrix associated with contemporaneous exogenous variables, x_t	$n_x \times n_\delta$
ρ	Vector Autoregression matrix for exogenous variables	$n_\delta \times n_\delta$
G	Matrix multiplying Exogenous Shock	$n_\delta \times 1$

Outputs		
$$y_t = \Pi s_t$$ $$s_t = M s_{t-1} + \bar{G}\epsilon_t$$ $$s_t = \begin{bmatrix}k_t \\ \delta_t\end{bmatrix}$$		
Model Variable	Description	Dimensions
s_t	Exogenous Variables and predetermined variables	$(n_x + p) \times 1$
Π	Matrix relating endogenous variables to exogenous and predetermined variables	$m \times (p + n_x)$
M		$(p + n_x) \times (p + n_x)$
$\bar{G}$	Matrix multiplying Exogenous Shock	$(n_x + p) \times l$

King-Watson input:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1. \\ 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1.1 & 0 \\ 0 & -0.7 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 4. & 1. \\ 3. & -2. \end{bmatrix},$$

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \rho = \begin{bmatrix} 0.9 & 0.1 \\ 0.05 & 0.2 \end{bmatrix}, G = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

produces output:

$$\Pi = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. \\ 0. & 1.225 & -21.0857 & 3.15714 \\ 0. & 0.7 & -3. & 2. \end{bmatrix}, M = \begin{bmatrix} 0. & 1.225 & -21.0857 & 3.15714 \\ 0. & 0.7 & -3. & 2. \\ 0. & 0. & 0.9 & 0.1 \\ 0. & 0. & 0.05 & 0.2 \end{bmatrix}$$

Usage Notes for King-Watson Algorithm

1. Identify predetermined variables
2. Cast the model in King-Watson form: endogenous variable must have at most one lead and no lag.
3. Create matlab "system" and "driver" programs generating the input matrices. "system" generates $(A, B, C_i, nx = n_x, ny = m)$, and a matlab vector containing indices corresponding to predetermined variables. "driver" generates (Q, ρ, G).
4. Call resolkw with sytem and driver filenames as inputs to generate $\Pi, M, \bar{G}$

Appendix B.2.1 King-Watson to Anderson-Moore

Obtaining Anderson-Moore inputs from King-Watson inputs

$$\boldsymbol{\theta} := \theta, \boldsymbol{x_t} := \begin{bmatrix} \Lambda_t \\ k_t \\ x_t \\ \delta_t \end{bmatrix}, \boldsymbol{z_t} := \begin{bmatrix} 0 \\ 0 \\ \epsilon_t \end{bmatrix},$$

$$\boldsymbol{\Psi} := \begin{bmatrix} 0 & 0 & G \end{bmatrix} \boldsymbol{\Upsilon} := \begin{bmatrix} 0 & 0 & \rho \end{bmatrix}$$

$$\begin{bmatrix} B_1 & B_2 \end{bmatrix} := B, \begin{bmatrix} A_1 & A_2 \end{bmatrix} := A$$

$$\boldsymbol{H} := \begin{bmatrix} 0 & -B_2 & 0 & 0 & -B_1 & A_2 & -C_0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & -I & Q & \\ 0 & 0 & 0 & 0 & -\rho & 0 & 0 & I & \end{bmatrix}$$

$$\begin{bmatrix} A_1 & 0 & -C_1 & 0 & & 0 & 0 & -C_\theta & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ -G & 0 & 0 & 0 & & 0 & 0 & 0 & 0 \end{bmatrix}$$

Obtaining King-Watson outputs from Anderson-Moore outputs

$$\begin{bmatrix} 0 & \hat{\Pi}_{\Lambda k} & 0 & \hat{\Pi}_{\Lambda\delta} \\ 0 & M_{kk} & 0 & M_{k\delta} \\ 0 & \hat{\Pi}_{xk} & 0 & \hat{\Pi}_{x\delta} \\ 0 & 0 & 0 & \rho \end{bmatrix} := B$$

$$\boldsymbol{M} := \begin{bmatrix} M_{kk} & M_{k\delta} \\ 0 & \rho \end{bmatrix}$$

$$\boldsymbol{\Pi} := \begin{bmatrix} \hat{\Pi}_{\Lambda k} & \hat{\Pi}_{\Lambda\delta} \end{bmatrix} M^{-1}$$

$$\bar{\boldsymbol{G}} := \begin{bmatrix} 0 & 0 & 0 & I \end{bmatrix} \Phi\Psi$$

Appendix B.2.2 Anderson-Moore to King-Watson

Obtaining King-Watson inputs from Anderson-Moore inputs

$$\boldsymbol{y_t} := \begin{bmatrix} x_{t-\tau} \\ \vdots \\ x_{t+\theta-1} \end{bmatrix}, \boldsymbol{\Lambda_t} := \begin{bmatrix} x_t \\ \vdots \\ x_{t+\theta-1} \end{bmatrix} \boldsymbol{k_t} := \begin{bmatrix} x_{t-\tau} \\ \vdots \\ x_{t-1} \end{bmatrix}, \boldsymbol{x_t} := z_t$$

$$\boldsymbol{A} := \begin{bmatrix} I & & & 0 \\ & \ddots & & \vdots \\ & & I & 0 \\ 0 & \dots & 0 & -H_\theta \end{bmatrix}$$

$$\boldsymbol{B} := \begin{bmatrix} 0 & I & & \\ \vdots & & \ddots & \\ 0 & & & I \\ H_{-\tau} & H_{-\tau+1} & \dots & H_{\theta-1} \end{bmatrix}$$

$$\boldsymbol{C} := \begin{bmatrix} 0 \\ \Psi \end{bmatrix}, \boldsymbol{Q} := I, \boldsymbol{\rho} := 0, \boldsymbol{G} := 0, \boldsymbol{p} := L\tau, \boldsymbol{\theta} := 1, \boldsymbol{m} := L(\tau+\theta)$$

