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

Reliably Computing Nonlinear Dynamic Stochastic Model Solutions: An Algorithm with Error Formulas

Gary S. Anderson

2018-070

Please cite this paper as:
Anderson, Gary S. (2018). "Reliably Computing Nonlinear Dynamic Stochastic Model Solutions: An Algorithm with Error Formulas," Finance and Economics Discussion Series 2018-070. Washington: Board of Governors of the Federal Reserve System, https://doi.org/10.17016/FEDS.2018.070.

Reliably Computing Nonlinear Dynamic Stochastic Model Solutions: An Algorithm with Error Formulas

Gary S. Anderson*

Friday 28th September, 2018: 11:18am

Abstract

This paper provides a new technique for representing discrete time nonlinear dynamic stochastic time invariant maps. Using this new series representation, the paper shows how to augment a solution strategy based on projection methods with an additional set of constraints thereby enhancing its reliability. The paper also provides general formulas for evaluating the accuracy of proposed solutions. The technique can readily accommodate models with occasionally binding constraints and regime switching. The algorithm uses Smolyak polynomial function approximation in a way which makes it possible to exploit a high degree of parallelism.

*The analysis and conclusions set forth are those of the author and do not indicate concurrence by other members of the research staff or the Board of Governors. I would like to thank Luca Guerrieri, Christopher Gust, Hess Chung, Benjamin Johannsen, Robert Tetlow and Etienne Gagnon for their comments and suggestions. I would also like to thank Stephanie Harrington for her help with document preparation.

Contents

1 Introduction

This paper provides a new technique for the solution of discrete time dynamic stochastic infinite-horizon economies with bounded-time-invariant solutions. It describes how to obtain decision rules satisfying a system of first-order conditions, "Euler Equations", and how to assess their accuracy.[1] For an overview of techniques for solving nonlinear models, see (Judd, 1992; Christiano and Fisher, 2000; Doraszelskiy and Judd, 2004; Gaspar and Judd, 1997; Judd et al., 2014; Marcet and Lorenzoni, 1999; Judd et al., 2011; Maliar and Maliar, 2001; Binning and Maih, 2017; Judd et al., 2017b).

This new series representation for bounded time series, adapted from (Anderson, 2010),

$$x(x_{t-1}, \epsilon_t) = Bx_{t-1} + \phi\psi_\epsilon\epsilon + (I - F)^{-1}\phi\psi_c + \sum_{\nu=0}^{\infty} F^\nu \phi \mathbf{Z}_{t+\nu}(x_{t-1}, \epsilon_t)$$

makes it possible to construct a series representation for any bounded time invariant discrete time map. As yet, the author has found no comparable use of a linear reference dynamical system for conveniently transforming one bounded infinite dimensional series into another.

It turns out that this representation provides a way to use Euler equation errors to approximate the error in components of decision rules.[2] This paper provides approximation formulas explicitly relating the Euler equation errors to the components of errors in the decision rule.

In addition, the series representation provides exploitable constraints on model solutions that enhance an algorithm's reliability. Practitioners using projection methods and other techniques have often found models difficult to solve. They find that it can be difficult to find an initial guess for a solution so that the calculations converge and that sometimes the convergent decision rules lack appropriate long run properties like stability. Furthermore, economists have an increasing interest in crafting models with occasionally binding constraints and regime switching where, unfortunately, these problems seem to arise more often. The series representation overcomes these problems by making it easier to choose an initial guess that converges and by ensuring that the solutions have appropriate long run properties.

The numerical implementation of the algorithm builds upon the work of (Judd et al., 2011, 2017b). In particular it uses the an-isotropic Smolyak Method, the adaptive parallelotope method (Judd et al., 2014) and precomputes all integrals required by the calculations (Judd et al., 2017b). The algorithm should scale well to large models as many of the algorithm's components can be computed in parallel with limited amounts of information flowing between compute nodes.

The paper is organized as follows. Section 2 presents a useful new series representation for any bounded time series. Section 3 shows how to use this series representation to represent characterize time invariant maps. Section 4 provides formulas for approximating dynamic stochastic model errors in proposed solutions. Section 5 presents a new solution algorithm for

[1] The series representation presented here should also prove useful for applications based on value function iteration but I defer treating this topic for future work.

[2] Others have also studied approximation error for dynamic stochastic models (Judd et al., 2017a; Santos and Peralta-alva, 2005; Santos, 2000).

improving proposed solutions. Section 6 applies the technique to a model with occasionally binding constraints. Section 7 concludes.

2 A New Series Representation For Bounded Time Series

Since (Blanchard and Kahn, 1980) numerous authors have developed solution algorithms and formulas for solving linear rational expectations models. It turns out that the solutions associated with these linear saddle point models, by quantifying the potential impact of anticipated future shocks, can play a new and somewhat surprising role in characterizing the solutions for highly nonlinear models. In preparation for discussing how these calculations can be useful for representing time invariant maps, this section describes the relationship between *linear reference models* and bounded time series.

2.1 Linear Reference Models and a Formula for "Anticipated Shocks"

For any linear homogeneous L dimensional deterministic system that produces a unique stable solution,

$$H_{-1}x_{t-1} + H_0 x_t + H_1 x_{t+1} = 0,$$

inhomogeneous solutions

$$H_{-1}x_{t-1} + H_0 x_t + H_1 x_{t+1} = \psi_\epsilon \epsilon + \psi_c$$

can be computed as

$$x_t = Bx_{t-1} + \phi\psi_\epsilon \epsilon + (I - F)^{-1}\phi\psi_c$$

where

$$\phi = (H_0 + H_1 B)^{-1} \text{ and } \quad F = -\phi H_1.$$

(Anderson, 2010)

It will be useful to collect the components of this linear reference model solution,$\mathcal{L} \equiv \{H, \psi_\epsilon, \psi_c; B, \phi, F\}$, for later use.[3]

Theorem 2.1 *Consider any bounded discrete time path*

$$\mathbf{x}_t \in R^L \ \ with \ \ \|\mathbf{x}_t\|_\infty \leq \overline{\mathcal{X}} \quad \forall t > 0. \tag{1}$$

Now, given the trajectories 1*, define* $\mathbf{z}_t$ *as*

$$\mathbf{z}_t \equiv H_{-1}\mathbf{x}_{t-1} + H_0 \mathbf{x}_t + H_1 \mathbf{x}_{t+1} - \psi_c - \psi_\epsilon \epsilon_t$$

[3]In Section 2.2, I will use these matrices to construct a series representation for any bounded path and in 5 compute improvements in proposed model solutions.

One can then express the $\mathbf{x}_t$ *solving*

$$H_{-1}x_{t-1} + H_0 x_t + H_1 x_{t+1} = \psi_\epsilon \epsilon + \psi_c + z_t$$

using

$$\mathbf{x}_t = Bx_{t-1} + \phi\psi_\epsilon\epsilon + (I-F)^{-1}\phi\psi_c + \sum_{\nu=0}^{\infty} F^s \phi \mathbf{z}_{t+\nu} \tag{2}$$

and

$$\mathbf{x}_{t+k+1} = B\mathbf{x}_{t+k} + (I-F)^{-1}\phi\psi_c + \sum_{\nu=0}^{\infty} F^\nu \phi \mathbf{z}_{t+k+\nu+1} \quad \forall t, k \geq 0.$$

Proof See (Anderson, 2010),

Consequently, given a bounded time series 1, and a stable linear homogeneous system, $\mathcal{L}$, one can easily compute a corresponding series representation like 2 for x_t. Interestingly, the linear model, H, the constant term ψ_c and the impact of the stochastic shocks ψ_ϵ can be chosen rather arbitrarily – the only constraints being the existence of a saddle-point solution for the linear system and that all z-values fall in the row space of H_1. Note that since the eigenvalues of F are all less than one in magnitude, the formula supports the intuition that distant future shocks influence current conditions less than imminent shocks.

The transformation $\{x_t, x_t, x_{t+1}, x_{t+2}\ldots\} \to \mathcal{L}(x_{t-1}, \epsilon_t) \to \{z_t, z_{t+1}, z_{t+2}, \ldots\}$ is invertible. $\{z_t, z_{t+1}, z_{t+2}, \ldots\} \to \mathcal{L}(x_{t-1}, \epsilon_t)^{-1} \to \{x_t, x_t, x_{t+1}, x_{t+2}\ldots\}$ and the inverse can be interpreted as giving the impact of "fully anticipated future shocks" on the path of x_t in a linear perfect foresight model. Note that the reference model is deterministic, so that the $z_t(x_{t-1}, \epsilon_t)$ functions will fully account for any stochastic aspects encountered in the models I analyze.

A key feature to note is that the series representation can accommodate arbitrarily complex time series trajectories, so long as these trajectories are bounded. Later, this observation will give us some confidence in the robustness of the algorithm for constructing series representations for unknown families of functions satisfying complicated systems of dynamic non-linear equations.

2.2 The Linear Reference Model in Action: Arbitrary Bounded Time Series

Consider the following matrix constructed from "almost" arbitrary coefficients

$$\begin{bmatrix} H_{-1} & H_0 & H_1 \end{bmatrix} = \begin{pmatrix} 0.1 & 0.5 & -0.5 & 1. & 0.4 & 0.9 & 1. & 1. & 0.9 \\ 0.2 & 0.2 & -0.5 & 7. & 0.4 & 0.8 & 3. & 2. & 0.6 \\ 0.1 & -0.25 & -1.5 & 2.1 & 0.47 & 1.9 & 2.1 & 2.1 & 3.9 \end{pmatrix} \tag{3}$$

with $\psi_c = \psi_\epsilon = 0$, $\psi_z = I$. I have chosen these coefficients so that the linear model has a unique stable solution and, since H_1 is full rank, its column space will span the column space for all non zero values for $z_t \in \mathcal{R}^3$.[4] It turns out that the algorithm is very forgiving about the choice of $\mathcal{L}$.

[4]The flexibility in choosing $\mathcal{L}$ will become more important in later in the paper where we must choose a linear reference model in a context where we will have little guidance about what might make a good choice.

The solution for this system is

$$B = \begin{pmatrix} -0.0282384 & -0.0552487 & 0.00939369 \\ -0.0664679 & -0.700462 & -0.0718527 \\ -0.163638 & -1.39868 & 0.331726 \end{pmatrix}$$
$$\phi = \begin{pmatrix} 0.0210079 & 0.15727 & -0.0531634 \\ 1.20712 & -0.0553003 & -0.431842 \\ 2.58165 & -0.183521 & -0.578227 \end{pmatrix}$$
$$F = \begin{pmatrix} -0.381174 & -0.223904 & 0.0940684 \\ -0.134352 & -0.189653 & 0.630956 \\ -0.816814 & -1.00033 & 0.0417094 \end{pmatrix}$$

It is straightforward to apply 2 to the following three bounded families of time series paths

$$x_{1,t} = \frac{\|x_{t-1}\|}{1 + \|\epsilon_t\|} D_\pi(t)$$

where $D_\pi(t)$ gives the t-th digit of π

$$x_{2,t} = \frac{\|x_{t-1}\|}{(1 + \|\epsilon_t\|)^2} (-1)^t$$
$$a x_{3,t} = \epsilon_t$$

and the ϵ_t are a sequence of pseudo random draws from the uniform distributions $\mathcal{U}(-4, 4)$ produced subsequent to the selection of a random seed.

$$randomseed(\|x_{t-1}\| + \|\epsilon_t\|)$$

The three panels in Figure 1 display these three time series generated for a particular initial state vector and shock value. The first set of trajectories characterizes a function of the digits in the decimal representation of π. The second set of trajectories oscillates between two values determined by a nonlinear function of the initial conditions, x_{t-1} and the shock. The third set of trajectories characterizes a sequence of uniformly distributed random numbers based on a seed determined by a nonlinear function of the initial conditions and the shock. These paths were chosen to emphasize that continuity is not important for the existence of the series representation, that the trajectories need not converge to a fixed point, and need not be produced by some linear rational expectations solution or by the iteration of a discrete-time map. The spanning condition and the boundedness of the paths are sufficient conditions for the existence of the series representation based on the linear reference model,$\mathcal{L}$.[5]

Though unintuitive and perhaps lacking a compelling interpretation, the values for z_t exist, provide a series for the x_t and are easy to compute. One can apply the calculations for any given initial condition to produce a z series exactly replicating any bounded set of trajectories. Consequently, these numerical linear algebra calculations can easily transform a z_t series into an x_t series and vice versa. Since this can be done for any path starting from any particular initial conditions, any discrete time invariant map that generates families of bounded paths has a series representation.

[5]Although potentially useful in some contexts, this paper will not investigate representations for families of unbounded, but slowly diverging trajectories.

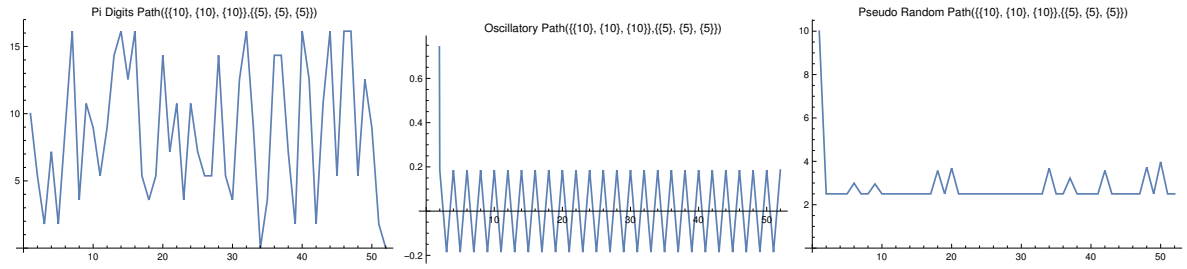

Figure 1: Arbitrary Bounded Time Series Paths

2.3 Assessing x_t Errors

The formula 2 computes the impact on the current state of fully anticipated future shocks. It characterizes the impact exactly. However one can contemplate the impact of at least two deviations from the exact calculation: one could truncate a series of correct values of z_t or one might have imprecise values of z_t along the path.

2.3.1 Truncation Error

The series representation can compute the entire series exactly if all the terms are included, but, it will at times be useful to exploit the fact that the terms for state vectors closer to the initial time have the most important impact. One could consider approximating $\mathcal{X}_t$ by truncating the series at a finite number of terms.

$$\hat{\mathbf{x}}_t \equiv Bx_{t-1} + \phi\psi_\epsilon\epsilon + (I-F)^{-1}\phi\psi_c + \sum_{s=0}^{k} F^s\phi z_t$$

We can bound the series approximation truncation errors:

$$\sum_{s=k+1}^{\infty} F^s\phi\psi_z = (I-F)^{-1}F^{k+1}\phi\psi_z$$

$$\|\mathbf{x}(x_{t-1},\epsilon) - \hat{\mathbf{x}}(x_{t-1},\epsilon,k)\|_\infty \leq \left\|(I-F)^{-1}F^{k+1}\phi\psi_z\right\|_\infty \left(\|H_{-1}\|_\infty + \|H_0\|_\infty + \|H_1\|_\infty\right)\left\|\overline{\mathcal{X}}\right\|_\infty$$

Figure 2 shows that this truncation error can be a very conservative measure of the accuracy of the truncated series. The orange line represents the computed approximation of the infinity norm of the difference between x_t from the full series and a truncated series for different lengths of the truncation. The blue line shows the infinity norm of the actual difference between the x_t computed using the full series and the value obtained using a truncated series. The series requires only the first 20 terms to compute the initial value of the state vector to high precision.

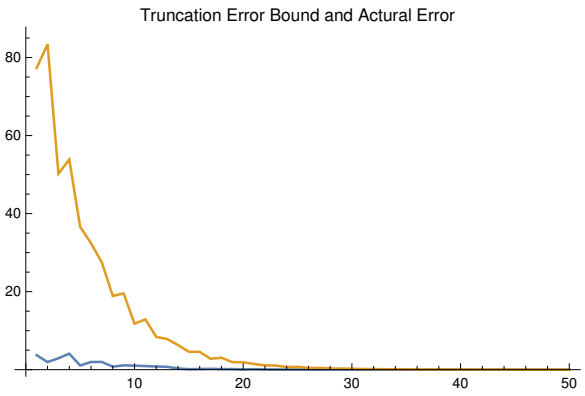

Figure 2: x_t Error Approximation Versus Actual Error

2.3.2 Path Error

We can assess the impact of perturbed values for z_t by computing the maximum discrepancy,Δz_t, impacting the z_t and applying the partial sum formula. One can approximate $\mathcal{X}_t$ using

$$\hat{\mathbf{x}}_t \equiv Bx_{t-1} + \phi\psi_\epsilon\epsilon + (I - F)^{-1}\phi\psi_c + \sum_{s=0}^{\infty} F^s\phi(z_t + \Delta z_t)$$

Note, yet again, that for approximating $\mathbf{x}(x_{t-1}, \epsilon)$, the impact of a given realization along the path declines for those realizations which are more temporally distant. However, we can conservatively bound the series approximation errors by using the largest possible Δz_t in the formula:

$$\|\mathbf{x}(x_{t-1}, \epsilon) - \hat{\mathbf{x}}(x_{t-1}, \epsilon, k)\|_\infty \leq \left\|(I - F)^{-1}\phi\psi_z\right\|_\infty \|\Delta z_t\|_\infty$$

3 Dynamic Stochastic Time Invariant Maps

Many dynamic stochastic models have solutions that fall in the class of bounded time invariant maps. Consequently, if we take care (see Section 3.2), we can apply the law of iterated expectations, to compute conditional expected solution paths forward from any initial value and realization of (x_{t-1}, ϵ_t).[6] As a result, we can use the family of conditional expectations paths along with a contrived linear reference model to generate a series representation for the model solutions and to approximate errors.