Obtaining Anderson-Moore outputs from King-Watson outputs

$$\begin{bmatrix} \boldsymbol{B} & \vartheta \\ 0 & \rho \end{bmatrix} := M, \begin{bmatrix} I & 0 \\ B_1 & \vartheta \\ \vdots & \vdots \\ B_\theta & B_{(\theta-1)R}\vartheta \end{bmatrix} := \Pi$$

Appendix B.3 Sims

Inputs		
$\Gamma_0 y_t = \Gamma_1 y_{t-1} + C + \Psi z_t + \Pi \eta_t$		
Model Variable	Description	Dimensions
y_t	State Variables	$L \times 1$
z_t	Exogenous Variables	$M_1 \times 1$
η_t	Expectational Error	$M_2 \times 1$
Γ_0	Structural Coefficients Matrix	$L \times L$
Γ_1	Structural Coefficients Matrix	$L \times L$
C	Constants	$L \times 1$
Ψ	Structural Exogenous Variables Coefficients Matrix	$L \times M_1$
Π	Structural Expectational Errors Coefficients Matrix	$L \times M_2$

Outputs		
$y_t = \Theta_1 y_{t-1} + \Theta_c + \Theta_0 z_t + \Theta_y \sum_{s=1}^{\infty} \Theta_f^{s-1} \Theta_z E_t z_{t+s}$		
Model Variable	Description	Dimensions
Θ_1		$L \times L$
Θ_c		$L \times 1$
Θ_0		$L \times M_1$
Θ_y		$L \times M_2$
Θ_f		$M_2 \times M_2$
Θ_z		$M_2 \times M_1$

Sims input:

$$\Gamma_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1.1 & 0 & 1 & 1. \\ 0 & 1 & 0 & 0 \end{bmatrix}, \Gamma_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0.7 & 0 & 0 \end{bmatrix}, C = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\Psi = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 4. & 1. \\ 3. & -2. \end{bmatrix}, \Pi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

produces output:

$$\Theta_c \begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \end{bmatrix}, \Theta_0 = \begin{bmatrix} 1.61364 & -4.40909 \\ 3. & -2. \\ 3.675 & -2.45 \\ 2.1 & -1.4 \end{bmatrix}, \Theta_f = \begin{bmatrix} 0.909091 & 1.23405 \\ 0. & 0. \end{bmatrix},$$

$$\Theta_y = \begin{bmatrix} 1.28794 & 1.74832 \\ -2.2204510^{-16} & -1.1102210^{-16} \\ 1.41673 & 0.702491 \\ -1.1102210^{-16} & 1.22066 \end{bmatrix}, \Theta_z = \begin{bmatrix} -0.0796719 & -2.29972 \\ 2.4577 & -1.63846 \end{bmatrix}$$

$$\Theta_y \Theta_z = \begin{bmatrix} 4.19421 & -5.82645 \\ -2.5516810^{-16} & 6.9254610^{-16} \\ 1.61364 & -4.40909 \\ 3. & -2. \end{bmatrix},$$

Usage Notes for Sims Algorithm

1. Identify predetermined variables
2. Cast the model in Sims form: at most 1 lag and 0 leads of endogenous variables.
3. Create the input matrices $\Gamma_0, \Gamma_1, C, \Psi,$ and Π
4. Call gensys to generate $\Theta_1, \Theta_c, \Theta_0, \Theta_y, \Theta_f, \Theta_z$

Appendix B.3.1 Sims to Anderson-Moore

Obtaining Anderson-Moore inputs from Sims inputs

$$\boldsymbol{x_{t+s}} := \begin{bmatrix} y_{t+s} \\ \eta_{t+s} \end{bmatrix} - x^*, \boldsymbol{z_t} := \begin{bmatrix} z_t \\ 1 \end{bmatrix}, \boldsymbol{\Psi} := \begin{bmatrix} \Psi & C \\ 0 & 0 \end{bmatrix}$$

$$\boldsymbol{H} := \begin{bmatrix} -\Gamma_1 & 0 & \Gamma_0 & -\Pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{bmatrix}$$

Obtaining Sims outputs from Anderson-Moore outputs

$$\boldsymbol{\Theta_1} := \begin{bmatrix} 0_{L(\tau-1)\times L} & I_{L(\tau-1)} & \\ B & & \\ \vdots & & 0_{L(\tau+\theta)\times L\theta} \\ B_\theta & & \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\Theta_0} & \boldsymbol{\Theta_c} \end{bmatrix} := \begin{bmatrix} 0_{L(\tau-1)\times L} \\ I \\ B_R \\ \vdots \\ B_{\theta R} \end{bmatrix} \Phi\Psi$$

$$\exists S \ni \Theta_f = SFS^{-1}, \Theta_y\Theta_z = \begin{bmatrix} 0_{L(\tau-1)\times L} \\ \hat{x}_t \\ \vdots \\ \hat{x}_{t+\theta+1} \end{bmatrix}$$

where $\hat{x_t}$'s come from iterating equation Appendix B.1 forward $\theta+1$ time periods with $z_{t+s} = 0, s \neq 1$ and $z_{t+1} = I$

Appendix B.3.2 Anderson-Moore to Sims

Obtaining Sims inputs from Anderson-Moore inputs

$$\boldsymbol{y_t} := \begin{bmatrix} x_{t-\tau} \\ \vdots \\ x_{t+\theta-1} \end{bmatrix}, \boldsymbol{z_t} := z_t$$

$$\boldsymbol{\Gamma_0} := \begin{bmatrix} I & & & 0 \\ & \ddots & & \vdots \\ & & I & 0 \\ 0 & H_0 & \dots & -H_\theta \end{bmatrix} \boldsymbol{\Gamma_1} := \begin{bmatrix} 0 & I & & & \\ \vdots & & \ddots & & \\ 0 & & & I & \\ H_{-\tau} & \dots & H_{-1} & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{\Pi} := \begin{bmatrix} 0_{L(\tau-1)\times L\theta} \\ I_{L\theta} \\ 0_{L\times L\theta} \end{bmatrix} \boldsymbol{\Psi} := \begin{bmatrix} 0_{L\tau\times L\theta} \\ \Psi \end{bmatrix}$$

Obtaining Anderson-Moore outputs from Sims outputs

$$\begin{bmatrix} \boldsymbol{B} & \end{bmatrix} := \Theta_1, \begin{bmatrix} \boldsymbol{\Phi\Psi} \end{bmatrix} := \Theta_y\Theta_f$$

Appendix B.4 Klein

Inputs		
$$A\mathcal{E}x_{t+1} = Bx_t + Cz_{t+1} + Dz_t$$ $$z_{t+1} = \phi z_t + \epsilon_t x_t = \begin{bmatrix} k_t \\ u_t \end{bmatrix}$$		
Model Variable	Description	Dimensions
x_t	Endogenous Variables	$L \times 1$
z_t	Exogenous Variables	$M \times 1$
n_k	The number of state Variables	$M \times 1$
k_t	State Variables	$n_k \times 1$
u_t	The Non-state Variables	$(L - n_k) \times 1$
A	Structural Coefficients Matrix for Future Variables	$L \times L$
B	Structural Coefficient Matrix for contemporaneous Variables	$L \times L$

Outputs		
$$u_t = Fk_t$$ $$k_t = Pk_{t-1}$$		
Model Variable	Description	Dimensions
F	Decision Rule	$(L - n_k) \times n_k$
P	Law of Motion	$n_k \times n_k$

Klein input:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1. \\ 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1.1 & 0 \\ 0 & -0.7 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 4. & 1. \\ 3. & -2. \end{bmatrix},$$

produces output:

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.05 & 0.2 \end{bmatrix}, F = \begin{bmatrix} -21.0857 & 3.15714 \\ -3. & 2. \end{bmatrix},$$

Usage Notes for Klein Algorithm

1. Identify predetermined variables. Order the system so that predetermined variables come last.
2. Create the input matrices A, B.[a]
3. Call solab with the input matrices and the number of predetermined variables to generate F and P

[a] The Gauss version allows additional functionality.

Appendix B.4.1 Klein to Anderson-Moore

Obtaining Anderson-Moore inputs from Klein inputs

$$\boldsymbol{\theta} := \theta, \boldsymbol{x_t} := \begin{bmatrix} k_t \\ u_t \\ z_t \end{bmatrix}, \boldsymbol{z_t} := \begin{bmatrix} z_t \\ z_{t+1} \end{bmatrix} \boldsymbol{\Psi} := \begin{bmatrix} D & C \end{bmatrix}, \Upsilon := \begin{bmatrix} \phi & \\ & \phi \end{bmatrix}$$

$$\begin{bmatrix} B_1 & B_2 \end{bmatrix} := B, \begin{bmatrix} A_1 & A_2 \end{bmatrix} := A$$

$$\boldsymbol{H} := \begin{bmatrix} 0 & -B_2 & 0 & 0 & -B_1 & A_2 & -C_0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & -I & Q & \dots \\ 0 & 0 & 0 & 0 & -\rho & 0 & 0 & I & \\ A_1 & 0 & -C_1 & 0 & & 0 & 0 & -C_\theta & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ -G & 0 & 0 & 0 & & 0 & 0 & 0 & 0 \end{bmatrix}$$

Obtaining Klein outputs from Anderson-Moore outputs

$$\begin{bmatrix} 0 & \hat{F}_{\Lambda k} & 0 & \hat{F}_{\Lambda \delta} \\ 0 & P_{kk} & 0 & P_{k\delta} \\ 0 & \hat{F}_{xk} & 0 & \hat{F}_{x\delta} \\ 0 & 0 & 0 & \rho \end{bmatrix} := B$$

$$\boldsymbol{P} := \begin{bmatrix} P_{kk} & P_{k\delta} \\ 0 & \rho \end{bmatrix}$$

$$\boldsymbol{F} := \begin{bmatrix} \hat{F}_{\Lambda k} & \hat{F}_{\Lambda \delta} \end{bmatrix} P^{-1}$$

$$\bar{\boldsymbol{G}} := \begin{bmatrix} 0 & 0 & 0 & I \end{bmatrix} \Phi \Psi$$

Appendix B.4.2 Anderson-Moore to Klein

Obtaining Klein inputs from Anderson-Moore inputs

$$\boldsymbol{y_t} := \begin{bmatrix} x_{t-\tau} \\ \vdots \\ x_{t+\theta-1} \end{bmatrix}, \boldsymbol{\Lambda_t} := \begin{bmatrix} x_t \\ \vdots \\ x_{t+\theta-1} \end{bmatrix} \boldsymbol{k_t} := \begin{bmatrix} x_{t-\tau} \\ \vdots \\ x_{t-1} \end{bmatrix}$$

$$\boldsymbol{A} := \begin{bmatrix} I & & & 0 \\ & \ddots & & \vdots \\ & & I & 0 \\ 0 & \dots & 0 & -H_\theta \end{bmatrix}$$

$$\boldsymbol{B} := \begin{bmatrix} 0 & I & & \\ \vdots & & \ddots & \\ 0 & & & I \\ H_{-\tau} & H_{-\tau+1} & \dots & H_{\theta-1} \end{bmatrix}$$

$$\boldsymbol{C} := \begin{bmatrix} 0 \\ \Psi \end{bmatrix}, \boldsymbol{Q} := I, \boldsymbol{\rho} := 0, \boldsymbol{G} := 0$$

Obtaining Anderson-Moore outputs from Klein outputs

$$\begin{bmatrix} 0 & I & 0 \\ B & & \Phi\Psi \\ 0 & & 0 \end{bmatrix} := P, \quad \begin{bmatrix} & I & & 0 \\ B_{1L} & & B_{1R} & \Phi\Psi \\ \vdots & & \vdots & \vdots \\ B_{\theta L} & & B_{\theta R} & B_{(\theta-1)R}\Phi\Psi \end{bmatrix} := F$$