So long as the trajectories are bounded, the time invariant maps can be represented using the framework from section 2. The series representation will provide a linearly weighted sum of $z_t(x_{t-1}, \epsilon_t)$ functions that give us an approximation for the model solutions. This section presents a familiar RBC model as an example.[7]

3.1 An RBC Example

Consider the model described in (Maliar and Maliar, 2005)[8]

$$\max \left\{ u(c_t) + E_t \sum_{t=1}^{\infty} \delta^t u(c_{t+1}) \right\}$$
$$c_t + k_t = (1-d)k_{t-1} + \theta_t f(k_t)$$
$$f(k_t) = k_t^{\alpha}$$
$$u(c) = \frac{c^{1-\eta} - 1}{1-\eta}$$

The first order conditions for the model are

$$\frac{1}{c_t^{\eta}} = \alpha\delta k_t^{\alpha-1} E_t \left(\frac{\theta_{t+1}}{c_{t+1}^{\eta}} \right)$$
$$c_t + k_t = \theta_{t-1} k_{t-1}^{\alpha}$$
$$\theta_t = \theta_{t-1}^{\rho} e^{\epsilon_t}$$

It is well known that when $\eta = d = 1$, we have a closed form solution (Lettau, 2003):

$$x_t(x_{t-1}, \epsilon_t) \equiv \mathcal{D}(x_{t-1}, \epsilon_t) \equiv \begin{bmatrix} c_t(x_{t-1}, \epsilon_t) \\ k_t(x_{t-1}, \epsilon_t) \\ \theta_t(x_{t-1}, \epsilon_t) \end{bmatrix} = \begin{bmatrix} (1-\alpha\delta)\theta_t k_{t-1}^{\alpha} \\ \alpha\delta\theta_t k_{t-1}^{\alpha}. \\ \theta_{t-1}^{\rho} e^{\epsilon_t}. \end{bmatrix}$$

[6] Economists have long used similar manipulation of time series paths for dynamic models. Indeed, Potter constructs his generalized impulse response functions using differences in conditional expectations paths (Potter, 2000; Koop et al., 1996).

[7] Later, in order to handle models with regime switching and occasionally binding constraints, I will need to consider more complicated collections of equation systems with Boolean gates. I will show how to apply the series formulation and to approximate the errors for these models as well.

[8] Here, I set their $\beta = 1$ and do not discuss quasi-geometric discounting or time-inconsistency.

For mean zero iid ϵ_t we can easily compute the conditional expectation of the model variables for any given θ_{t+k}, k_{t+k}

$$\mathbb{D}(x_{t+k+1}) \equiv \begin{bmatrix} E_t(c_{t+k+1}|\theta_{t+k}, k_{t+k}) \\ E_t(k_{t+k+1}|\theta_{t+k}, k_{t+k}) \\ E_t(\theta_{t+k+1}|\theta_{t+k}, k_{t+k}) \end{bmatrix} = \begin{bmatrix} (1-\alpha\delta)k_{t+k}^{\alpha} e^{\frac{\sigma^2}{2}} \theta_{t+k}^{\rho} \\ \alpha\delta k_{t+k}^{\alpha} e^{\frac{\sigma^2}{2}} \theta_{t+k}^{\rho} \\ e^{\frac{\sigma^2}{2}} \theta_{t+k}^{\rho} \end{bmatrix}$$

For any given values of $k_{t-1}, \theta_{t-1}, \epsilon_t$, the model decision rule, $(\mathcal{D})$, and decision rule conditional expectation, $(\mathbb{D})$, can generate a conditional expectations path forward from any given initial (x_{t-1}, ϵ_t):

$$\mathbf{x}_t(x_{t-1}, \epsilon_t) = \mathcal{D}(x_{t-1}, \epsilon_t)$$
$$\mathbf{x}_{t+k+1}(x_{t-1}, \epsilon_t) = \mathbb{D}(\mathbf{x}_{t+k}(x_{t-1}, \epsilon_t))$$

and corresponding paths for $\{(z_{1t}(x_{t-1}, \epsilon_t), z_{2t}(x_{t-1}, \epsilon_t), z_{3t}(x_{t-1}, \epsilon_t)), \ldots\}$

$$z_{t+k} \equiv H_{-1}\mathbf{x}_{t+k-1} + H_0\mathbf{x}_{t+k} + H_1\mathbf{x}_{t+K+1}$$

Formula 2 requires that

$$\mathbf{x}(x_{t-1}, \epsilon_t) = B \begin{bmatrix} c_{t-1} \\ k_{t-1} \\ \theta_{t-1} \end{bmatrix} + \phi\psi_\epsilon\epsilon_t + (I-F)^{-1}\phi\psi_c + \sum_{\nu=0}^{\infty} F^s \phi z_{t+\nu}(k_{t-1}, \theta_{t-1}, \epsilon_t)$$

For example, using $d = 1$ and the following parameter values and using the arbitrary linear reference model (3) we can generate a series representation for the model solutions.

$$\begin{array}{l} \texttt{alpha} \to \texttt{0.36} \\ \texttt{delta} \to \texttt{0.95} \\ \texttt{rho} \to \texttt{0.95} \\ \texttt{sigma} \to \texttt{0.01} \end{array} \quad \text{we have} \quad \begin{bmatrix} c_{ss} \\ k_{ss} \\ \theta_{ss} \end{bmatrix} = \begin{bmatrix} \texttt{0.360408} \\ \texttt{0.187324} \\ \texttt{1.001} \end{bmatrix}$$

$$\begin{bmatrix} k_{t-1} \\ \theta_{t-1} \\ \epsilon_t \end{bmatrix} = \begin{bmatrix} \texttt{0.18} \\ \texttt{1.1} \\ \texttt{0.01} \end{bmatrix} \tag{4}$$

Figure 3 shows, from top to bottom, the paths of $\theta_t, c_t,$ and k_t from the initial values given in Equation 4.

By including enough terms we can compute the solution for x_t exactly. Figure 4 shows the impact that truncating the series has on approximation of the time t values of the state variables. The approximated magnitude for the error, B_n, shown in red is again very pessimistic compared to the actual error, $Z_n = \|x_t - x_t^\star\|_\infty$, shown in blue. With enough terms, even using an "almost" arbitrarily chosen linear model, the series approximation provides a machine precision accurate value for the time t state vector.

Figure 5 shows that using a linearization that better tracks the nonlinear model paths, improves the approximation Z_n, shown in blue and the approximate error B_n shown in red. Using the linearization of the RBC model around the ergodic mean produces a tighter but still pessimistic bound on the errors for the initial state vector. Again, the first few terms make most of the difference in approximating the value of the state variables.

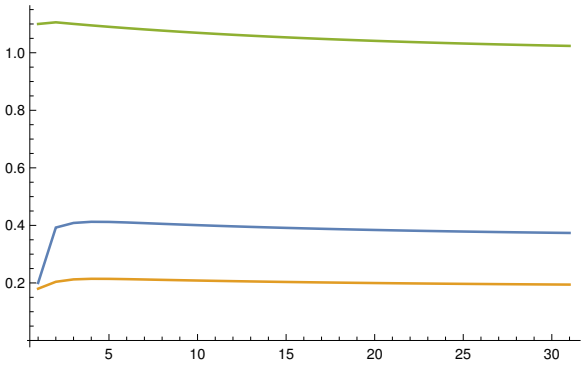

Figure 3: model variable values

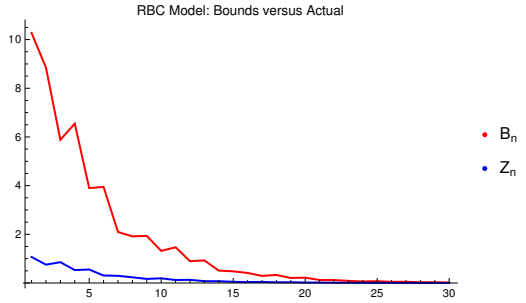

Figure 4: RBC Known Solution: Truncation Approximate Error Versus Actual: "Almost Arbitrary" Linear Reference Model

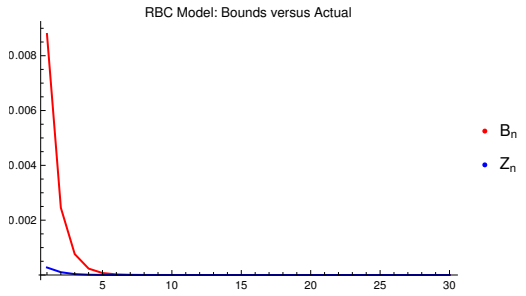

Figure 5: RBC Known Solution: Truncation Approximate Error Versus Actual: Stochastic Steady State Linearization Linear Reference Model

3.2 A Model Specification Constraint

For convenience of notation in what follows, I will focus on models built up from components of the form

$$h_i(x_{t-1}, x_t, x_{t+1}, \epsilon_t) = h_{io}^{det}(x_{t-1}, x_t, \epsilon_t) + \sum_{j=1}^{p_i}[h_{ij}^{det}(x_{t-1}, x_t, \epsilon_t) h_{ij}^{nondet}(x_{t+1})] = 0$$

This is a very broad class of models including most widely used macroeconomics models. This constraint will make it easier to write down and manipulate the conditional expectations expressions in the description of the algorithms in subsequent sections. For example, the Euler equations for the neoclassical growth model can be written as

$$h_{10^{det}}(\cdot) = \frac{1}{c_t^\eta},\ h_{11}^{det}() = \alpha\delta k_t^{\alpha-1},\ h_{11}^{nondet}(\cdot) = E_t\left(\frac{\theta_{t+1}}{c_{t+1}^\eta}\right)$$
$$h_{20}^{det}(\cdot) = c_t + k_t - \theta_t k_{t-1}^\alpha,\ h_{21}^{det}(\cdot) = 0$$
$$h_{30}^{det}(\cdot) = \ln\theta_t - (\rho\ln\theta_{t-1} + \epsilon_t),\ h_{31}^{det}(\cdot) = 0$$

Since we need to compute the conditional expectation of nonlinear expressions, this setup will make it possible for us to use auxiliary variables to correctly compute the required expected values. We will be working with models where expectations are computed at time t, with ϵ_t known. We can construct a linear reference model for the modified RBC system by augmenting the RBC model with the equation

$$\mathbf{N}_t = \frac{\theta_t}{c_t},$$

substituting $\mathbf{N}_{t+1}$ for $\frac{\theta_{t+1}}{c_{t+1}}$ in the first equation and linearizing the RBC model about the ergodic mean.

This leads to:

$$\begin{bmatrix} H_{-1} & H_0 & H_1 \end{bmatrix} =$$
$$\begin{pmatrix} 0. & 0. & 0. & 0. & -7.6986 & 9.47963 & 0. & 0. & 0. & 0. & -0.999 & 0. \\ 0. & -1.05263 & 0. & 0. & 1. & 1. & 0. & -0.547185 & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 7.7063 & 0. & 1. & -2.77463 & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & -0.949953 & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. \end{pmatrix}$$

with

$$\psi_\epsilon = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \psi_z = I$$

These coefficients produce a unique stable linear solution.

$$B = \begin{pmatrix} 0. & 0.692632 & 0. & 0.342028 \\ 0. & 0.36 & 0. & 0.177771 \\ 0. & -5.33763 & 0. & 0. \\ 0. & 0. & 0. & 0.949953 \end{pmatrix}, \phi = \begin{pmatrix} -0.0444237 & 0.658 & 0. & 0.360048 \\ 0.0444237 & 0.342 & 0. & 0.187137 \\ 0.342342 & -5.07075 & 1. & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

$$F = \begin{pmatrix} 0. & 0. & -0.0443793 & 0. \\ 0. & 0. & 0.0443793 & 0. \\ 0. & 0. & 0.342 & 0. \\ 0. & 0. & 0. & 0. \end{pmatrix}$$

$$\psi_c = \begin{pmatrix} -3.7735 \\ -0.197184 \\ 2.77741 \\ 0.0500976 \end{pmatrix}$$

Recomputing the truncation errors with the expanded model produces nearly identical results for variable approximation errors.

4 Approximating Model Solution Errors

This section shows how to use the model equation errors to construct an approximate error for the components of any proposed model solution.[9]

4.1 Approximating a Global Decision Rule Error Bound

Consider a bounded region, $\mathcal{A}$, that contains all iterated conditional expectations paths. By bounding the largest deviation in the paths for the Δz_t^p, we can bound the largest deviation in a proposed solution from the exact solution.

Given an exact solution $x_t^\star = g^\star(x_{t-1}, \epsilon_t)$ define the conditional expectations function,

$$G^\star(x) \equiv \mathcal{E}\left[g^\star(x, \epsilon)\right]$$

then with

$$E_t x_{t+1}^\star = G^\star(g^\star(x_{t-1}, \epsilon_t))$$

we have

$$\mathbb{M}(x_{t-1}^\star, x_t^\star, E_t x_{t+1}^\star, \epsilon_t) = 0 \ \forall (x_{t-1}, \epsilon_t)$$
$$x_t^\star(x_{t-1}, \epsilon_t) \in R^L \ \ \left\|x_t^\star(x_{t-1}, \epsilon_t)\right\|_2 \leq \overline{\mathcal{X}} \ \forall t \ > 0$$

Now consider a proposed solution for the model, $x_t^p = g^p(x_{t-1}, \epsilon_t)$ define $G^p(x) \equiv \mathcal{E}\left[g^p(x, \epsilon)\right]$ so that

$$E_t x_{t+1} = G^p(g^p(x_{t-1}, \epsilon_t))$$
$$\mathbf{e}_t^p(x_{t-1}, \epsilon) \equiv \mathbb{M}(x_{t-1}, x_t^p, E_t x_{t+1}^p, \epsilon_t)$$

$$\left\|x_t^\star(x_{t-1}, \epsilon) - x_t^p(x_{t-1}, \epsilon)\right\|_\infty \leq \max_{\{x_-, \epsilon\}} \left\|(I - F)^{-1} \phi \mathbb{M}(x_-, g^p(x_-, \epsilon), G^p(g^p(x_-, \epsilon)), \epsilon)\right\|_2$$

which can be approximated by

$$\max_{x_{t-1}, \epsilon_t} \left(\phi \mathbf{e}_t^p(x_{t-1}, \epsilon)\right)^2$$

For proof see Appendix A.

[9] This work contrasts with the analysis provided in (Judd et al., 2017a; Peralta-Alva and Santos, 2014; Santos and Peralta-alva, 2005; Santos, 2000). Their approaches identify Euler Equation Errors, but use statistical techniques to estimate the impact of these errors on solution accuracy. Here I provide a formula for the approximate error that does not require statistical inference.

4.2 Approximating Local Decision Rule Errors

Given a proposed model solution define the error in x_t at (x_{t-1}, ϵ_t)

$$\mathbf{e}(x_{t-1}, \epsilon_t) \equiv \mathbf{x}^\star(x_{t-1}, \epsilon_t) - \mathbf{x}^p(x_{t-1}, \epsilon_t)$$

compute the conditional expectations functions for the decision rule and use them to generate a path to use with a linear reference model $\mathcal{L} \equiv \{H, \psi_\epsilon, \psi_c; B, \phi, F\}$ in 2 to approximate $\mathbf{e}$.

$$\mathbf{X}^p(x) \equiv \mathcal{E}\left[\mathbf{x}^p(x, \epsilon)\right].$$

$$x_{t+k+1} = \mathbf{X}^p(x_{t+k}) \; k = 1, \ldots, K$$

We can define functions $\mathbf{z}_t^p$ by

$$\begin{aligned} \mathbf{z}_t^p &\equiv \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t) \\ \mathbf{z}_{t+k}^p &\equiv \mathbb{M}(x_{t+k-1}, x_{t+k}, x_{t+k+1}, 0) \\ \hat{\mathbf{e}}(x_{t-1}, \epsilon_t) &\equiv \sum_{\nu=0}^{K} F^s \phi \mathbf{z}_{t+\nu}^p \end{aligned}$$

4.3 Assessing the Error Approximation

This section reports on the results of an experiment evaluating how well the formula performs. I compute the known decision rule for the RBC model with parameters $\{\alpha = 0.36, \delta = 0.95, \eta = 1, \rho = 0.95, \sigma = 0.01\}$. I consider this the fixed "true" model.

I generate 10 other settings for the model parameters by choosing parameters from the following ranges.

$$0.20 < \alpha < 0.40, \;\; 0.85 < \delta < 0.99, \;\; 0.85 < \rho < 0.99, \;\; 0.005 < \sigma < 0.015$$

producing the following 10 model specifications:

α	δ	η	ρ	σ
0.35	0.92875	1.	0.957188	0.00875
0.275	0.9725	1.	0.906875	0.01375
0.375	0.8675	1.	0.924375	0.01125
0.225	0.94625	1.	0.891563	0.00625
0.325	0.91125	1.	0.944063	0.006875
0.2625	0.922187	1.	0.98125	0.011875
0.3625	0.887187	1.	0.85875	0.014375
0.2125	0.965938	1.	0.965938	0.009375
0.3125	0.860938	1.	0.878438	0.008125
0.2375	0.904688	1.	0.933125	0.013125

I linearized each of the ten models at their respective stochastic steady states producing 10 linear reference models $\mathcal{L}_i$ and 10 decision rules based on the linearization. As a result I have $100 = 10 \times 10$ combinations of (inaccurate by design) decision rule functions and linear reference models to use in the error formula experiments.

I generate 30 representative points, $(k_{i,t-1}, \theta_{i,t-1}, \epsilon_{i,t})$, from the ergodic set for the fixed true reference model using the Niederreiter quasi-random algorithm. For each of the 100 pairings of decision rule and linear reference model, I compute the error approximation at each of the 30 representative points and regress the actual error, computed using the known analytic solution, and the predicted error from the error formula.

$$\mathbf{e_i} = \alpha + \beta \hat{\mathbf{e}}_\mathbf{i} + u_i$$

Although the true model and the true decision rule stays fixed in the experiment, the proposed decision rules and the linear reference model both change.

Figures 6 and 7 present the 100 R^2 and estimated variance and Figures 8 and 9 presents the 100 resulting regression coefficients for a variety of values for K, of number of terms in the error formula, ($K = 0, 1, 10, 30$)[10] The regressions with $K = 0$ correspond to the maximand of the global error formula. The figures show that the formula does a good job of predicting the error produced by the inaccurate decision rule functions. Even with $K = 0$, using only one term in the error formula, still provides useful information about the magnitude of the discrepancy between the proposed decision rules values and true values. The R^2 values are all above 60 percent and are often near one. The β coefficients are generally close to one and the constant tends to be close to zero.

[10]I don't display results for θ_t because the formula predicts the error in θ_t perfectly since it is determined by a backward looking equation.

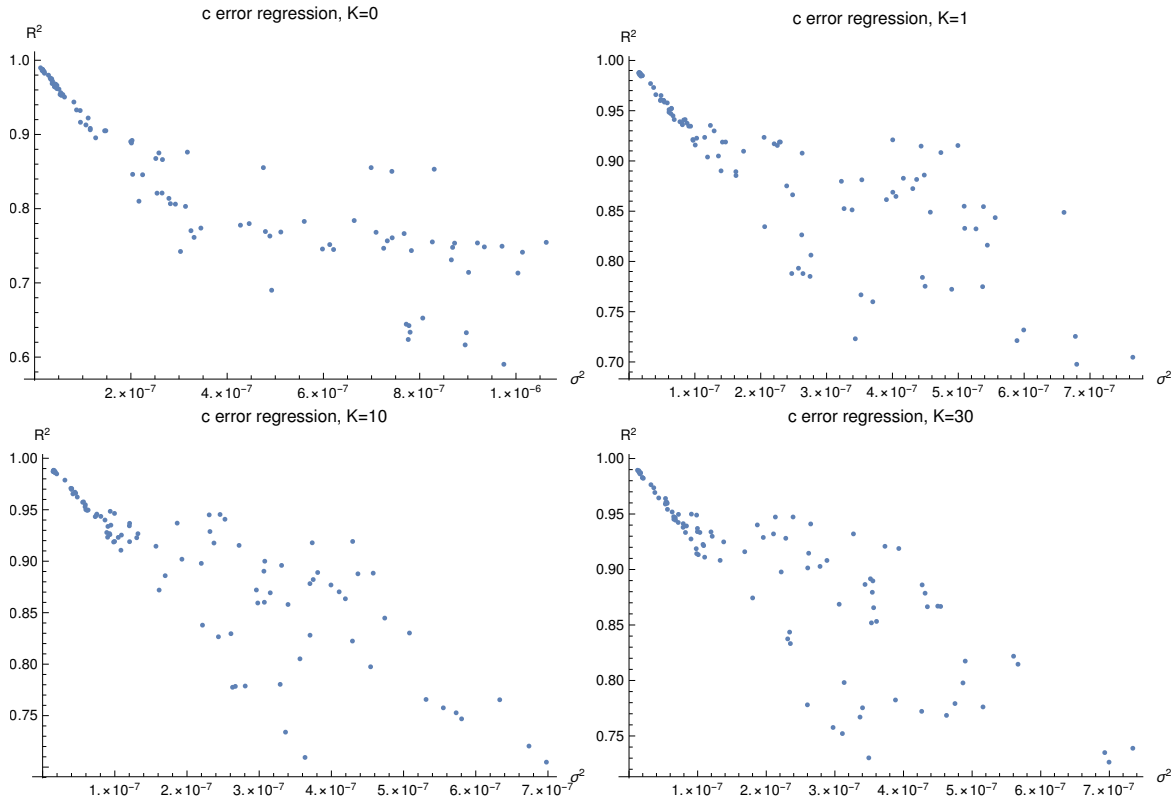

Figure 6: $c_t\left(\hat{\sigma^2}, R^2\right)$

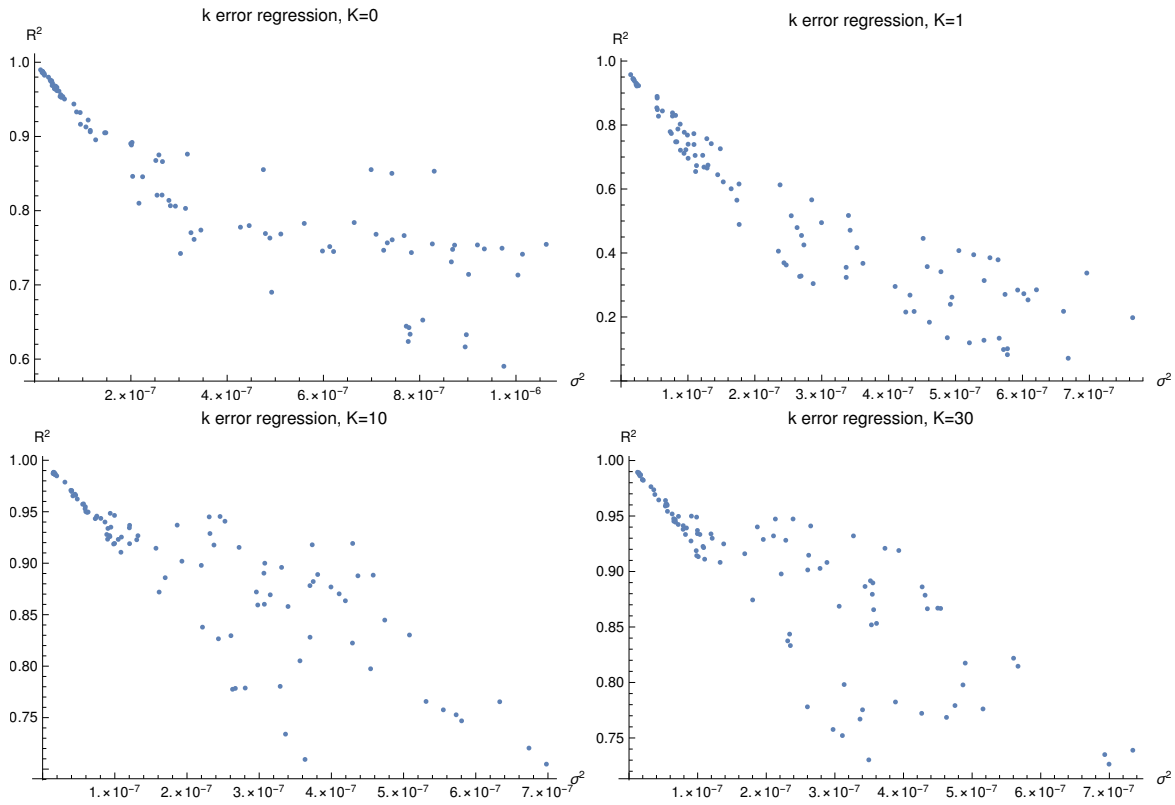

Figure 7: $k_t\left(\hat{\sigma^2}, R^2\right)$

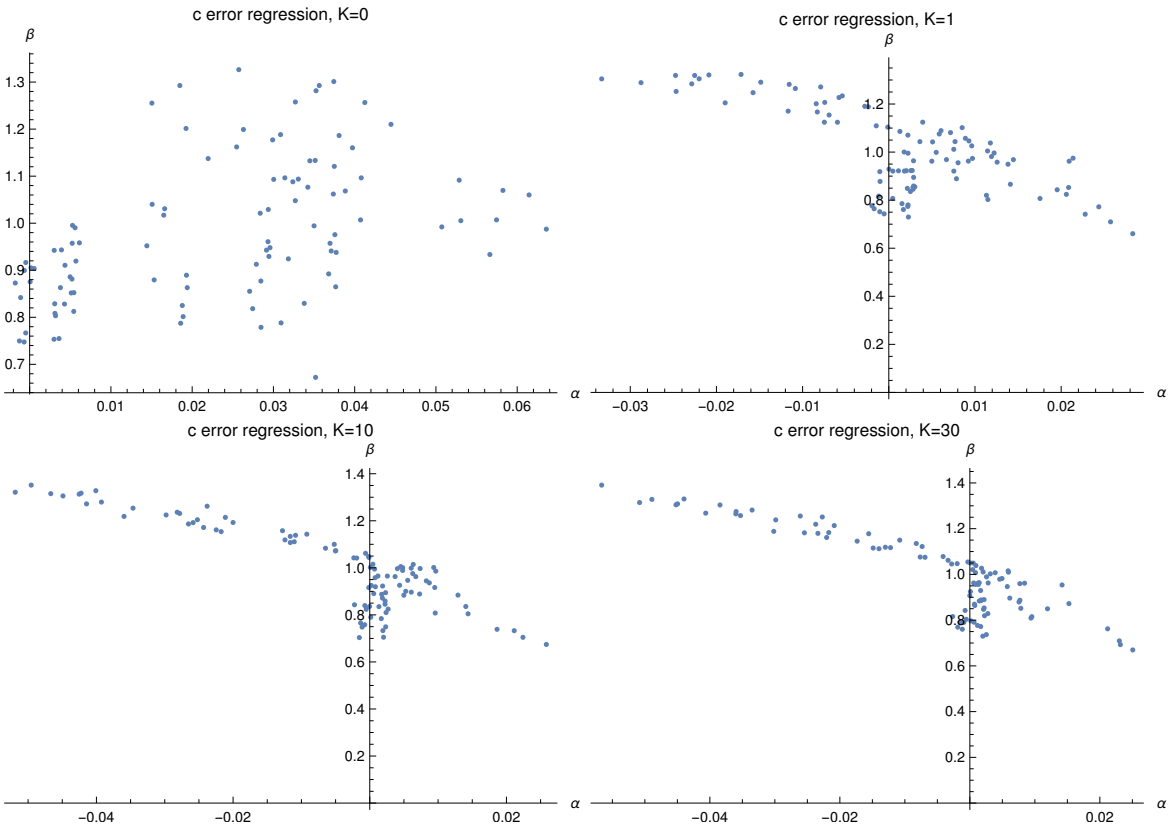

Figure 8: $c_t(\alpha, \beta)$

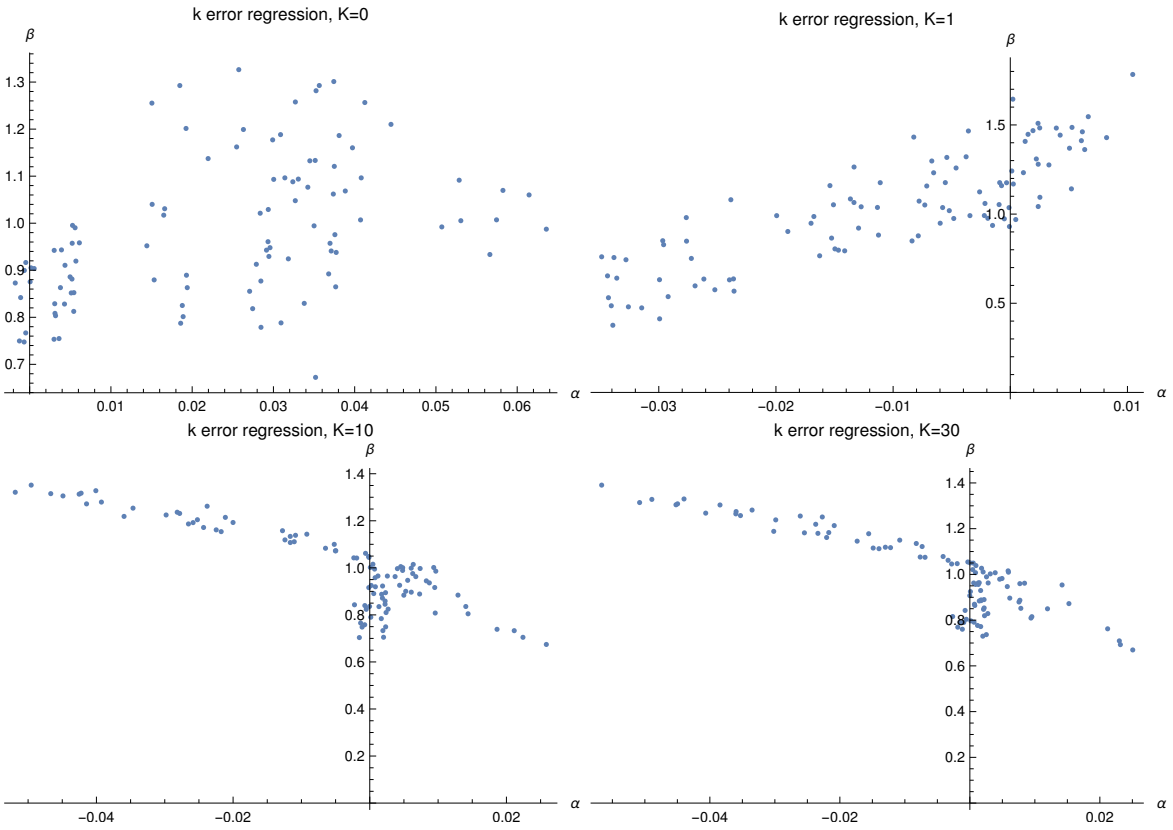

Figure 9: $k_t(\alpha, \beta)$

5 Improving Proposed Solutions

5.1 Some Practical Considerations for Applying the Series Formula

I assume that all models of interest can be characterized by a finite number of equations systems. I will require that, for any given (x_{t-1}, ϵ_t), this collection of equation systems produces a unique solution for x_t. This formulation will allow us to apply the technique to models with occasionally binding constraints and models with regime switching. I will only consider time invariant model solutions $x_t = \mathbf{x}(x_{t-1}, \epsilon)$. Given a decision rule $\mathbf{x}(x_{t-1}, \epsilon)$, denote its conditional expectation $\mathbf{X}(x_{t-1}) \equiv \mathcal{E}[\mathbf{x}(x_{t-1}, \epsilon)]$. Using the convention described in section 3.2, I will include enough auxiliary variables so that I can correctly compute expected values for model variables by applying the law of iterated expectations.

It will be convenient for describing the algorithms to write the models in the form

$$\{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}]) | (\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}$$

where, each

$$\mathbb{C}_i \equiv \{(\mathbf{b}_i(x_{t-1}, \epsilon_t), \mathbb{M}_i(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}]) = 0, \mathbf{c}_i(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}]))\}$$

represents a set of model equations, $\mathbb{M}_i$, with Boolean valued gate functions, $\mathbf{b}_i, \mathbf{c}_i$. In the algorithm's implementation, the $\mathbf{b}_i$ will be a Boolean function precondition and the $\mathbf{c}_i$ will be a Boolean function post-condition for a given equation system. If some $\mathbf{b}_i$ is false, then there is no need to try to solve model i. If $\mathbf{c}_i$ is false, the system has produced an unacceptable solution which should be ignored. I suggest organizing equations using pre and post conditions as this can help with parallelizing a solution regardless of whether or not one uses my series representation. The $\mathbf{d}$ will be a function for choosing among the candidate $x_t(x_{t-1}, \epsilon_t)$ values. The function $\mathbf{d}$ uses the truth values from the $(\mathbf{b}_\cdot, \mathbf{c}_\cdot)$ and the possible values of the state variables to select one and only one x_t. Thus, we will have an equation system and corresponding gatekeeper logical expressions indicating which equation system is in force for producing the unique solution for a given (x_{t-1}, ϵ_t).