Appendix B.5 Binder-Pesaran

Inputs		
$\hat{C}x_t = \hat{A}x_{t-1} + \hat{B}\mathcal{E}x_{t+1} + D_1 w_t + D_2 w_{t+1}$ $w_t = \Gamma w_{t-1}$		
Model Variable	Description	Dimensions
x_t	State Variables	$L \times 1$
w_t	Exogenous Variables	$M \times 1$
$\hat{A}$	Structural Coefficients Matrix	$L \times L$
$\hat{B}$	Exogenous Shock Coefficients Matrix	$L \times L$
$\hat{C}$	Exogenous Shock Coefficients Matrix	$L \times M$
D_1	Exogenous Shock Coefficients Matrix	$L \times M$
D_2	Exogenous Shock Coefficients Matrix	$L \times M$
Γ	Exogenous Shock Coefficients Matrix	$L \times M$
Outputs		
$x_t = Cx_{t-1} + Hw_t$ $x_t = Cx_{t-1} + \sum_{i=0}^{\infty} F^i E(w_{t+1})$		
Model Variable	Description	Dimensions
C	reduced form codifying convergence constraints	$L \times L(\tau + \theta)$
H	exogenous shock scaling matrix	$L \times M$

Binder-Peseran input:

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0.7 \end{bmatrix} B = \begin{bmatrix} -1 & -1. \\ 0 & 0 \end{bmatrix} C = \begin{bmatrix} -1.1 & 0 \\ 0 & 1 \end{bmatrix} D_1 = \begin{bmatrix} 4. & 1. \\ 3. & -2. \end{bmatrix}$$
$$D_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Gamma = \begin{bmatrix} 0.9 & 0.1 \\ 0.05 & 0.2 \end{bmatrix}$$

produces output:

$$C = \begin{bmatrix} 0. & 1.225 \\ 0. & 0.7 \end{bmatrix} H = \begin{bmatrix} 21.0857 & -3.15714 \\ 3. & -2. \end{bmatrix}$$

Usage Notes for Binder-Pesaran Algorithm

1. Cast model in Binder-Pesaran form: at most one lead and one lag of endogenous variables. The matrix $\hat{C}$ must be nonsingular.
2. Modify existing Matlab script implementing Binder-Pesaran's Recursive Method to create the input matrices $\hat{A}, \hat{B}, \hat{C}, D_1, D_2\Gamma$ and to update the appropriate matrices in the "while loop."[8]
3. Call the modified script to generate C and H and F.

Appendix B.5.1 Binder-Pesaran to Anderson-Moore

Obtaining Anderson-Moore inputs from Binder-Pesaran inputs

$$\boldsymbol{x_t} := x_t, \boldsymbol{z_t} := \begin{bmatrix} w_t \\ w_{t+1} \end{bmatrix}$$
$$\boldsymbol{H} := \begin{bmatrix} -\hat{A} & \hat{C} & \hat{B} \end{bmatrix}$$
$$\boldsymbol{\Upsilon} := \begin{bmatrix} \Gamma & \\ & \Gamma \end{bmatrix}, \boldsymbol{\Psi} := \begin{bmatrix} D_1 & D_2 \end{bmatrix}$$

Obtaining Binder-Pesaran outputs from Anderson-Moore outputs

$$\boldsymbol{C} := B, \boldsymbol{H} := \Phi\Psi$$

Appendix B.5.2 Anderson-Moore to Binder-Pesaran

Obtaining Binder-Pesaran inputs from Anderson-Moore inputs

$$\boldsymbol{A} := \begin{bmatrix} -I_{L(\tau-1)} & 0_{L(\tau-1)\times L(\theta-1)} \\ 0_{L(\theta-1)} & 0_{L(\theta-1)\times L(\theta-1)} \\ H_{-\tau} & 0_{L\times L(\tau+\theta-2)} \end{bmatrix}$$
$$\boldsymbol{B} := \begin{bmatrix} 0_{L(\tau-1)} & 0_{L(\tau-1)\times L(\theta-1)} \\ I_{L(\theta-1)\times L} & 0_{L(\theta-1)} \\ 0_{L\times L(\theta+\tau-2)} & H_\theta \end{bmatrix}$$
$$\boldsymbol{C} := \begin{bmatrix} I_{L(\tau-1)} & 0_{L(\tau-1)\times L(\theta-1)} & 0_{L(\tau-1)\times L} \\ 0_{L(\theta-1)\times L} & 0_{L(\theta-1)} & I_{L(\theta-1)\times L(\theta-1)} \\ H_{-\tau+1} & \cdots & H_{\theta-1} \end{bmatrix}$$
$$\boldsymbol{D_1} := \begin{bmatrix} 0 \\ \Psi \end{bmatrix}, D_2 = 0, \Gamma = 0$$

Obtaining Anderson-Moore outputs from Binder-Pesaran outputs

$$\begin{bmatrix} 0 \\ \Phi\Psi \\ B_R\Phi\Psi \\ \vdots \\ B_{(\theta-1)R}\Phi\Psi \end{bmatrix} := H$$
$$\begin{bmatrix} 0 & I & 0 \\ B & & 0 \\ \vdots & & 0 \\ B_\theta & & 0 \end{bmatrix} := C$$

Appendix B.6 Uhlig

Inputs		
$$Ax_t + Bx_{t-1} + Cy_t + Dz_t = 0$$ $$\mathcal{E}(Fx_{t+1} + Gx_t + Hx_{t-1} + Jy_{t+1} + Ky_t + Lz_{t+1} + Mz_t)$$ $$z_{t+1} = Nz_t + \epsilon_{t+1}$$ $$\mathcal{E}\epsilon_{t+1} = 0$$		
Model Variable	Description	Dimensions
x_t	State Variables	$m \times 1$
y_t	Endogenous "jump" Variables	$n \times 1$
z_t	Exogenous Variables	$k \times 1$
A, B	Structural Coefficients Matrix	$l \times m$
C	Structural Coefficients Matrix	$l \times n$
D	Structural Coefficients Matrix	$l \times k$
F, G, H	Structural Coefficients Matrix	$(m + n - l) \times m$
J, K	Structural Coefficients Matrix	$(m + n - l) \times n$
L, M	Structural Coefficients Matrix	$m + n - l \times k$
N	Structural Coefficients Matrix	$k \times k$