Examples of such systems include the simple RBC model from Section 3.2. Other examples include models with occasionally binding constraints where solutions exhibit complementary slackness conditions, or models with regime switching. See Section 6 for a model with occasionally binding constraints and Section B for an example of a specification for regime switching.

5.2 How My Approach Differs From the Usual Approach

It may be worthwhile at this point to briefly compare and contrast the approach I propose here with what is typically done for solving dynamic stochastic models. As with the standard approach, I characterize the solutions using a linearly weighted sums of orthogonal polynomials and solve the model equations at predetermined collocation points. However, in this new algorithm, I augment the model Euler equations with equations based on 2 linking the conditional expectations for future values in the proposed solution with the time t value

of the model variables. As a result, the algorithm determines linearly weighted polynomials characterizing the trajectories for the usual variables $x_t(x_{t-1}, \epsilon_t)$ as well as new $z_t(x_{t-1}, \epsilon_t)$ variables. These new constraints help keep proposed solutions bounded during the solution iterations thus improving algorithm reliability and accuracy.

5.3 Algorithm Overview

I closely follow the techniques outlined in (Judd et al., 2013, 2014). As a result, much of what must be done to use this algorithm is the same as for the usual approach. One must specify an initial guess for decision rule $\hat{\mathbf{x}}(x_{t-1}, \epsilon, k)$ and obtain conditional expectations functions. I use the an-isotropic Smolyak Lagrange interpolation with adaptive domain. I precompute all of the conditional expectations for the integrals as outlined in (Judd et al., 2017b) so that the time t computations 5.1 are all deterministic.

There are number of new steps. One must specify a linear reference model. One must choose a value for K the number of terms in the series representation. There are trade-offs. Fewer means more truncation error. More terms corresponds to more accuracy when the decision rule is accurate but More terms is computationally more expensive. The initial guess is typically some distance away from the solution. Consequently the terms in the series will be incorrect. The Smolyak grid adds more error to the calculation. If there are many grid points it is more likely that increasing the K can improve the quality of the solution. If the grid is too sparse, increasing K will likely be less useful.

5.3.1 Single Equation System Case

For now, consider a single model equation system case with no Boolean gates. A description of the actual, multi-system implementation follows. We seek

$$\mathbb{M}(x_{t-1}, \mathbf{x}^\star(x_{t-1}, \epsilon_t), \mathbf{X}^\star(\mathbf{x}^\star(x_{t-1}, \epsilon_t)), \epsilon_t) = 0 \ \forall (x_{t-1}, \epsilon_t).$$

Given a proposed model solution

$$x_t = \mathbf{x}^p(x_{t-1}, \epsilon_t)$$

compute

$$\mathbf{X}^p(x_{t-1}) \equiv \mathcal{E}\left[\mathbf{x}^p(x_{t-1}, \epsilon_t)\right].$$

We will use a linear reference model $\mathcal{L} \equiv \{H, \psi_\epsilon, \psi_c; B, \phi, F\}$ to construct a series of $\mathbf{z}^p$ functions that help improve the accuracy of the proposed solution.

We can define functions $\mathbf{z}^p, \mathbf{Z}^p$ by

$$\mathbf{z}^p(x_{t-1}, \epsilon) \equiv H \begin{bmatrix} x_{t-1} \\ \mathbf{x}^p(x_{t-1}, \epsilon) \\ \mathbf{X}^p(\mathbf{x}(x_{t-1}, \epsilon)) \end{bmatrix} + \phi_\epsilon \epsilon_t + \phi_c$$

$$\mathbf{Z}^p(x_{t-1}) \equiv \mathcal{E}\left[\mathbf{z}^p(x_{t-1}, \epsilon)\right]$$

- Algorithm loop begins here. Define conditional expectations paths for x_t, z_t

$$x_{t+k+1} = \mathbf{X}(x_{t+k}), \quad z_{t+k+1} = \mathbf{Z}(x_{t+k}) \quad \forall k \geq 0$$

Using the $\mathbf{z}^p(x_{t-1}, \epsilon)$ Series

- We get expressions for x_t, $\mathcal{E}[x]_{t+1}$ consistent with $\mathcal{L}$ and the conditional expectations path

$$\mathcal{X}_t = Bx_{t-1} + \phi\psi_\epsilon\epsilon + (I - F)^{-1}\phi\psi_c + \sum_{\nu=0}^{\infty} F^s\phi\mathbf{Z}(x_{t+\nu})$$

$$\mathcal{E}[\mathcal{X}_{t+1}] = B\mathcal{X}_t + (I - F)^{-1}\phi\psi_c + \sum_{\nu=0}^{\infty} F^{\nu-1}\phi\mathbf{Z}(x_{t+\nu}) \quad \forall t, k \geq 0$$

- Use the model equations, $\mathbb{M}(x_{t-1}, x_t, \mathcal{E}[x_{t+1}], \epsilon) = 0$ and $x_t(x_{t-1}, \epsilon_t) = \mathcal{X}_t(x_{t-1}, \epsilon_t)$ to determine $\mathbf{x}^{p'}(x_{t-1}, \epsilon), \mathbf{z}^{p'}(x_{t-1}, \epsilon)$
- $\mathbf{x}^p(x_{t-1}, \epsilon) = \mathbf{x}^{p'}(x_{t-1}, \epsilon), \mathbf{z}^p(x_{t-1}, \epsilon) = \mathbf{z}^{p'}(x_{t-1}, \epsilon)$ – Repeat loop.

5.3.2 Multiple Equation System Case

An outline of the current Mathematica implementation of the algorithm follows. Table 1 provides notation. Figure 10 provides a graphic depiction of the function call tree. For more

$\mathcal{L} \equiv \{H, \psi_\epsilon, \psi_c; B, \phi, F\}$ The linear reference model

$\mathbf{x}^p(x_{t-1}, \epsilon_t)$ The proposed decision rule x_t components

$\mathbf{z}^p(x_{t-1}, \epsilon_t)$ The proposed decision rule z_t components

$\mathbf{X}^p(x_{t-1})$ The proposed decision rule conditional expectations x_t components

$\mathbf{Z}^p(x_{t-1})$ The proposed decision rule their conditional expectations z_t components

κ One less than the number of terms in series representation($\kappa >= 0$)

$\mathbb{S}$ Smolyak an-isotropic grid polynomials $(p_1(x, \epsilon), \ldots, p_N(x, \epsilon))$ and their conditional expectations $(P_1(x), \ldots, P_N(x))$

$\mathbb{T}$ Model evaluation points Neidereiter sequence in the model variable ergodic region.

γ $\{\mathbf{x}(x_{t-1}, \epsilon_t), \mathbf{z}(x_{t-1}, \epsilon_t)\}$ collects the decision rule components

$\mathbb{G}$ $\{\mathbf{x}(x_{t-1}, \epsilon_t), \mathbf{z}(x_{t-1}, \epsilon_t)\}$ collects the decision rule conditional expectations components

$\mathbb{H}$ $\{\gamma, \mathbb{G}\}$ collects decision rule information

$\{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}$ The equation system $\mathcal{R}^{3N_x+N_\epsilon+N_x} \to \mathcal{R}^{N_x}$

Table 1: Algorithm Notation

algorithmic detail, see Appendix C. The current implementation has a couple of atypical features. Many of the calculations exploit Mathematica's capability to blend numerical and symbolic computation and to easily use functions as arguments for other functions. In addition, the implementation exploits parallelism at several points. The parallel features also apply when employing the usual technique for solving the set types of models. Below, I note where these features play a role.

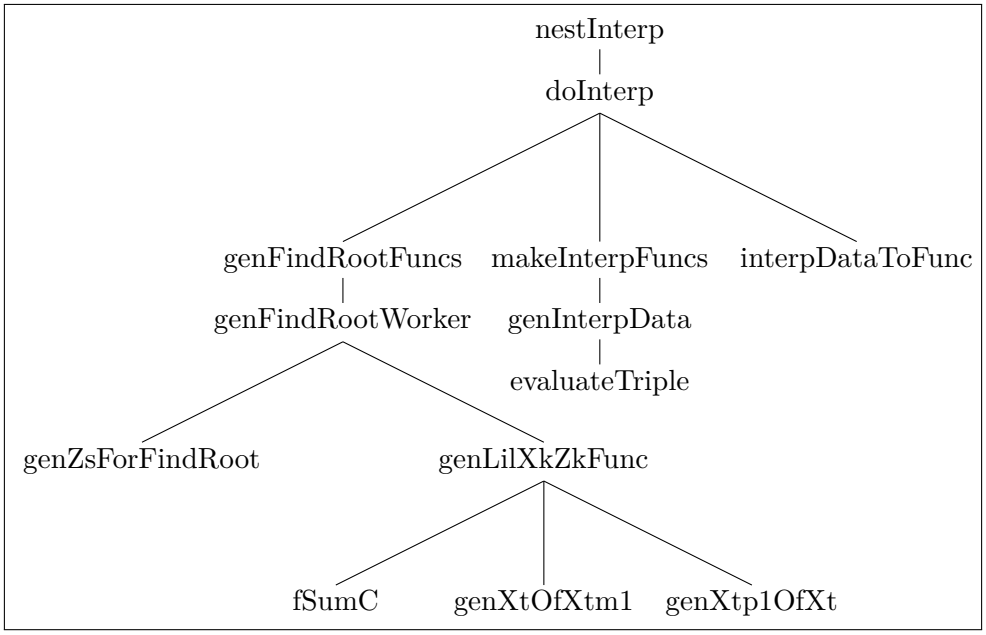

Figure 10: Function Call Tree

nestInterp — Computes a sequence of decision rule and conditional expectation rule functions terminating when the evaluation of the decision rule functions at the tests points no longer changes much.

doInterp — **Symbolically** generates functions that return a unique x_t for any value of (x_{t-1}, ϵ_t) for the family model equations, $\{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}$ and uses these functions to construct interpolating functions approximating the decision rules.

genFindRootFuncs — The function can use 2 or omit it thus implementing the usual approach for solving these models.

The routine **symbolically** generates a collection of function triples and an outer model solution selection function. Each of the function constructions can be done in **parallel**. Each of the triples returns the Boolean gate values and a unique value for x_t for any given value of (x_{t-1}, ϵ_t) the selection function chooses one solution from the set of model equations. The routine subsequently uses these functions to construct interpolating functions approximating the decision rules. Sub-components of this step exploit parallelism.

genLilXkZkFunc — **Symbolically** creates a function that uses an initial Z_t path to generate a function of $(x_{t-1}, \epsilon_t, z_t)$ providing (potentially symbolic) inputs

$$\begin{bmatrix} x_{t-1} \\ x_t \\ E_t(x_{t+1}), \epsilon_t) \end{bmatrix}$$

for model system equations $\mathbb{M}$. This routine provides the information for constructing $x_t(x_{t-1}, \epsilon_t)$ using the series expression 2.

makeInterpFuncs — Solves model equations at collocation points producing interpolation data and subsequently returns the interpolating functions for the decision rule and decision rule conditional expectation. This can be done in **parallel**.

5.4 Approximating a Known Solution: $\eta = 1, U(c) = Log(c)$

This subsection uses the series representation to compute the solution for the case when the exact solution is known and to compare the various approximations to the known actual solution. In what follows, both the traditional and the new approach use an-isotropic Smolyak polynomial function approximation with precomputed expectations. Figure 11 characterizes the use of the parallelotope transformation described in (Judd et al., 2013). The left panel shows the 200 values of k_t and θ_t resulting from a stochastic simulation of the the known decision rule. The right panel shows the transformed variables that constitute an improved set of variables for constructing function approximation values.

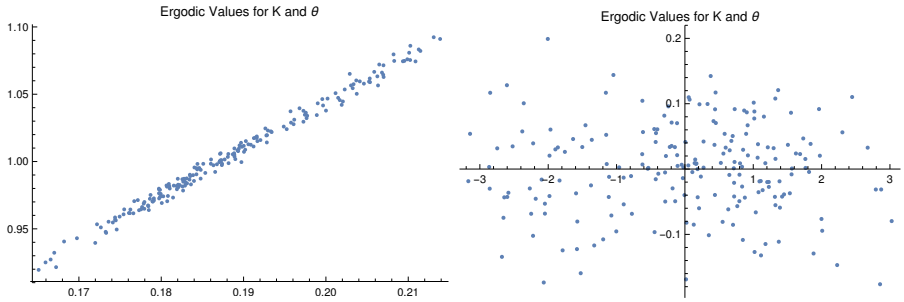

Figure 11: The Ergodic Values for k_t, θ_t

$$\bar{X} = \begin{pmatrix} 0.18853 \\ 1.00466 \\ 0. \end{pmatrix}$$

$$\sigma_X = \begin{pmatrix} 0.0112443 \\ 0.0387807 \\ 0.01 \end{pmatrix}$$

$$V = \begin{pmatrix} -0.707107 & -0.707107 & 0. \\ -0.707107 & 0.707107 & 0. \\ 0. & 0. & 1. \end{pmatrix}$$

$$\max U = \begin{pmatrix} 3.02524 \\ 0.199124 \\ 3. \end{pmatrix}$$

$$\min U = \begin{pmatrix} -3.16879 \\ -0.17629 \\ -3. \end{pmatrix}$$

Table 2 shows the evaluation points when the Smolyak polynomial degrees of approximation were (1,1,1). Table 3 shows the decision rule prescription for (c_t, k_t, θ_t) at each evaluation point. Table 4 shows the log of the absolute value of the actual error in the decision rule at each evaluation point. Table 5 shows the log of the absolute value of the estimated error in the decision rule at each evaluation points. Note that the formula provides accurate estimates of the error component by component.

Table 6 shows the evaluation points when the Smolyak polynomial degrees of approximation were (2,2,2). Table 7 shows the decision rule prescription for (c_t, k_t, θ_t) at each evaluation point. Table 8 shows the log of the absolute value of the actual error in the decision rule at each evaluation point. Table 9 shows the log of the absolute value of the estimated error in the decision rule at each evaluation points. Note that the formula provides accurate estimates of the error component by component.

Each of the estimated errors were computed with $K = 5$. Table 10 shows the effect of increasing value of K. Generally increasing K increases the accuracy of the solution. The value of K has more effect when the Smolyak grid is more densely populated with collocation points.

$\begin{bmatrix} k_1 & \theta_1 & \epsilon_1 \\ & \vdots & \\ k_{30} & \theta_{30} & \epsilon_{30} \end{bmatrix}$		
0.16818	0.942376	−0.0251104
0.210392	1.08282	0.0232085
0.181393	0.968212	−0.0090041
0.1949	1.017	0.0111288
0.188668	1.00876	−0.0210838
0.20145	1.06007	0.0272351
0.17245	0.945462	−0.00497752
0.214179	1.0876	0.0191819
0.195059	1.03442	−0.0130307
0.182278	0.983104	0.00307563
0.208273	1.06025	−0.029137
0.166906	0.916853	0.00106234
0.20192	1.05244	−0.0311503
0.187205	1.00787	0.0171686
0.213199	1.08502	−0.015044
0.17147	0.942887	0.0252218
0.179847	0.985589	−0.00699081
0.194563	1.03016	0.00911549
0.165563	0.915543	−0.0230971
0.20705	1.05852	0.0211952
0.173322	0.954406	−0.0110174
0.2136	1.1016	0.00508892
0.184601	0.986988	−0.0271237
0.198833	1.03324	0.0131421
0.20721	1.07594	−0.0190705
0.166932	0.928749	0.0292484
0.192926	1.0059	−0.00296424
0.179118	0.958169	0.0141487
0.172672	0.949185	−0.0180639
0.212949	1.09638	0.030255

Table 2: RBC Known Solution: Model Evaluation Points d=(1,1,1)

$\begin{bmatrix} c_1 & k_1 & \theta_1 \\ & \vdots & \\ c_{30} & k_{30} & \theta_{30} \end{bmatrix}$		
0.318386	0.16551	0.9202
0.413447	0.214841	1.10215
0.341848	0.177697	0.960777
0.375209	0.195014	1.02742
0.35634	0.185242	0.987273
0.400686	0.208232	1.08481
0.329563	0.171293	0.943165
0.416315	0.216325	1.1025
0.372502	0.193618	1.01959
0.351946	0.182927	0.98707
0.384948	0.200107	1.02813
0.31841	0.165481	0.92202
0.377048	0.196011	1.01878
0.368995	0.19178	1.02495
0.402167	0.209001	1.06547
0.339185	0.176282	0.97149
0.347449	0.180596	0.979286
0.378776	0.196859	1.03785
0.308332	0.160276	0.896658
0.401836	0.208829	1.07707
0.330982	0.172039	0.945622
0.415597	0.215941	1.10137
0.343927	0.178809	0.960659
0.384305	0.199732	1.04488
0.393281	0.204401	1.05291
0.332894	0.173006	0.962198
0.36484	0.189639	1.00262
0.345376	0.179513	0.974651
0.326232	0.169582	0.933652
0.422656	0.219618	1.12227

Table 3: RBC Known Solution: Values at Evaluation Points d=(1,1,1)

$\begin{bmatrix} \ln(\lvert \mathbf{e}_{c,1} \rvert) \\ \\ \ln(\lvert \mathbf{e}_{c,30} \rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert \mathbf{e}_{k,1} \rvert) \\ \vdots \\ \ln(\lvert \mathbf{e}_{k,30} \rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert \mathbf{e}_{\theta,1} \rvert) \\ \\ \ln(\lvert \mathbf{e}_{\theta,30} \rvert) \end{bmatrix}$
−6.86423	−7.61023	−6.2776
−6.77469	−7.25654	−6.193
−8.27193	−9.26988	−7.90952
−9.53366	−9.94641	−9.07674
−9.70109	−11.0692	−10.8193
−7.01131	−7.52677	−6.4226
−8.62144	−9.43568	−8.12434
−6.93069	−7.36841	−6.2991
−8.82105	−9.39136	−7.96867
−9.48549	−9.94897	−9.04695
−6.83337	−7.45765	−6.41963
−11.193	−13.9426	−8.33167
−7.13458	−7.70834	−6.51919
−9.46667	−10.6151	−10.154
−7.13317	−7.97018	−6.71715
−6.76168	−7.44531	−6.20438
−9.46294	−10.7345	−8.60101
−8.99962	−9.32606	−8.36754
−6.5744	−7.28112	−6.0606
−7.26443	−7.74753	−6.65645
−7.95047	−8.75065	−7.34261
−7.90465	−8.02499	−7.40682
−7.78668	−8.88177	−7.3262
−8.44035	−8.86441	−7.83514
−7.21479	−7.97368	−6.58639
−6.40774	−7.09877	−5.86537
−11.8365	−11.1009	−11.8397
−7.81106	−8.43094	−7.01803
−7.28745	−8.06534	−6.74087
−6.33557	−6.84989	−5.76803

Table 4: RBC Known Solution: Actual Errors d=(1,1,1)

$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{c,1}\rvert) \\ \\ \ln(\lvert\hat{\mathbf{e}}_{c,30}\rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{k,1}\rvert) \\ \vdots \\ \ln(\lvert\hat{\mathbf{e}}_{k,30}\rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{\theta,1}\rvert) \\ \\ \ln(\lvert\hat{\mathbf{e}}_{\theta,30}\rvert) \end{bmatrix}$
−6.84011	−7.59703	−6.2776
−6.79189	−7.33194	−6.193
−8.27296	−9.26585	−7.90952
−9.54759	−9.96466	−9.07674
−9.72214	−11.142	−10.8193
−7.019	−7.58967	−6.4226
−8.61034	−9.41771	−8.12434
−6.95566	−7.44593	−6.2991
−8.83746	−9.43331	−7.96867
−9.48388	−9.95406	−9.04695
−6.8561	−7.50703	−6.41963
−10.8845	−13.6119	−8.33167
−7.15654	−7.75234	−6.51919
−9.48715	−10.5641	−10.154
−7.16809	−8.02412	−6.71715
−6.74694	−7.43005	−6.20438
−9.46173	−10.7234	−8.60101
−9.00034	−9.36818	−8.36754
−6.55256	−7.25512	−6.0606
−7.28911	−7.80039	−6.65645
−7.93653	−8.73939	−7.34261
−7.87653	−8.13406	−7.40682
−7.79071	−8.88802	−7.3262
−8.45665	−8.89927	−7.83514
−7.25059	−8.02416	−6.58639
−6.38709	−7.07562	−5.86537
−11.9887	−11.1639	−11.8397
−7.8057	−8.42757	−7.01803
−7.27379	−8.05278	−6.74087
−6.35248	−6.93629	−5.76803

Table 5: RBC Known Solution: Error Approximations d=(1,1,1)

$\begin{bmatrix} k_1 \\ \\ k_{30} \end{bmatrix}$	$\begin{matrix} \theta_1 \\ \vdots \\ \theta_{30} \end{matrix}$	$\begin{bmatrix} \epsilon_1 \\ \\ \epsilon_{30} \end{bmatrix}$
0.175411	1.00081	−0.00861854
0.182233	1.01265	−0.00121566
0.181807	0.987487	−0.00615091
0.183086	0.994888	−0.00306638
0.179142	1.00488	−0.00800163
0.179142	1.01672	−0.000598756
0.178715	0.991558	−0.00553401
0.184685	1.00636	−0.00183257
0.179142	1.0108	−0.00676782
0.179142	0.998959	−0.00430019
0.185538	0.997479	−0.00923544
0.180208	0.980456	−0.00460865
0.18138	1.00821	−0.0095439
0.177969	1.00821	−0.00214102
0.184365	1.00673	−0.00707627
0.178396	0.991928	−0.000907209
0.176263	1.00821	−0.00584246
0.179675	1.00821	−0.00337483
0.179248	0.983046	−0.00831008
0.184792	0.999329	−0.00152412
0.177436	0.997479	−0.00645937
0.180847	1.02116	−0.00399174
0.180421	0.995999	−0.00892699
0.182979	0.998959	−0.00275793
0.180847	1.01524	−0.00769318
0.177436	0.991558	−0.000290303
0.183832	0.990077	−0.00522555
0.18202	0.984526	−0.0026037
0.178049	0.994426	−0.00753895
0.18146	1.01811	−0.000136076

Table 6: RBC Known Solution: Model Evaluation Points d=(2,2,2)

$\begin{bmatrix} k_1 & \theta_1 & \epsilon_1 \\ & \vdots & \\ k_{30} & \theta_{30} & \epsilon_{30} \end{bmatrix}$		
0.175411	1.00081	−0.00861854
0.182233	1.01265	−0.00121566
0.181807	0.987487	−0.00615091
0.183086	0.994888	−0.00306638
0.179142	1.00488	−0.00800163
0.179142	1.01672	−0.000598756
0.178715	0.991558	−0.00553401
0.184685	1.00636	−0.00183257
0.179142	1.0108	−0.00676782
0.179142	0.998959	−0.00430019
0.185538	0.997479	−0.00923544
0.180208	0.980456	−0.00460865
0.18138	1.00821	−0.0095439
0.177969	1.00821	−0.00214102
0.184365	1.00673	−0.00707627
0.178396	0.991928	−0.000907209
0.176263	1.00821	−0.00584246
0.179675	1.00821	−0.00337483
0.179248	0.983046	−0.00831008
0.184792	0.999329	−0.00152412
0.177436	0.997479	−0.00645937
0.180847	1.02116	−0.00399174
0.180421	0.995999	−0.00892699
0.182979	0.998959	−0.00275793
0.180847	1.01524	−0.00769318
0.177436	0.991558	−0.000290303
0.183832	0.990077	−0.00522555
0.18202	0.984526	−0.0026037
0.178049	0.994426	−0.00753895
0.18146	1.01811	−0.000136076

Table 7: RBC Known Solution: Values at Evaluation Points d=(2,2,2)

$\begin{bmatrix} \ln(\lvert \mathbf{e}_{c,1} \rvert) \\ \\ \ln(\lvert \mathbf{e}_{c,30} \rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert \mathbf{e}_{k,1} \rvert) \\ \vdots \\ \ln(\lvert \mathbf{e}_{k,30} \rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert \mathbf{e}_{\theta,1} \rvert) \\ \\ \ln(\lvert \mathbf{e}_{\theta,30} \rvert) \end{bmatrix}$
−13.6548	−13.6548	−14.1969
−18.0614	−13.7477	−14.7744
−17.2538	−16.064	−17.1189
−18.2334	−14.931	−18.4609
−14.9375	−15.0484	−18.06
−13.3718	−13.4466	−14.3339
−15.3357	−15.1409	−16.6528
−13.4818	−17.055	−18.6856
−16.5151	−17.974	−16.0224
−16.8629	−17.1652	−16.9262
−15.0336	−13.0546	−15.0929
−14.8104	−15.1068	−17.0829
−13.9454	−15.0733	−15.3665
−17.0059	−15.0225	−16.8123
−14.3533	−13.6955	−16.1402
−14.697	−15.2052	−15.5889
−15.6999	−16.5298	−16.5502
−15.469	−15.8159	−16.1557
−12.9005	−17.0451	−15.5094
−13.407	−14.5127	−15.9061
−15.605	−15.0689	−15.5069
−14.5434	−14.4278	−15.1171
−14.8235	−14.8132	−16.6327
−15.0699	−15.1443	−17.7803
−13.5813	−14.7228	−14.6813
−15.1304	−15.2556	−15.467
−17.3355	−17.4808	−20.5415
−13.2332	−15.2877	−16.1059
−15.668	−14.9417	−15.2354
−14.2264	−12.8483	−14.2733

Table 8: RBC Known Solution: Actual Errorsd=(2,2,2)

$\ln(\lvert\hat{\mathbf{e}}_{c,1}\rvert)$	$\ln(\lvert\hat{\mathbf{e}}_{k,1}\rvert)$	$\ln(\lvert\hat{\mathbf{e}}_{\theta,1}\rvert)$
	$\vdots$	
$\ln(\lvert\hat{\mathbf{e}}_{c,30}\rvert)$	$\ln(\lvert\hat{\mathbf{e}}_{k,30}\rvert)$	$\ln(\lvert\hat{\mathbf{e}}_{\theta,30}\rvert)$
−13.5994	−13.7316	−14.1969
−19.713	−13.7563	−14.7744
−17.8538	−15.9394	−17.1189
−18.4186	−14.9247	−18.4609
−14.9453	−15.0569	−18.06
−13.3661	−13.4484	−14.3339
−15.2186	−15.0483	−16.6528
−13.4591	−16.4547	−18.6856
−16.6757	−17.4658	−16.0224
−16.7507	−17.3581	−16.9262
−17.6809	−13.212	−15.0929
−14.8476	−15.0629	−17.0829
−14.1282	−15.8166	−15.3665
−16.8176	−14.9953	−16.8123
−14.7882	−13.8997	−16.1402
−14.5928	−15.0521	−15.5889
−15.8049	−16.3126	−16.5502
−15.479	−15.8001	−16.1557
−12.8385	−15.3986	−15.5094
−13.3789	−14.598	−15.9061
−15.6645	−15.1186	−15.5069
−14.4965	−14.459	−15.1171
−14.7832	−14.7719	−16.6327
−15.0335	−15.1856	−17.7803
−13.7298	−15.3206	−14.6813
−15.2349	−15.1718	−15.467
−17.3423	−17.473	−20.5415
−13.2297	−15.3227	−16.1059
−15.7978	−15.0212	−15.2354
−14.0272	−12.8995	−14.2733

Table 9: RBC Known Solution: Error Approximationsd=(2,2,2)

$\begin{bmatrix} K \end{bmatrix}$	$d = (1,1,1)$	$d = (2,2,2)$	$d = (3,3,3)\Big]$
$NonSeries$	-6.51367	-12.2771	-16.8947
0	-6.0793	-6.26833	-6.29843
1	-6.00805	-6.24202	-6.49835
2	-6.52219	-7.43876	-7.04785
3	-6.57578	-8.03864	-8.32589
4	-6.62831	-9.1522	-9.49314
5	-6.77305	-10.5516	-10.4247
6	-6.38499	-11.6399	-11.455
7	-6.63849	-11.3627	-13.0213
8	-6.38735	-11.7288	-13.9976
9	-6.46759	-11.9826	-15.3372
10	-6.45966	-12.1345	-15.4157
10	-6.77776	-11.5025	-15.8211
15	-6.67778	-12.4593	-15.8899
20	-6.40802	-11.7296	-17.1652
25	-6.53265	-12.1441	-17.1518
30	-6.81949	-11.6097	-16.3748
35	-6.33508	-12.1446	-16.6963
40	-6.52442	-11.4042	-15.6302

Table 10: RBC Known Solution: Impact of Series Length on Approximation Error

5.5 Approximating an Unknown Solution: $\eta = 3$

Table 11 shows the evaluation points when the Smolyak polynomial degrees of approximation were (1,1,1). Table 12 shows the decision rule prescription for (c_t, k_t, θ_t) at each evaluation point. Table 13 shows the log of the absolute value of the estimated error in the decision rule at each evaluation points. Note that the formula provides accurate estimates of the error component by component.

Table 14 shows the evaluation points when the Smolyak polynomial degrees of approximation were (2,2,2). Table 15 shows the decision rule prescription for (c_t, k_t, θ_t) at each evaluation point. Table 9 shows the log of the absolute value of the estimated error in the decision rule at each evaluation points. Note that the formula provides accurate estimates of the error component by component.

$\begin{bmatrix} k_1 & \theta_1 & \epsilon_1 \\ & \vdots & \\ k_{30} & \theta_{30} & \epsilon_{30} \end{bmatrix}$		
0.1489	0.887266	−0.044574
0.233573	1.16936	0.0595096
0.175324	0.939453	−0.00987946
0.202434	1.03737	0.0334887
0.18999	1.02063	−0.0359003
0.215667	1.12355	0.0681832
0.157417	0.893645	−0.00120583
0.241135	1.17907	0.0508359
0.202828	1.07209	−0.0185531
0.177152	0.96917	0.0161414
0.229252	1.12428	−0.0532476
0.146251	0.836349	0.0118046
0.21657	1.10837	−0.0575844
0.187072	1.01878	0.0464991
0.239172	1.17389	−0.0228899
0.155454	0.888463	0.0638464
0.172322	0.973989	−0.00554264
0.201821	1.06358	0.0291519
0.143572	0.833667	−0.0402371
0.226812	1.12076	0.0551727
0.159193	0.911511	−0.0142163
0.240045	1.20694	0.0204782
0.181795	0.977034	−0.0489108
0.210338	1.06995	0.0378255
0.227206	1.15548	−0.0315635
0.146355	0.86005	0.07252
0.198455	1.01516	0.00313098
0.170749	0.919322	0.0399939
0.157874	0.901074	−0.0293951
0.238725	1.19651	0.0746884

Table 11: RBC Approximated Solution: Model Evaluation Points d=(1,1,1)

$\begin{bmatrix} c_1 & k_1 & \theta_1 \\ & \vdots & \\ c_{30} & k_{30} & \theta_{30} \end{bmatrix}$		
0.21611	0.208557	0.847027
0.452893	0.269857	1.22393
0.267239	0.230108	0.931775
0.35028	0.251768	1.07036
0.299961	0.240849	0.983569
0.420887	0.262256	1.18984
0.243253	0.217177	0.896521
0.459129	0.272277	1.22374
0.340827	0.250513	1.04998
0.294783	0.234714	0.986909
0.368953	0.260446	1.06547
0.221402	0.20607	0.854741
0.349843	0.25471	1.04621
0.339335	0.244203	1.0664
0.416676	0.267429	1.14247
0.27496	0.218765	0.960064
0.283425	0.231206	0.96923
0.360306	0.252522	1.09083
0.193846	0.200678	0.799735
0.420092	0.266042	1.17298
0.245256	0.218836	0.900489
0.456834	0.270845	1.21817
0.26908	0.233193	0.929114
0.373581	0.256909	1.10605
0.393659	0.262194	1.11644
0.264319	0.211099	0.942148
0.321357	0.247159	1.01754
0.281918	0.228897	0.964162
0.232855	0.216163	0.875304
0.479992	0.272575	1.26627

Table 12: RBC Approximated Solution: Values at Evaluation Points d=(1,1,1)

$\begin{bmatrix} \ln(\lvert \hat{\mathbf{e}}_{c,1} \rvert) \\ \\ \ln(\lvert \hat{\mathbf{e}}_{c,30} \rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert \hat{\mathbf{e}}_{k,1} \rvert) \\ \vdots \\ \ln(\lvert \hat{\mathbf{e}}_{k,30} \rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert \hat{\mathbf{e}}_{\theta,1} \rvert) \\ \\ \ln(\lvert \hat{\mathbf{e}}_{\theta,30} \rvert) \end{bmatrix}$
−4.38747	−5.	−5.01321
−5.68045	−5.805	−4.89809
−5.10224	−5.33003	−6.6064
−6.34158	−6.62031	−7.87535
−5.60248	−5.64831	−9.6263
−5.7969	−6.18996	−5.11512
−4.92963	−5.07008	−6.83086
−5.88332	−5.88269	−5.01359
−6.63272	−6.15415	−6.68716
−5.69579	−5.56877	−7.76351
−6.081	−5.72439	−5.16437
−4.82324	−4.80882	−7.05272
−6.86202	−5.74095	−5.25257
−6.47222	−6.25808	−9.00805
−5.96599	−6.66276	−5.44834
−8.66584	−4.99997	−4.90498
−5.4139	−5.50185	−7.33409
−6.42036	−7.03866	−7.08005
−4.13978	−4.80089	−4.78296
−6.06664	−6.39354	−5.377
−4.86241	−5.1415	−6.06258
−7.33554	−6.38356	−6.11469
−5.02079	−5.37422	−6.04755
−6.43212	−7.99418	−6.57026
−6.17638	−6.42884	−5.31385
−6.75784	−4.79589	−4.56474
−5.93264	−5.92418	−10.3455
−5.92431	−5.29301	−5.71515
−4.62067	−5.0846	−5.46376
−5.22287	−5.41884	−4.46492