Outputs		
$$x_t = Px_{t-1} + Qz_t$$ $$y_t = Rx_{t-1} + Sz_t$$		
Model Variable	Description	Dimensions
P		$m \times m$
Q		$m \times k$
R		$n \times m$
S		$n \times k$

For Uhlig cannot find C with the appropriate rank condition without augmenting the system with a "dummy variable" and equation like $W_t = D_t + V_t$.
Uhlig input:

$$A = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} B = \begin{bmatrix} -0.7 & 0 \\ 0 & 0 \end{bmatrix} C = \begin{bmatrix} 0 \\ 1 \end{bmatrix} D = \begin{bmatrix} -3. & 2. \\ 0 & 0 \end{bmatrix}$$
$$F = \begin{bmatrix} 1. & 0 \end{bmatrix} G = \begin{bmatrix} 0 & 0 \end{bmatrix} H = \begin{bmatrix} 0 & 0 \end{bmatrix} J = \begin{bmatrix} 1 \end{bmatrix} K = \begin{bmatrix} -1.1 \end{bmatrix}$$
$$L = \begin{bmatrix} 0 & 0 \end{bmatrix} M = \begin{bmatrix} -4. & -1. \end{bmatrix} N = \begin{bmatrix} 0.9 & 0.1 \\ 0.05 & 0.2 \end{bmatrix}$$

produces output:

$$P = \begin{bmatrix} 0.7 & 0. \\ 1.925 & 6.1629810^{-33} \end{bmatrix} Q = \begin{bmatrix} 3. & -2. \\ 24.0857 & -5.15714 \end{bmatrix}$$
$$R = \begin{bmatrix} 1.225 & 6.1629810^{-33} \end{bmatrix} S = \begin{bmatrix} 21.0857 & -3.15714 \end{bmatrix}$$

Usage Notes for Uhlig Algorithm

1. Cast model in Uhlig Form. One must be careful to choose endogenous "jump" and and "non jump" variables to guarantee that the C matrix is appropriate rank. The rank null space must be $(l-n)$ where l is the number of equations with no lead variables,(the row dimension of C) and n is the total number of "jump" variables(thee column dimension of C).
2. Create matrices $A, B, C, D, F, G, H, J, K, L, M, N$
3. Call the function do_it to generate P, Q, R, S, W

Appendix B.6.1 Uhlig to Anderson-Moore

Obtaining Anderson-Moore inputs from Uhlig inputs

$$\boldsymbol{x_t} := \begin{bmatrix} x_t \\ y_t \end{bmatrix} \boldsymbol{H} := \begin{bmatrix} B & 0 & A & C & D & 0 & 0 \\ H & 0 & G & K & F & J & 0 \end{bmatrix}$$
$$\boldsymbol{\Upsilon} := N, \boldsymbol{\Psi} := L,$$

Obtaining Uhlig outputs from Anderson-Moore outputs

$$\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{P} \end{bmatrix} := B, \begin{bmatrix} \boldsymbol{Q} \\ \boldsymbol{S} \end{bmatrix} := \Phi\Psi$$

Appendix B.6.2 Anderson-Moore to Uhlig

Obtaining Uhlig inputs from Anderson-Moore inputs

$$\boldsymbol{A} := \begin{bmatrix} -I_{L(\tau-1)} & 0_{L(\tau-1)\times L(\theta-1)} \\ 0_{L(\theta-1)} & 0_{L(\theta-1)\times L(\theta-1)} \\ H_{-\tau} & 0_{L\times L(\tau+\theta-2)} \end{bmatrix}$$
$$\boldsymbol{B} := \begin{bmatrix} 0_{L(\tau-1)} & 0_{L(\tau-1)\times L(\theta-1)} \\ I_{L(\theta-1)\times L} & 0_{L(\theta-1)} \\ 0_{L\times L(\theta+\tau-2)} & H_\theta \end{bmatrix}$$
$$\boldsymbol{C} := \begin{bmatrix} I_{L(\tau-1)} & 0_{L(\tau-1)\times L(\theta-1)} & 0_{L(\tau-1)\times L} \\ 0_{L(\theta-1)\times L} & 0_{L(\theta-1)} & I_{L(\theta-1)\times L(\theta-1)} \\ H_{-\tau+1} & \cdots & H_{\theta-1} \end{bmatrix}$$

find permutation matrices ϱ_L, ϱ_R

$$\begin{bmatrix} \boldsymbol{B} & 0 & \boldsymbol{A} & \boldsymbol{C} & 0 & 0 \\ \boldsymbol{H} & 0 & \boldsymbol{G} & \boldsymbol{K} & \boldsymbol{F} & \boldsymbol{J} \end{bmatrix} = \varrho_L H (I_3 \otimes \varrho_R)$$

and such that with l row dimension of C and n the column dimension of C

$$rank(nullSpace(C^T)) = l - n$$
$$\begin{bmatrix} \boldsymbol{D} \boldsymbol{M} \end{bmatrix} = \begin{bmatrix} 0 \\ \varrho_L \Psi \end{bmatrix}$$

Obtaining Anderson-Moore outputs from Uhlig outputs

$$\boldsymbol{B} := \begin{bmatrix} P \\ R \end{bmatrix} \boldsymbol{\Phi\Psi} := \begin{bmatrix} Q \\ S \end{bmatrix}$$