Table 13: RBC Approximated Solution: Error Approximations d=(1,1,1)

$\begin{bmatrix} k_1 & \theta_1 & \epsilon_1 \\ & \vdots & \\ k_{30} & \theta_{30} & \epsilon_{30} \end{bmatrix}$		
0.154714	0.909409	0.0583516
0.238295	1.1837	−0.0441935
0.181916	0.956081	0.0241699
0.208439	1.05216	−0.0185573
0.195384	1.03868	0.0498062
0.220186	1.14073	−0.052739
0.163808	0.913113	0.0156245
0.246242	1.19138	−0.0356481
0.207785	1.08971	0.0327153
0.182983	0.987658	−0.00146639
0.234987	1.13638	0.0668971
0.153414	0.855121	0.00280632
0.221646	1.1239	0.0711698
0.192257	1.03778	−0.0313754
0.244262	1.1865	0.036988
0.161828	0.908228	−0.0484663
0.177563	0.994718	0.0198972
0.206952	1.08084	−0.0142845
0.150574	0.853223	0.0540789
0.232434	1.13348	−0.0399208
0.165188	0.931842	0.0284426
0.244182	1.22206	−0.00573911
0.187804	0.994441	0.0626243
0.216046	1.08454	−0.02283
0.231781	1.17103	0.0455335
0.152787	0.880818	−0.0570117
0.204792	1.02954	0.0113518
0.177553	0.935951	−0.0249663
0.164075	0.921007	0.0433971
0.243068	1.21122	−0.0591481

Table 14: RBC Approximated Solution: Model Evaluation Points d=(2,2,2)

$\begin{bmatrix} k_1 & \theta_1 & \epsilon_1 \\ & \vdots & \\ k_{30} & \theta_{30} & \epsilon_{30} \end{bmatrix}$		
0.274683	0.220094	0.968634
0.40339	0.266729	1.12301
0.296394	0.23513	0.981672
0.335137	0.250659	1.03019
0.358182	0.247172	1.08965
0.365685	0.257823	1.07502
0.263846	0.22194	0.931716
0.417917	0.27018	1.13964
0.3831	0.253685	1.12112
0.299405	0.23603	0.986824
0.451246	0.26553	1.20725
0.23049	0.209622	0.864261
0.437392	0.260092	1.19978
0.312821	0.241638	1.00386
0.465984	0.26895	1.22073
0.236754	0.21458	0.869437
0.309625	0.235153	1.01498
0.350497	0.251486	1.06137
0.246402	0.212757	0.907811
0.376735	0.263313	1.08232
0.277956	0.225197	0.962114
0.454981	0.269165	1.20294
0.337604	0.2424	1.059
0.352851	0.255287	1.05577
0.454299	0.264058	1.21595
0.219639	0.206105	0.837304
0.337981	0.249525	1.03978
0.26425	0.227363	0.9159
0.27897	0.224894	0.965819
0.410798	0.268804	1.13077

Table 15: RBC Approximated Solution: Values at Evaluation Points d=(2,2,2)

$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{c,1}\rvert) \\ \\ \ln(\lvert\hat{\mathbf{e}}_{c,30}\rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{k,1}\rvert) \\ \vdots \\ \ln(\lvert\hat{\mathbf{e}}_{k,30}\rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{\theta,1}\rvert) \\ \\ \ln(\lvert\hat{\mathbf{e}}_{\theta,30}\rvert) \end{bmatrix}$
−5.34255	−5.33875	−11.8565
−6.15664	−6.15255	−12.3108
−5.41646	−5.41585	−13.8565
−5.64275	−5.64369	−18.1473
−5.9222	−5.92211	−16.0029
−5.87248	−5.86293	−12.0622
−5.20976	−5.20965	−13.8815
−6.26734	−6.27376	−13.3781
−6.11431	−6.11424	−13.5244
−5.43751	−5.43768	−14.3313
−6.84735	−6.83506	−13.4062
−4.96411	−4.96311	−15.7406
−6.74538	−6.74996	−13.0128
−5.51523	−5.51584	−14.6129
−7.01914	−7.01377	−13.0087
−4.98907	−4.9888	−12.8895
−5.54918	−5.5502	−14.2886
−5.78761	−5.7866	−13.6307
−5.10822	−5.11197	−16.4254
−5.91312	−5.91964	−15.971
−5.32484	−5.32492	−12.9666
−6.81071	−6.80294	−12.6755
−5.75943	−5.75928	−13.8696
−5.76735	−5.76907	−14.3413
−6.93921	−6.94492	−12.2327
−4.87233	−4.87293	−13.0072
−5.68297	−5.68316	−20.3557
−5.16688	−5.16342	−17.0775
−5.33829	−5.33817	−12.6149
−6.21494	−6.20121	−12.1051

Table 16: RBC Approximated Solution: Error Approximationsd=(2,2,2)

6 A Model with Occasionally Binding Constraints

Stochastic dynamic non linear economic models often embody occasionally binding constraints (OBC). Since (Christiano and Fisher, 2000) a host of authors have described a variety of approaches.[11] (Holden, 2015; Guerrieri and Iacoviello, 2015b; Benigno et al., 2009; Hintermaier and Koeniger, 2010b; Brumm and Grill, 2010; Nakov, 2008; Haefke, 1998; Nakata, 2012; Gordon, 2011; Billi, 2011; Hintermaier and Koeniger, 2010a; Guerrieri and Iacoviello, 2015a)

Consider adding a constraint to the simple RBC model:

$$I_t \geq \upsilon I_{ss}$$

$$\begin{gathered}
\max \left\{ u(c_t) + E_t \sum_{t=0}^{\infty} \delta^{t+1} u(c_{t+1}) \right\} \\
c_t + k_t = (1-d)k_{t-1} + \theta_t f(k_{t-1}) \\
\theta_t = \theta_{t-1}^{\rho} e^{\epsilon_t} \\
f(k_t) = k_t^{\alpha} \\
u(c) = \frac{c^{1-\eta} - 1}{1-\eta} \\
\mathbb{L} = \left\{ u(c_t) + E_t \sum_{t=0}^{\infty} \delta^{t} u(c_{t+1}) \right\} + \\
\sum_{t=0}^{\infty} \Big\{ \delta^t \lambda_t (c_t + k_t - ((1-d)k_{t-1} + \theta_t f(k_{t-1}))) + \\
\delta^t \mu_t ((k_t - (1-d)k_{t-1}) - \upsilon I_{ss}) \Big\}
\end{gathered}$$

so that we can impose

$$I_t = (k_t - (1-d)k_{t-1}) \geq \upsilon I_{ss}$$

The first order conditions become

$$\begin{gathered}
\frac{1}{c_t^{\eta}} + \lambda_t \\
\lambda_t - \delta \lambda_{t+1} \theta_{t+1} \alpha k_t^{(\alpha-1)} - \delta(1-d)\lambda_{t+1} + \mu_t - \delta(1-d)\mu_{t+1}
\end{gathered}$$

For example, the Euler equations for the neoclassical growth model can be written as

[11] The algorithms described in (Holden, 2015) and (Guerrieri and Iacoviello, 2015b) also exploit the use of "anticipated shocks", but do not use the comprehensive formula employed here.

$$if(\mu > 0 \wedge (k_t - (1-d)k_{t-1} - \upsilon I_{ss}) = 0)$$
$$\lambda_t - \frac{1}{c_t}$$
$$c_t + I_t - \theta_t k_{t-1}^{\alpha}$$
$$N_t - \lambda_t \theta_t$$
$$\theta_t - e^{(\rho \ln(\theta_{t-1}) + \epsilon)}$$
$$\lambda_t + \mu_t - (\alpha k_t^{(\alpha-1)} \delta N_{t+1} + \lambda_{t+1}\delta(1-d) + \mu_{t+1} + \delta(1-d)\mu_{t+1}$$
$$I_t - (K_t - (1-d)k_{t-1})$$
$$\mu_t(k_t - (1-d)k_{t-1} - \upsilon I_{ss})$$

$$if(\mu = 0 \wedge (k_t - (1-d)k_{t-1} - \upsilon I_{ss}) \geq 0)$$
$$\lambda_t - \frac{1}{c_t}$$
$$c_t + I_t - \theta_t k_{t-1}^{\alpha}$$
$$N_t - \lambda_t \theta_t$$
$$\theta_t - e^{(\rho \ln(\theta_{t-1}) + \epsilon)}$$
$$\lambda_t + \mu_t - (\alpha k_t^{(\alpha-1)} \delta N_{t+1} + \lambda_{t+1}\delta(1-d) + \mu_{t+1} + \delta(1-d)\mu_{t+1}$$
$$I_t - (K_t - (1-d)k_{t-1})$$
$$\mu_t(k_t - (1-d)k_{t-1} - \upsilon I_{ss})$$

Table 17 shows the evaluation points when the Smolyak polynomial degrees of approximation were (1,1,1). Table 18 shows the decision rule prescription for (c_t, k_t, θ_t) at each evaluation point. Table 19 shows the log of the absolute value of the estimated error in the decision rule at each evaluation points. Note that the formula provides accurate estimates of the error component by component.

Table 20 shows the evaluation points when the Smolyak polynomial degrees of approximation were (2,2,2). Table 21 shows the decision rule prescription for (c_t, k_t, θ_t) at each evaluation point. Table 22 shows the log of the absolute value of the estimated error in the decision rule at each evaluation points. Note that the formula provides accurate estimates of the error component by component.

Why not here?

$\begin{bmatrix} k_1 & \theta_1 & \epsilon_1 \\ & \vdots & \\ k_{30} & \theta_{30} & \epsilon_{30} \end{bmatrix}$		
3.26643	0.944454	−0.0261085
4.12098	1.06801	0.0270199
3.67835	0.940854	−0.00839901
3.92114	0.989356	0.0137378
3.69543	1.00026	−0.0216811
3.88415	1.05817	0.0314473
3.44151	0.931012	−0.00397164
4.26001	1.06084	0.0225926
3.78979	1.02922	−0.0128264
3.60107	0.971308	0.0048831
4.20171	1.02562	−0.0305358
3.41024	0.891085	0.00266941
3.96698	1.03809	−0.0327495
3.63406	1.00526	0.0203789
4.2347	1.05957	−0.0150401
3.41619	0.929745	0.0292336
3.4676	0.988852	−0.00618532
3.80052	1.02168	0.0115241
3.35789	0.89452	−0.0238948
4.15837	1.02748	0.0248063
3.41102	0.947657	−0.0106127
4.12137	1.0963	0.00709678
3.67874	0.96914	−0.0283222
3.97561	1.00824	0.0159515
4.02702	1.06734	−0.0194674
3.31666	0.918703	0.033661
3.9173	0.973011	−0.00175796
3.65197	0.928429	0.0170584
3.42219	0.938626	−0.0183606
4.13254	1.08727	0.0347678

Table 17: Occasionally Binding Constraints Model Evaluation Points d=(1,1,1)

$\begin{bmatrix} c_1 \\ \\ c_{30} \end{bmatrix}$	$\begin{bmatrix} I_1 \\ \vdots \\ I_{30} \end{bmatrix}$	$\begin{bmatrix} k_1 \\ \\ k_{30} \end{bmatrix}$
1.03662	0.379903	3.34624
1.36955	0.431143	4.13607
1.13338	0.361691	3.69353
1.25263	0.381718	3.92139
1.18795	0.376988	3.71505
1.32486	0.433045	3.92962
1.08991	0.364941	3.48842
1.37645	0.426962	4.25615
1.24202	0.39202	3.80933
1.17598	0.373255	3.63206
1.26718	0.39439	4.1772
1.03482	0.3675	3.46926
1.25075	0.392683	3.96566
1.22878	0.398246	3.68118
1.33116	0.409411	4.21636
1.11619	0.384274	3.48551
1.15026	0.38833	3.52625
1.26672	0.394799	3.82276
0.997682	0.362814	3.41768
1.33254	0.40944	4.15357
1.095	0.369696	3.46366
1.36855	0.443297	4.14433
1.14269	0.364517	3.69245
1.28393	0.389658	3.97475
1.30274	0.41229	4.03407
1.08632	0.391331	3.40604
1.21329	0.372403	3.91112
1.13942	0.372397	3.68273
1.07662	0.365006	3.47022
1.38964	0.453375	4.16566

Table 18: Occasionally Binding Constraints Values at Evaluation Points for Occasionally Binding Constraints d=(1,1,1)

$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{c,1}\rvert) \\ \\ \ln(\lvert\hat{\mathbf{e}}_{c,30}\rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{k,1}\rvert) \\ \vdots \\ \ln(\lvert\hat{\mathbf{e}}_{k,30}\rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}}_{\theta,1}\rvert) \\ \\ \ln(\lvert\hat{\mathbf{e}}_{\theta,30}\rvert) \end{bmatrix}$
−4.31395	−4.55496	−4.55496
−4.49983	−4.13333	−4.13333
−5.5352	−5.24857	−5.24857
−5.8198	−5.84969	−5.84969
−4.60287	−4.60328	−4.60328
−4.41983	−4.10467	−4.10467
−4.62802	−4.55181	−4.55181
−5.26804	−4.74828	−4.74828
−8.43803	−8.15898	−8.15898
−5.35434	−5.28501	−5.28501
−3.85154	−3.66516	−3.66516
−4.38957	−4.32099	−4.32099
−4.47738	−4.23689	−4.23689
−5.01928	−5.04091	−5.04091
−5.51293	−4.94452	−4.94452
−3.83164	−3.64992	−3.64992
−4.53368	−4.51412	−4.51412
−4.74133	−4.66482	−4.66482
−7.3604	−5.00693	−5.00693
−7.71361	−5.96299	−5.96299
−4.71182	−4.60582	−4.60582
−4.81987	−4.52925	−4.52925
−6.36037	−5.73385	−5.73385
−6.04304	−5.80405	−5.80405
−6.58347	−5.58301	−5.58301
−3.45988	−3.27868	−3.27868
−4.15596	−4.15415	−4.15415
−4.2487	−4.33841	−4.33841
−5.03715	−4.72138	−4.72138
−4.38481	−3.89053	−3.89053

Table 19: Occasionally Binding Constraints Error Approximations d=(1,1,1)

$\begin{bmatrix} k_1 & \theta_1 & \epsilon_1 \\ & \vdots & \\ k_{30} & \theta_{30} & \epsilon_{30} \end{bmatrix}$		
3.11653	0.930006	−0.0265567
4.04206	1.06199	0.0292736
3.58012	0.922498	−0.00794662
3.83937	0.975085	0.015316
3.58288	0.98926	−0.0219042
3.77881	1.05289	0.0339261
3.31687	0.9134	−0.0032941
4.20017	1.05275	0.0246211
3.68084	1.02108	−0.0125991
3.48492	0.957443	0.00601095
4.14443	1.01357	−0.0312093
3.29279	0.868696	0.00368469
3.87738	1.02958	−0.0335355
3.51259	0.995409	0.0222948
4.17209	1.05153	−0.0149254
3.28879	0.912185	0.0315998
3.33019	0.978321	−0.00562036
3.69499	1.0125	0.0129897
3.23305	0.873004	−0.0242305
4.09524	1.01604	0.0269473
3.27803	0.932401	−0.0102729
4.03468	1.09384	0.00833722
3.57274	0.954351	−0.028883
3.89532	0.995891	0.0176423
3.93671	1.06203	−0.0195779
3.18007	0.900584	0.0362524
3.83958	0.95671	−0.000967836
3.55394	0.908726	0.0188054
3.29307	0.922137	−0.0184148
4.04972	1.08358	0.0374155

Table 20: Occasionally Binding Constraints Model Evaluation Points d=(2,2,2)

$\begin{bmatrix} k_1 & \theta_1 & \epsilon_1 \\ & \vdots & \\ k_{30} & \theta_{30} & \epsilon_{30} \end{bmatrix}$		
0.996234	0.372233	3.17711
1.35273	0.449889	4.08774
1.08239	0.37197	3.59408
1.22794	0.381138	3.83657
1.15723	0.375819	3.60041
1.29529	0.457947	3.85887
1.03658	0.371604	3.35678
1.37221	0.431977	4.21213
1.21472	0.395466	3.70822
1.14148	0.37158	3.50801
1.24362	0.394261	4.12425
0.972304	0.376186	3.33969
1.22692	0.392432	3.88208
1.19298	0.407354	3.56868
1.31345	0.414672	4.16956
1.06882	0.383074	3.34298
1.1142	0.387566	3.38473
1.23929	0.401702	3.72719
0.926116	0.382809	3.29255
1.32171	0.410856	4.09657
1.04696	0.373008	3.32323
1.35156	0.46278	4.09399
1.09896	0.370843	3.58631
1.26258	0.39151	3.8973
1.28036	0.420156	3.9632
1.04154	0.382172	3.24423
1.18102	0.373737	3.82935
1.09669	0.372006	3.57055
1.02297	0.373053	3.33681
1.38105	0.472609	4.11735