Appendix C Computational Efficiency

Table 2: Matlab Flop Counts: Uhlig Examples

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
uhlig 0	$(2514, 4098)$	3.38385	23.9387	28.8007	88.7832	0.922832
	(4,1,1)	(8,4)	(8,4)	(8,3)	4	(3,1)
uhlig 1	$(6878, 9817)$	3.81346	46.9301	29.1669	15347.3	1.88979
	(6,1,1)	(12,6)	(12,6)	(12,5)	6	(5,1)
uhlig 2	$(6798, 9773)$	3.292	52.2499	35.7318	6468.78	1.80494
	(6,1,1)	(12,6)	(12,6)	(12,5)	6	(5,1)
uhlig 3	$(112555, 276912)$	3.62782	40.5874	22.6611	205753.	16.7633
	(14,1,1)	(28,14)	(28,14)	(28,13)	14	(10,4)
uhlig 4	$(6850, 9789)$	3.58292	45.675	29.5441	16250.5	1.89752
	(6,1,1)	(12,6)	(12,6)	(12,5)	6	(5,1)
uhlig 5	$(6850, 9789)$	3.58292	45.675	29.5441	16250.5	1.89752
	(6,1,1)	(12,6)	(12,6)	(12,5)	6	(5,1)
uhlig 6	$(7809, 26235)$	2.53182	47.5947	46.3196	190.905	3.42169
	(6,1,1)	(12,6)	(12,6)	(12,4)	6	(4,2)
uhlig 7	$(82320, 108000)$	2.8447	48.1792	28.9946	131.785	2.60423
	(13,1,1)	(26,13)	(26,13)	(26,11)	13	(11,2)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column normalized by dividing by comparable AIM value.

Table 3: Matlab Flop Counts: King & Watson, Klein, Binder & Peseran and Sims Examples

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
king 2	(2288, 2566)	2.36713	9.64904	20.7072	165.623	1.76311
	(3,1,1)	(6,3)	(6,3)	(6,1)	3	(2,1)
king 3	(1028, 2306)	2.81809	21.1284	77.1858	368.545	*NA*
	(3,1,1)	(6,3)	(6,3)	(6,-1)	3	(2,1)
king 4	(40967, 146710)	5.2211	39.8819	30.493	185.299	14.2569
	(9,1,1)	(18,9)	(18,9)	(18,5)	9	(3,6)
sims 0	(3501, 5576)	4.70923	59.9526	59.2871	*NA*	5.72694
	(5,1,1)	(10,5)	(10,5)	(10,3)	5	(4,1)
sims 1	(16662, 39249)	4.20292	56.8375	48.4135	*NA*	9.2849
	(8,1,1)	(16,8)	(16,8)	(16,5)	8	(6,2)
klein 1	(3610, 15147)	2.13213	10.2756	16.7828	35555.2	2.30526
	(3,1,1)	(6,3)	(6,3)	(6,2)	3	(1,2)
bp 1	(533, 1586)	3.1257	13.8987	59.7749	613.814	2.88743
	(2,1,1)	(4,2)	(4,2)	(4,0)	2	(1,1)
bp 2	(3510, 5659)	3.5265	77.2456	67.7846	800.98	21.4259
	(5,1,1)	(10,5)	(10,5)	(10,1)	5	(4,1)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column normalized by dividing by comparable AIM value.

Table 4: Matlab Flop Counts: General Examples

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
firmValue 1	$(535,1586)$	2.90467	11.5252	92.9738	534.686	*NA*
	(2,1,1)	(4,2)	(4,2)	(4,0)	2	(1,1)
athan 1	$(18475,70162)$	3.75578	135.329	113.463	1385.73	196.251
	(9,1,1)	(18,9)	(18,9)	(18,1)	9	(8,1)
fuhrer 1	$(352283,1885325)$	*NA*	125.141	148.459	*NA*	*NA*
	(12,1,4)	(60,12)	(60,48)	(60,1)	48	(9,39)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column normalized by dividing by comparable AIM value.

Table 5: Using Anderson-Moore to Improve Matrix Polynomial Methods

Model(m,n)	Uhlig	AM/Uhlig
uhlig 0(3,1)	2320	2053
uhlig 1(5,1)	12998	12731
uhlig 2(5,1)	12270	12003
uhlig 3(10,4)	1886798	304534
uhlig 4(5,1)	12998	12731
uhlig 5(5,1)	12998	12731
uhlig 6(4,2)	26720	24233
uhlig 7(11,2)	214380	213450

Note:

m, n See Appendix B.6

Appendix D Accuracy

Table 6: Matlab Relative Errors in B: Uhlig Examples

$$\frac{\parallel B_i - B_{exact} \parallel}{\parallel B_{exact} \parallel}$$

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
uhlig 0	$1 := 1.69446\,10^{-16}$ (4,1,1)	8. (8,4)	40. (8,4)	11. (8,3)	134848. 4	49. (3,1)
uhlig 1	$1 := 3.82375\,10^{-15}$ (6,1,1)	0.518098 (12,6)	2.63371 (12,6)	0.617504 (12,5)	10217.1 6	3.32793 (5,1)
uhlig 2	$1 := 4.88785\,10^{-15}$ (6,1,1)	1. (12,6)	2.00589 (12,6)	6.43964 (12,5)	9520.85 6	2.82951 (5,1)
uhlig 3	$1 := 1.15015\,10^{-15}$ (14,1,1)	2.65517 (28,14)	1.78046 (28,14)	1.45676 (28,13)	$8.57994\,10^{14}$ 14	20.9358 (10,4)
uhlig 4	$1 := 1.18357\,10^{-15}$ (6,1,1)	2.2561 (12,6)	1.00348 (12,6)	14.7909 (12,5)	10673.4 6	34.6115 (5,1)
uhlig 5	$1 := 1.18357\,10^{-15}$ (6,1,1)	2.2561 (12,6)	1.00348 (12,6)	14.7909 (12,5)	10673.4 6	34.6115 (5,1)
uhlig 6	$1 := 1.60458\,10^{-15}$ (6,1,1)	7.32281 (12,6)	38.3459 (12,6)	13.4887 (12,4)	$7.63524\,10^{13}$ 6	203.291 (4,2)
uhlig 7	$1 := 2.33332\,10^{-14}$ (13,1,1)	7.59657 (26,13)	4.15248 (26,13)	0.335751 (26,11)	*NA* 13	2.63499 (11,2)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column normalized by dividing by comparable AIM value.