Table 21: Occasionally Binding Constraints Values at Evaluation Points for Occasionally Binding Constraints d=(2,2,2)

$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}_{c,1}}\rvert) \\ \\ \ln(\lvert\hat{\mathbf{e}_{c,30}}\rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}_{k,1}}\rvert) \\ \vdots \\ \ln(\lvert\hat{\mathbf{e}_{k,30}}\rvert) \end{bmatrix}$	$\begin{bmatrix} \ln(\lvert\hat{\mathbf{e}_{\theta,1}}\rvert) \\ \\ \ln(\lvert\hat{\mathbf{e}_{\theta,30}}\rvert) \end{bmatrix}$
−4.3713	−4.37518	−4.37518
−4.80064	−4.79951	−4.79951
−4.51635	−4.51745	−4.51745
−4.97684	−4.97675	−4.97675
−4.76238	−4.76248	−4.76248
−5.20213	−5.19772	−5.19772
−4.42504	−4.42526	−4.42526
−4.74432	−4.74585	−4.74585
−5.81449	−5.81175	−5.81175
−5.44103	−5.4409	−5.4409
−3.41118	−3.41245	−3.41245
−2.35772	−2.35772	−2.35772
−4.10566	−4.10512	−4.10512
−7.55599	−7.55774	−7.55774
−4.76462	−4.76603	−4.76603
−3.88786	−3.88797	−3.88797
−4.30233	−4.30326	−4.30326
−5.00949	−5.00948	−5.00948
−2.04364	−2.04336	−2.04336
−5.59945	−5.60358	−5.60358
−5.12234	−5.12392	−5.12392
−5.73503	−5.7328	−5.7328
−4.80694	−4.80739	−4.80739
−7.06764	−7.06949	−7.06949
−5.20027	−5.196	−5.196
−3.4838	−3.48367	−3.48367
−3.61222	−3.61225	−3.61225
−4.37205	−4.3713	−4.3713
−4.80664	−4.80713	−4.80713
−4.63986	−4.63587	−4.63587

Table 22: Occasionally Binding Constraints Error Approximations d=(2,2,2)

7 Conclusions

This paper introduces a new series representation for bounded time series and shows how to develop series for representing bounded time invariant maps. The series representation plays a strategic role in developing a formula for approximating the errors in dynamic model solution decision rules. The paper also shows how to augment the usual "Euler Equation" model solution methods with constraints reflecting how the updated conditional expectation paths relate to the time t solution values thereby enhancing solution algorithm reliability and accuracy. The prototype was written in Mathematica and is available on gitHub at https://github.com/es335mathwiz/AMASeriesRepresentation.

References

Gary Anderson. A reliable and computationally efficient algorithm for imposing the saddle point proprerty in dynamic models. *Journal of Economic Dynamics and Control*, 34:472–489, June 2010. URL http://www.sciencedirect.com/science/article/pii/S0165188909001924.

Gianluca Benigno, Huigang Chen, Christopher Otrok, Alessandro Rebucci, and Eric R. Young. Optimal policy with occasionally binding credit constraints. First Draft Nov 2007, April 2009.

Roberto M. Billi. Optimal inflation for the us economy. *American Economic Journal: Macroeconomics*, 3(3):29–52, July 2011. URL http://ideas.repec.org/a/aea/aejmac/v3y2011i3p29-52.html.

Andrew Binning and Junior Maih. Modelling Occasionally Binding Constraints Using Regime-Switching. Working Papers No 9/2017, Centre for Applied Macro- and Petroleum economics (CAMP), BI Norwegian Business School, December 2017. URL https://ideas.repec.org/p/bny/wpaper/0058.html.

Olivier Jean Blanchard and Charles M Kahn. The solution of linear difference models under rational expectations. *Econometrica*, 48(5):1305–11, July 1980. URL http://ideas.repec.org/a/ecm/emetrp/v48y1980i5p1305-11.html.

Johannes Brumm and Michael Grill. Computing equilibria in dynamic models with occasionally binding constraints. Technical Report 95, University of Mannheim Center for Doctoral Studies in Economics, July 2010.

Lawrence J. Christiano and Jonas D. M. Fisher. Algorithms for solving dynamic models with occasionally binding constraints. *Journal of Economic Dynamics and Control*, 24(8):1179–1232, July 2000. URL http://www.sciencedirect.com/science/article/B6V85-408BVCF-2/1/217643ed2bbecee4fa5a1ec60ed9407e.

Ulrich Doraszelskiy and Kenneth L. Judd. Avoiding the curse of dimensionality in dynamic stochastic games. Harvard University and Hoover Institution and NBER, November 2004. URL file:///P:/compmpe.pdf.

Jess Gaspar and Kenneth L. Judd. Solving large scale rational expectations models. Technical Report 0207, National Bureau of Economic Research, Inc, February 1997. URL file:///P:/t0207.pdf. available at http://ideas.repec.org/p/nbr/nberte/0207.html.

Grey Gordon. Computing dynamic heterogeneous-agent economies: Tracking the distribution. PIER Working Paper Archive 11-018, Penn Institute for Economic Research, Department of Economics, University of Pennsylvania, June 2011. URL http://ideas.repec.org/p/pen/papers/11-018.html.

Luca Guerrieri and Matteo Iacoviello. OccBin: A toolkit for solving dynamic models with occasionally binding constraints easily. *Journal of Monetary Economics*, 70:22–38,

2015a. ISSN 03043932. doi: 10.1016/j.jmoneco.2014.08.005. URL http://linkinghub.elsevier.com/retrieve/pii/S0304393214001238.

Luca Guerrieri and Matteo Iacoviello. Occbin: A toolkit for solving dynamic models with occasionally binding constraints easily. *Journal of Monetary Economics*, 70:22–38, 2015b.

Christian Haefke. Projections parameterized expectations algorithms (matlab). QM&RBC Codes, Quantitative Macroeconomics & Real Business Cycles, November 1998. URL http://ideas.repec.org/c/dge/qmrbcd/69.html.

Thomas Hintermaier and Winfried Koeniger. The method of endogenous gridpoints with occasionally binding constraints among endogenous variables. *Journal of Economic Dynamics and Control*, 34(10):2074–2088, 2010a. ISSN 01651889. doi: 10.1016/j.jedc.2010.05.002. URL http://dx.doi.org/10.1016/j.jedc.2010.05.002.

Thomas Hintermaier and Winfried Koeniger. The method of endogenous gridpoints with occasionally binding constraints among endogenous variables. *Journal of Economic Dynamics and Control*, 34(10):2074–2088, October 2010b. URL http://ideas.repec.org/a/eee/dyncon/v34y2010i10p2074-2088.html.

Thomas Holden. Existence, uniqueness and computation of solutions to dsge models with occasionally binding constraints. University Surrey, 2015.

Kenneth Judd. Projection methods for solving aggregate growth models. *Journal of Economic Theory*, 58:410–452, 1992.

Kenneth Judd, Lilia Maliar, and Serguei Maliar. Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models. Working Papers. Serie AD 2011-15, Instituto Valenciano de Investigaciones EconÃşmicas, S.A. (Ivie), July 2011. URL https://ideas.repec.org/p/ivi/wpasad/2011-15.html.

Kenneth L Judd, Lilia Maliar, Serguei Maliar, and Rafael Valero. Smolyak Method for Solving Dynamic Economic Models : Lagrange Interpolation , Anisotropic Grid and Adaptive Domain. unpublished, 2013.

Kenneth L. Judd, Lilia Maliar, Serguei Maliar, and Rafael Valero. Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain. *Journal of Economic Dynamics and Control*, 44:92–123, July 2014. ISSN 01651889. doi: 10.1016/j.jedc.2014.03.003. URL http://linkinghub.elsevier.com/retrieve/pii/S0165188914000621.

Kenneth L. Judd, Lilia Maliar, and Serguei Maliar. Lower bounds on approximation errors to numerical solutions of dynamic economic models. *Econometrica*, 85(3):991–1012, 5 2017a. ISSN 1468-0262. doi: 10.3982/ECTA12791. URL http:https://doi.org/10.3982/ECTA12791.

Kenneth L. Judd, Lilia Maliar, Serguei Maliar, and Inna Tsener. How to solve dynamic stochastic models computing expectations just once. *Quantitative Economics*, 8(3):851–893, November 2017b. URL https://ideas.repec.org/a/wly/quante/v8y2017i3p851-893.html.

Gary Koop, M. Hashem Pesaran, and Simon M. Potter. Impulse response analysis in nonlinear multivariate models. *Journal of Econometrics*, 74(1):119–147, September 1996. URL http://www.sciencedirect.com/science/article/B6VC0-3XDS2R2-6/2/f8b1713c81765453e1a5e9a52593b52e.

Martin Lettau. Inspecting the mechanism: Closed-form solutions for asset prices in real business cycle models. *Economic Journal*, 113(489):550–575, July 2003.

Lilia Maliar and Serguei Maliar. Parametrized Expectations Algorithm And The Moving Bounds. Working Papers. Serie AD 2001-23, Instituto Valenciano de Investigaciones EconÃşmicas, S.A. (Ivie), August 2001. URL https://ideas.repec.org/p/ivi/wpasad/2001-23.html.

Lilia Maliar and Serguei Maliar. Solving the neoclassical growth model with quasi-geometric discounting: A grid-based euler-equation method. *Computational Economics*, 26:163–172, 2005. ISSN 09277099. doi: 10.1007/s10614-005-1732-y.

Albert Marcet and Guido Lorenzoni. Parameterized expectations approach: Some practical issues. In Ramon Marimon and Andrew Scott, editors, *Computational Methods for the Study of Dynamic Economies*, chapter 7, pages 143–171. Oxford University Press, New York, 1999.

Taisuke Nakata. Optimal fiscal and monetary policy with occasionally binding zero bound constraints. New York University, January 2012.

Anton Nakov. Optimal and simple monetary policy rules with zero floor on the nominal interest rate. *International Journal of Central Banking*, 4(2):73–127, June 2008. URL http://ideas.repec.org/a/ijc/ijcjou/y2008q2a3.html.

A Peralta-Alva and Manuel Santos. *Schmedders, K and Judd, K*, volume 3, chapter 9, pages 517–556. Elsevier Science, 2014.

Simon M. Potter. Nonlinear impulse response functions. *Journal of Economic Dynamics and Control*, 24(10):1425–1446, September 2000. URL http://www.sciencedirect.com/science/article/B6V85-40T9HN0-3/1/3f27e3e4d4b890df59d4f83850c1c3d1.

By Manuel S Santos and Adrian Peralta-alva. Accuracy of simulations for stochastic dynamic models. *Econometrica*, 73(6):1939–1976, 11 2005. ISSN 1468-0262. doi: 10.1111/j.1468-0262.2005.00642.x. URL http:https://doi.org/10.1111/j.1468-0262.2005.00642.x.

Manuel Santos. Accuracy of numerical solutions using the euler equation residuals. *Econometrica*, 68(6):1377–1402, 2000. URL https://EconPapers.repec.org/RePEc:ecm:emetrp:v:68:y:2000:i:6:p:1377-1402.

A Error Approximation Formula Proof

We consider time invariant maps such as often arise from solving an optimization problem codified in a systems of equations. In what follows, we construct an error approximation for proposed model solutions. Consider a bounded region, $\mathcal{A}$, where all iterated conditional expectations paths remain bounded. Given an exact solution $x_t^\star = g^\star(x_{t-1}, \epsilon_t)$ define the conditional expectations function,

$$G^\star(x) \equiv \mathcal{E}\left[g^\star(x, \epsilon)\right]$$

then with

$$E_t x_{t+1}^\star = G^\star(g^\star(x_{t-1}, \epsilon_t))$$

we have

$$\mathbb{M}(x_{t-1}^\star, x_t^\star, E_t x_{t+1}^\star, \epsilon_t) = 0 \ \forall (x_{t-1}, \epsilon_t)$$

In words, the exact solution exactly satisfies the model equations. Using $G^\star$ and $\mathcal{L}$, construct the family of trajectories and corresponding $z_t^\star(x_{t-1}, \epsilon)$

$$x_t^\star(x_{t-1}, \epsilon_t) \in R^L \ \ \|x_t^\star(x_{t-1}, \epsilon_t)\|_2 \leq \bar{\mathcal{X}} \ \forall t \ > 0$$

$$\begin{aligned} z_t^\star(x_{t-1}, \epsilon_t) \equiv & H_{-1} x_{t-1}^\star(x_{t-1}, \epsilon_t) + \\ & H_0 x_t^\star(x_{t-1}, \epsilon_t) + \\ & H_1 x_{t+1}^\star(x_{t-1}, \epsilon_t). \end{aligned}$$

Consequently, the exact solution has a representation given by

$$\begin{aligned} x_t^\star(x_{t-1}, \epsilon) = & Bx_{t-1} + \phi\psi_\epsilon \epsilon + (I - F)^{-1}\phi\psi_c + \\ & \sum_{\nu=0}^{\infty} F^s \phi z_{t+\nu}^\star(x_{t-1}, \epsilon) \end{aligned}$$

and

$$\mathcal{E}\left[x_{t+1}^\star(x_{t-1}, \epsilon)\right] = Bx_{t+k}^\star + \sum_{\nu=0}^{\infty} F^\nu \phi \mathcal{E}\left[z_{t+1+\nu}^\star(x_{t-1}, \epsilon)\right] + (I - F)^{-1}\phi\psi_c$$

with

$$\mathbb{M}(x_{t-1}, x_t^\star, E_t x_{t+1}^\star, \epsilon_t) = 0 \ \forall (x_{t-1}, \epsilon_t)$$

Now consider a proposed solution for the model, $x_t^p = g^p(x_{t-1}, \epsilon_t)$ define $G^p(x) \equiv \mathcal{E}\left[g^p(x, \epsilon)\right]$ so that

$$\begin{aligned} E_t x_{t+1} &= G^p(g^p(x_{t-1}, \epsilon_t)) \\ \mathbf{e}_t^p(x_{t-1}, \epsilon) &\equiv \mathbb{M}(x_{t-1}, x_t^p, E_t x_{t+1}^p, \epsilon_t) \end{aligned}$$

By construction, the conditional expectations paths will all be bounded in the region $\mathcal{A}^p$. Using G^p and $\mathcal{L}$ construct the family of trajectories and corresponding $z_t^p(x_{t-1},\epsilon)$[12]

$$x_t^p(x_{t-1},\epsilon_t) \in R^L \;\; \|x_t^p(x_{t-1},\epsilon_t)\|_2 \leq \bar{\mathcal{X}} \; \forall t \; > 0$$

$$\begin{aligned} z_t^p(x_{t-1},\epsilon_t) \equiv & H_{-1}x_{t-1}^p(x_{t-1},\epsilon_t) + \\ & H_0 x_t^p(x_{t-1},\epsilon_t) + \\ & H_1 x_{t+1}^p(x_{t-1},\epsilon_t). \end{aligned}$$

The proposed solution has a representation given by

$$\begin{aligned} x_t^p(x_{t-1},\epsilon) = Bx_{t-1} + \phi\psi_\epsilon\epsilon + (I-F)^{-1}\phi\psi_c + \\ \sum_{\nu=0}^{\infty} F^s \phi z_{t+\nu}^p(x_{t-1},\epsilon) \end{aligned}$$

and

$$\mathcal{E}\left[x_{t+1}^p(x_{t-1},\epsilon)\right] = Bx_{t+k}^p + \sum_{\nu=0}^{\infty} F^\nu \phi z_{t+1+\nu}^p(x_{t-1},\epsilon) + (I-F)^{-1}\phi\psi_c$$

with

$$\mathbf{e}_t^p(x_{t-1},\epsilon) \equiv \mathbb{M}(x_{t-1}, x_t^p, E_t x_{t+1}^p, \epsilon_t)$$

$$\begin{aligned} x_t^\star(x_{t-1},\epsilon) - x_t^p(x_{t-1},\epsilon) &= \sum_{\nu=0}^{\infty} F^s \phi (z_{t+\nu}^\star(x_{t-1},\epsilon) - z_{t+\nu}^p(x_{t-1},\epsilon)) \\ \Delta z_{t+\nu}^p(x_{t-1},\epsilon_t) &\equiv (z_{t+\nu}^\star(x_{t-1},\epsilon) - z_{t+\nu}^p(x_{t-1},\epsilon)) \\ x_t^\star(x_{t-1},\epsilon) - x_t^p(x_{t-1},\epsilon) &= \sum_{\nu=0}^{\infty} F^s \phi \Delta z_{t+\nu}^p(x_{t-1},\epsilon_t) \\ \|x_t^\star(x_{t-1},\epsilon) - x_t^p(x_{t-1},\epsilon)\|_\infty &\leq \sum_{\nu=0}^{\infty} F^s \phi \, \|\Delta z_{t+\nu}^p(x_{t-1},\epsilon_t)\|_\infty \end{aligned}$$

By bounding the largest deviation in the paths for the Δz_t^p we can bound the largest difference in x_t.[13] The exact solution satisfies the model equations exactly. The error associated with the proposed solution leads to a conservative bound on the largest change in z needed to match the exact solution.