Table 7: Matlab Relative Errors in B: King & Watson and Sims Examples

$$\frac{\| B_i - B_{exact} \|}{\| B_{exact} \|}$$

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
king 2	$1 := 0.$ (3,1,1)	$3.65169\,10^{-16}$ (6,3)	$1.61642\,10^{-16}$ (6,3)	0. (6,1)	0. 3	0 (2,1)
king 3	$1 := 0.$ (3,1,1)	0. (6,3)	$2.22045\,10^{-16}$ (6,3)	$2.49183\,10^{-15}$ (6,1)	0. 3	*NA* (2,1)
king 4	$1 := 2.25\,10^{-15}$ (9,1,1)	$3.99199\,10^{-15}$ (18,9)	$1.48492\,10^{-15}$ (18,9)	$2.0552\,10^{-15}$ (18,5)	$6.38338\,10^{-16}$ 9	$2.63969\,10^{-9}$ (3,6)
sims 0	$1 := 0.$ (5,1,1)	$8.84516\,10^{-17}$ (10,5)	$3.57788\,10^{-16}$ (10,5)	$2.34864\,10^{-16}$ (10,3)	*NA* 5	$5.55112\,10^{-17}$ (4,1)
sims 1	$1 := 1.18603\,10^{-15}$ (8,1,1)	$7.58081\,10^{-16}$ (16,8)	$9.08685\,10^{-16}$ (16,8)	$1.28298\,10^{-15}$ (16,6)	*NA* 8	$8.11729\,10^{-16}$ (6,2)
klein 1	$1 := 1.29398\,10^{-14}$ (3,1,1)	$3.2259\,10^{-14}$ (6,3)	$8.72119\,10^{-14}$ (6,3)	$1.76957\,10^{-13}$ (6,1)	$4.54742\,10^{-10}$ 3	$4.61522\,10^{-13}$ (1,2)
bp 1	$1 := 1.54607\,10^{-15}$ (2,1,1)	$2.82325\,10^{-15}$ (4,2)	$5.10874\,10^{-15}$ (4,2)	$1.23685\,10^{-14}$ (4,0)	$5.95443\,10^{-10}$ 2	$1.06208\,10^{-14}$ (1,1)
bp 2	$1 := 5.00765\,10^{-16}$ (5,1,1)	$5.15101\,10^{-15}$ (10,5)	$5.52053\,10^{-15}$ (10,5)	$5.21764\,10^{-15}$ (10,1)	$7.14801\,10^{-16}$ 5	$3.98148\,10^{-14}$ (4,1)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column not normalized by dividing by comparable AIM value.

Table 8: Matlab Relative Errors in ϑ: Uhlig Examples

$$\frac{\| \vartheta_i - \vartheta_{exact} \|}{\| \vartheta_{exact} \|}$$

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
uhlig 0	$1 := 9.66331\,10^{-16}$ (4,1,1)	0.532995 (8,4)	*NA* (8,4)	1.34518 (8,3)	159728. 4	9.54822 (3,1)
uhlig 1	$1 := 2.25938\,10^{-15}$ (6,1,1)	3.78248 (12,6)	*NA* (12,6)	2.32241 (12,5)	$5.30459\,10^{6}$ 6	2.33909 (5,1)
uhlig 2	$1 := 5.05614\,10^{-15}$ (6,1,1)	0.43572 (12,6)	*NA* (12,6)	2.62987 (12,5)	$1.51637\,10^{6}$ 6	1. (5,1)
uhlig 3	$1 := 8.93\,10^{-16}$ (14,1,1)	2.08127 (28,14)	*NA* (28,14)	1.39137 (28,13)	*NA* 14	14.3934 (10,4)
uhlig 4	$1 := 0.0114423$ (6,1,1)	1. (12,6)	*NA* (12,6)	1. (12,5)	0.999999 6	1. (5,1)
uhlig 5	$1 := 1.37388\,10^{-15}$ (6,1,1)	3.79327 (12,6)	*NA* (12,6)	9.97923 (12,5)	$8.03937\,10^{6}$ 6	12.2245 (5,1)
uhlig 6	$1 := 6.97247\,10^{-14}$ (6,1,1)	1.00929 (12,6)	*NA* (12,6)	1.83538 (12,4)	$1.12899\,10^{13}$ 6	27.5959 (4,2)
uhlig 7	$1 := 7.47708\,10^{-14}$ (13,1,1)	1.67717 (26,13)	*NA* (26,13)	0.131889 (26,11)	*NA* 13	0.665332 (11,2)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column normalized by dividing by comparable AIM value.

Table 9: Matlab Relative Errors in ϑ: King & Watson and Sims Examples

$$\frac{\| \vartheta_i - \vartheta_{exact} \|}{\| \vartheta_{exact} \|}$$

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
king 2	$1 := 4.996\,10^{-16}$ (3,1,1)	$4.996\,10^{-16}$ (6,3)	*NA* (6,3)	$4.996\,10^{-16}$ (6,1)	$4.996\,10^{-16}$ 3	$4.996\,10^{-16}$ (2,1)
king 3	$1 := 2.58297\,10^{-15}$ (3,1,1)	$5.02999\,10^{-15}$ (6,3)	*NA* (6,3)	$6.42343\,10^{-15}$ (6,1)	$2.58297\,10^{-15}$ 3	*NA* (2,1)
king 4	$1 := 3.45688\,10^{-16}$ (9,1,1)	$1.12445\,10^{-15}$ (18,9)	*NA* (18,9)	$8.40112\,10^{-16}$ (18,5)	$2.65811\,10^{-16}$ 9	$2.47355\,10^{-9}$ (3,6)
sims 0	$1 := 7.66638\,10^{-16}$ (5,1,1)	$7.85337\,10^{-16}$ (10,5)	*NA* (10,5)	$9.53623\,10^{-16}$ (10,3)	*NA* 5	$7.66638\,10^{-16}$ (4,1)
sims 1	$1 := 9.73294\,10^{-16}$ (8,1,1)	$1.57671\,10^{-15}$ (16,8)	*NA* (16,8)	$4.23589\,10^{-15}$ (16,6)	*NA* 8	$9.73294\,10^{-16}$ (6,2)
klein 1	$1 := 2.33779\,10^{-15}$ (3,1,1)	$1.9107\,10^{-14}$ (6,3)	*NA* (6,3)	$1.07269\,10^{-13}$ (6,1)	$2.3967\,10^{-9}$ 3	$2.66823\,10^{-13}$ (1,2)
bp 1	$1 := 1.4802\,10^{-15}$ (2,1,1)	$8.91039\,10^{-15}$ (4,2)	*NA* (4,2)	$1.20169\,10^{-14}$ (4,0)	$5.55737\,10^{-9}$ 2	$1.16858\,10^{-14}$ (1,1)
bp 2	$1 := 6.1809\,10^{-16}$ (5,1,1)	$1.25347\,10^{-15}$ (10,5)	*NA* (10,5)	$1.3268\,10^{-15}$ (10,1)	$6.1809\,10^{-16}$ 5	$8.9026\,10^{-15}$ (4,1)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column not normalized by dividing by comparable AIM value.

Table 10: Matlab Relative Errors in F: Uhlig Examples

$$\frac{\parallel F_i - F_{exact} \parallel}{\parallel F_{exact} \parallel}$$

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
uhlig 0	$1 := 4.60411\,10^{-16}$	*NA*	38.383	*NA*	6280.03	*NA*
	(4,1,1)	(8,4)	(8,4)	(8,3)	4	(3,1)
uhlig 1	$1 := 6.12622\,10^{-16}$	*NA*	4.25457	*NA*	3126.17	*NA*
	(6,1,1)	(12,6)	(12,6)	(12,5)	6	(5,1)
uhlig 2	$1 := 6.13246\,10^{-16}$	*NA*	3.58093	*NA*	3918.23	*NA*
	(6,1,1)	(12,6)	(12,6)	(12,5)	6	(5,1)
uhlig 3	$1 := 7.28843\,10^{-16}$	*NA*	12.9392	*NA*	$1.40986\,10^{15}$	*NA*
	(14,1,1)	(28,14)	(28,14)	(28,13)	14	(10,4)
uhlig 4	$1 := 6.03573\,10^{-16}$	*NA*	2.73637	*NA*	1028.17	*NA*
	(6,1,1)	(12,6)	(12,6)	(12,5)	6	(5,1)
uhlig 5	$1 := 6.03573\,10^{-16}$	*NA*	2.73637	*NA*	1028.17	*NA*
	(6,1,1)	(12,6)	(12,6)	(12,5)	6	(5,1)
uhlig 6	$1 := 6.0499\,10^{-16}$	*NA*	308.153	*NA*	$1.65292\,10^{13}$	*NA*
	(6,1,1)	(12,6)	(12,6)	(12,4)	6	(4,2)
uhlig 7	$1 := 7.52423\,10^{-14}$	*NA*	1.72491	*NA*	*NA*	*NA*
	(13,1,1)	(26,13)	(26,13)	(26,11)	13	(11,2)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column normalized
by dividing by comparable AIM value.

Table 11: Matlab Relative Errors in F: King & Watson and Sims Examples

$$\frac{\parallel F_i - F_{exact} \parallel}{\parallel F_{exact} \parallel}$$

Model	AIM (L,τ,θ)	KW (m,p)	Sims (L,M_2)	Klein (L,n_k)	BP L	Uhlig (m,n)
Computes	(B,ϑ,ϕ,F)	(B,ϑ)	(B,ϕ,F)	(B)	(B,ϑ,ϕ,F)	(B,ϑ)
king 2	$1 := 7.49401\,10^{-16}$ (3,1,1)	*NA* (6,3)	$1.36349\,10^{-15}$ (6,3)	*NA* (6,1)	$7.49401\,10^{-16}$ 3	*NA* (2,1)
king 3	$1 := 3.9968\,10^{-16}$ (3,1,1)	*NA* (6,3)	$6.53472\,10^{-15}$ (6,3)	*NA* (6,1)	$3.9968\,10^{-16}$ 3	*NA* (2,1)
king 4	$1 := 2.13883\,10^{-15}$ (9,1,1)	*NA* (18,9)	$2.58686\,10^{-15}$ (18,9)	*NA* (18,5)	$7.29456\,10^{-16}$ 9	*NA* (3,6)
sims 0	$1 := 7.13715\,10^{-16}$ (5,1,1)	*NA* (10,5)	$1.1917\,10^{-15}$ (10,5)	*NA* (10,3)	*NA* 5	*NA* (4,1)
sims 1	$1 := 2.41651\,10^{-15}$ (8,1,1)	*NA* (16,8)	$2.92152\,10^{-15}$ (16,8)	*NA* (16,6)	*NA* 8	*NA* (6,2)
klein 1	$1 := 1.24387\,10^{-15}$ (3,1,1)	*NA* (6,3)	$1.66911\,10^{-13}$ (6,3)	*NA* (6,1)	$3.70873\,10^{-10}$ 3	*NA* (1,2)
bp 1	$1 := 4.43937\,10^{-16}$ (2,1,1)	*NA* (4,2)	$7.51277\,10^{-15}$ (4,2)	*NA* (4,0)	$5.03025\,10^{-11}$ 2	*NA* (1,1)
bp 2	$1 := 5.82645\,10^{-16}$ (5,1,1)	*NA* (10,5)	$1.71285\,10^{-15}$ (10,5)	*NA* (10,1)	$5.82645\,10^{-16}$ 5	*NA* (4,1)

Note:

$L,\tau,\theta,B,\vartheta,\phi,F$ See Appendix B.1

m,p See Appendix B.2

L,M_2 See Appendix B.3

L,n_k See Appendix B.4

L See Appendix B.5

m,n See Appendix B.6

KW, Sims, Klein, BP, Uhlig column not normalized by dividing by comparable AIM value.