[12]The algorithm presented below for finding solutions provides a mechanism for generating such proposed solutions.

[13]Since the future values are probability weighted averages of the Δz_t^p values for the given initial conditions and the condition expectations paths remain in the region $\mathcal{A}^p$, the largest error for Δz_{t+k}^p are pessimistic bounds for the errors from the conditional expectations path.

$$\Delta z_t \leq \max_{\{x_-,\epsilon\}} \|\phi \mathbb{M}(x_-, g^p(x_-,\epsilon), G^p(g^p(x_-,\epsilon)),\epsilon)\|_2$$
$$\|x_t^\star(x_{t-1},\epsilon) - x_t^p(x_{t-1},\epsilon)\|_\infty \leq \max_{\{x_-,\epsilon\}} \left\|(I-F)^{-1}\phi \mathbb{M}(x_-, g^p(x_-,\epsilon), G^p(g^p(x_-,\epsilon)),\epsilon)\right\|_2$$

$$z_t^\star = H_- x^\star{}_{t-1} + H_0 x^\star{}_t + H_+ x^\star{}_{t+1}$$
$$z_t^p = H_- x^p{}_{t-1} + H_0 x^p{}_t + H_+ x^p{}_{t+1}$$
$$0 = \mathbb{M}(x_{t-1}^\star, x_t^\star, x_{t+1}^\star, \epsilon_t)$$
$$\mathbf{e}_t^p(x_{t-1},\epsilon) = \mathbb{M}(x_{t-1}^p, x_t^p, x_{t+1}^p, \epsilon_t)$$
$$\max_{x_{t-1},\epsilon_t} \left(\|\phi \Delta z_t\|_2\right)^2 = \max_{x_{t-1},\epsilon_t} \left(\phi(H_0(x_t^p - x_t^\star) + H_+(x_{t+1}^p - x_{t+1}^\star))\right)^2$$

with

$$\mathbb{M}(x_{t-1}^\star, x_t^\star, x_{t+1}^\star, \epsilon_t) = 0$$
$$\Delta \mathbf{e}_t^p(x_{t-1},\epsilon) \approx \frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t}(x_t^\star - x_t^p) + \frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_{t+1}}(x_{t+1}^\star - x_{t+1}^p)$$

when $\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t}$ is non-singular, we can write

$$(x_t^\star - x_t^p) \approx \left(\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t}\right)^{-1} \left(\Delta \mathbf{e}_t^p(x_{t-1},\epsilon) - \frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_{t+1}}(x_{t+1}^\star - x_{t+1}^p)\right)$$

collecting results

$$\left(\|\phi \Delta z_t\|_2\right)^2 =$$
$$\left(\phi(H_0 \left(\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t}\right)^{-1}\right.$$
$$\left.\left(\Delta \mathbf{e}_t^p(x_{t-1},\epsilon) - \frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_{t+1}}(x_{t+1}^\star - x_{t+1}^p)\right) + H_+(x_{t+1}^p - x_{t+1}^\star))\right)^2 =$$
$$\left(\phi(H_0 \left(\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t}\right)^{-1} (\Delta \mathbf{e}_t^p(x_{t-1},\epsilon)) - \right.$$
$$\left.\left(H_0 \left(\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t}\right)^{-1} \frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_{t+1}}(x_{t+1}^\star - x_{t+1}^p)\right) + H_+(x_{t+1}^p - x_{t+1}^\star))\right)^2 =$$
$$\left(\phi(H_0 \left(\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t}\right)^{-1} (\Delta \mathbf{e}_t^p(x_{t-1},\epsilon)) - \right.$$
$$\left.\left(H_0 \left(\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t}\right)^{-1} \frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_{t+1}} - H_+\right)(x_{t+1}^\star - x_{t+1}^p))\right)^2 =$$

approximating the derivatives using the H matrices

$$\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_t} \approx H_0$$
$$\frac{\partial \mathbb{M}(x_{t-1}, x_t, x_{t+1}, \epsilon_t)}{\partial x_{t+1}} \approx H_+$$

produces

$$\max_{x_{t-1}, \epsilon_t} (\phi \Delta \mathbf{e}_t^p(x_{t-1}, \epsilon))^2$$

B A Regime Switching Example

To apply the series formula, one must construct conditional expectations paths from each initial $x(x_{t-1}, \epsilon_t)$. Consider the case when the transition probabilities p_{ij} are constant and do not depend on x_{t-1}, ϵ_t, x_t.

$$E_t[x_{t+1}] = \sum_{j=1}^{N} p_{ij} E_t(x_{t+1}(x_t)|s_{t+1} = j)$$
$$E_t[x_{t+2}] = \sum_{j=1}^{N} p_{ij} E_t(x_{t+2}(x_{t+1})|s_{t+2} = j)$$

If we know the current state we can use the known conditional expectation function for the state:

$$\begin{bmatrix} E_t(x_{t+1}(x_t)|s_t = 1) \\ \vdots \\ E_t(x_{t+1}(x_t)|s_t = N) \end{bmatrix}$$

For the next state we can compute expectations if the errors are independent

$$\begin{bmatrix} E_t(x_{t+2}(x_t)|s_t = 1) \\ \vdots \\ E_t(x_{t+2}(x_t)|s_t = N) \end{bmatrix} = (P \otimes I) \begin{bmatrix} E_t(x_{t+2}(x_{t+1})|s_{t+1} = 1) \\ \vdots \\ E_t(x_{t+2}(x_{t+1})|s_{t+1} = N) \end{bmatrix}$$

Consider two states $s_t \in 0, 1$ where the depreciation rates are different: $d_0 > d_1$

$$Prob(s_t = j|s_{t-1} = i) = p_{ij}$$

$$if(s_t = 0 \wedge \mu > 0 \wedge (k_t - (1 - d_0)k_{t-1} - \upsilon I_{ss}) = 0)$$
$$\lambda_t - \frac{1}{c_t}$$
$$c_t + k_t - \theta_t k_{t-1}^{\alpha}$$
$$N_t - \lambda_t \theta_t$$
$$\theta_t - e^{(\rho \ln(\theta_{t-1}) + \epsilon)}$$
$$\lambda_t + \mu_t - (\alpha k_t^{(\alpha-1)} \delta N_{t+1} + \lambda_{t+1}\delta(1 - d_0) + \mu_{t+1} + \delta(1 - d_0)$$
$$I_t - (K_t - (1 - d_0)k_{t-1})$$
$$\mu_t(k_t - (1 - d_0)k_{t-1} - \upsilon I_{ss})$$

$$if(s_t = 0 \wedge \mu = 0 \wedge (k_t - (1 - d_0)k_{t-1} - \upsilon I_{ss}) \geq 0)$$
$$\lambda_t - \frac{1}{c_t}$$
$$c_t + k_t - \theta_t k_{t-1}^{\alpha}$$
$$N_t - \lambda_t \theta_t$$
$$\theta_t - e^{(\rho \ln(\theta_{t-1}) + \epsilon)}$$
$$\lambda_t + \mu_t - (\alpha k_t^{(\alpha-1)} \delta N_{t+1} + \lambda_{t+1}\delta(1 - d_0) + \mu_{t+1} + \delta(1 - d_0)$$
$$I_t - (K_t - (1 - d_0)k_{t-1})$$
$$\mu_t(k_t - (1 - d_0)k_{t-1} - \upsilon I_{ss})$$

$$if(s_t = 1 \wedge \mu > 0 \wedge (k_t - (1 - d_1)k_{t-1} - \upsilon I_{ss}) = 0)$$
$$\lambda_t - \frac{1}{c_t}$$
$$c_t + k_t - \theta_t k_{t-1}^{\alpha}$$
$$N_t - \lambda_t \theta_t$$
$$\theta_t - e^{(\rho \ln(\theta_{t-1}) + \epsilon)}$$
$$\lambda_t + \mu_t - (\alpha k_t^{(\alpha-1)} \delta N_{t+1} + \lambda_{t+1}\delta(1 - d_1) + \mu_{t+1} + \delta(1 - d_1)$$
$$I_t - (K_t - (1 - d_1)k_{t-1})$$
$$\mu_t(k_t - (1 - d_1)k_{t-1} - \upsilon I_{ss})$$

$$if(s_t = 1 \wedge \mu = 0 \wedge (k_t - (1 - d_1)k_{t-1} - \upsilon I_{ss}) \geq 0)$$

$$\lambda_t - \frac{1}{c_t}$$

$$c_t + k_t - \theta_t k_{t-1}^{\alpha}$$

$$N_t - \lambda_t \theta_t$$

$$\theta_t - e^{(\rho \ln(\theta_{t-1}) + \epsilon)}$$

$$\lambda_t + \mu_t - (\alpha k_t^{(\alpha-1)} \delta N_{t+1} + \lambda_{t+1}\delta(1 - d_1) + \mu_{t+1} + \delta(1 - d_1)$$

$$I_t - (K_t - (1 - d_1)k_{t-1})$$

$$\mu_t(k_t - (1 - d_1)k_{t-1} - \upsilon I_{ss})$$

C Algorithm Pseudo-code

$\mathcal{L} \equiv \{H, \psi_\epsilon, \psi_c; B, \phi, F\}$ The linear reference model

$\mathbf{x}^p(x_{t-1}, \epsilon_t)$ The proposed decision rule x_t components

$\mathbf{z}^p(x_{t-1}, \epsilon_t)$ The proposed decision rule z_t components

$\mathbf{X}^p(x_{t-1})$ The proposed decision rule conditional expectations x_t components

$\mathbf{Z}^p(x_{t-1})$ The proposed decision rule their conditional expectations z_t components

κ One less than the number of terms in series representation($\kappa >= 0$)

$\mathbb{S}$ Smolyak an-isotropic grid polynomials $(p_1(x, \epsilon), \ldots, p_N(x, \epsilon))$ and their conditional expectations $(P_1(x), \ldots, P_N(x))$

$\mathbb{T}$ Model evaluation points Neidereiter sequence in the model variable ergodic region.

γ $\{\mathbf{x}(x_{t-1}, \epsilon_t), \mathbf{z}(x_{t-1}, \epsilon_t)\}$ collects the decision rule components

$\mathbb{G}$ $\{\mathbf{x}(x_{t-1}, \epsilon_t), \mathbf{z}(x_{t-1}, \epsilon_t)\}$ collects the decision rule conditional expectations components

$\mathbb{H}$ $\{\gamma, \mathbb{G}\}$ collects decision rule information

$\{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}$ The equation system $\mathcal{R}^{3N_x+N_\epsilon+N_x} \to \mathcal{R}^{N_x}$

input : $(\mathcal{L}, \mathbb{H}^0, \kappa, \{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}, \mathbb{S}, \mathbb{T})$

1 /* Compute a sequence of decision rule and conditional expectation rule function terminating when the evaluation of the functions at the tests points no longer changes much. Norm of the difference controlled by default (‘‘normConvTol’’->10^{-10}) */

2 **NestWhile**
(**Function**$(v, doInterp(\mathcal{L}, v, \kappa, \{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}, \mathbb{S})), \mathbb{H}^0, notConvergedAt($

output: $\{\mathbb{H}^0, \mathbb{H}^1, \ldots, \mathbb{H}^c\}$

Algorithm 1: *nestInterp*

input : $(\mathcal{L}, \mathbb{H}^i, \kappa, \{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}, \mathbb{S})$

1 /* Symbolically generates a collection of function triples and an outer model solution selection function. Each of triples returns the boolean gate values and a unique value for x_t for any given value of (x_{t-1}, ϵ_t) one of the model equations and subsequently uses these functions to construct interpolating functions approximating the decision rules. */

2 $theFindRootFuncs =$
$genFindRootFuncs(\mathcal{L}, \mathbb{H}, \kappa, \{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\})$

3 $\mathbb{H}^{i+1} = makeInterpFuncs(theFindRootFuncs, \mathbb{S})$

output: $\mathbb{H}^{i+1}$

Algorithm 2: *doInterp*

input : $(\mathcal{L}, \mathbb{H}, \kappa, \{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}, \mathbb{S})$

1 /* Applies genFindRootWorker to each model component. Symbolically generates a collection of function triples and an outer model solution selection function. Each of the triples returns the Boolean gate values and a unique value for x_t for any given value of (x_{t-1}, ϵ_t) for one from the set of model equations. It subsequently uses these functions to construct interpolating functions approximating the decision rules. Each of the function constructions can be done in **parallel**. */

2 $theFuncOptionsTriples =$
Map(**Function**$(v, \{v[1], genFindRootWorker(\mathcal{L}, \mathbb{H}, \kappa, v[2]), v[3]), \{\mathbb{C}_1, \ldots, \mathbb{C}_M\})$

output: $\{theFuncOptionsTriples, \mathbf{d}\}$

Algorithm 3: *genFindRootFuncs*

input : $(\mathcal{L}, \mathbb{G}, \kappa, \mathbb{M})$

1 /* **Symbolically** constructs functions that return x_t for a given (x_{t-1}, ϵ_t) */

2 $\mathcal{Z} = \{Z_{t+1}, \ldots, Z_{t+k-1}\} = genZsForFindRoot(\mathcal{L}, x_t^p, \mathbb{G})$

3 $modEqnArgsFunc = genLilXkZkFunc(\mathcal{L}, \mathcal{Z})$

4 $\mathcal{X} = \{x_{t-1}, x_t, E_t(x_{t+1}), \epsilon_t\} = modEqnArgsFunc(x_{t-1}, \epsilon_t, z_t)$

5 $funcOfXtZt = FindRoot((x_t^p, z_t), \{\mathbb{M}(\mathcal{X}), x_t - x_t^p\})$

6 $funcOfXtm1Eps = Function((x_{t-1}, \epsilon_t), funcOfXtZt)$

output: $funcOfXtm1Eps$

Algorithm 4: *genFindRootWorker*

input : $(x_t^p, \mathcal{L}, \mathbb{G}, \kappa, \mathbb{M})$

/* Symbolically compute the $\kappa - 1$ future conditional Z's vectors needed for the series formula.(empty list for $\kappa = 1$ */

$X_t = x_t^p$ **for** $i = 1, i < \kappa - 1, i++$ **do**

  $\{X_{t+1}, Z_{t+i}\} = \mathbb{G}(Z_{t+i-1})$

end

$= \{(X_{t+i}, Z_{t+1}), \ldots, (X_{t+\kappa-1} Z_{t+\kappa-1})\} = genZsForFindRoot(\mathcal{L}, x_t^p, \mathbb{G})$

output: $\{Z_{t+1}, \ldots, Z_{t+k-1}\}$

Algorithm 5: $genZsForFindRoot$

input : $(\mathcal{L} = \{H, \psi_\epsilon, \psi_c; B, \phi, F\})$

/* Create functions that use the solution from the linear reference model for an initial proposed decision rule function and decision rule conditional expectation function. */

$\gamma(x, \epsilon) = \begin{bmatrix} Bx + (I - F)^{-1}\phi\psi_c + \psi_\epsilon\epsilon_t \\ 0_{N_x} \end{bmatrix}$

$\mathbb{G}(x, \epsilon) = \begin{bmatrix} Bx + (I - F)^{-1}\phi\psi_c + \psi_\epsilon \\ 0_{N_x} \end{bmatrix}$

$\mathbb{H} = \{\gamma, \mathbb{G}\}$

output: $\{\mathbb{H}\}$

Algorithm 6: $genBothX0Z0Funcs$

input : $(\mathcal{L}, \{Z_{t+1}, \ldots, Z_{t+k-1}\})$

/* Symbolically creates a function that uses an initial Z_t path to generate a function of $(x_{t-1}, \epsilon_t, z_t)$ providing (potentially symbolic) inputs $(x_{t-1}, x_t, E_t(x_{t+1}), \epsilon_t)$,for model system equations $\mathbb{M}$ */

$fCon = fSumC(\phi, F, \psi_z, \{Z_{t+1}, \ldots, Z_{t+k-1}\})$

$xtVals = genXtOfXtm1(\mathcal{L}, x_{t-1}, \epsilon_t, z_t, fCon)$

$xtp1Vals = genXtp1OfXt(\mathcal{L}, xtVals, fCon)$

$fullVec = \{xtm1Vars, xtVals, xtp1Vals, epsVars\}$

output: $fullVec$

Algorithm 7: $genLilXkZkFunc$

input : $(\mathcal{L}, x_{t-1}, \epsilon_t, z_t, fCon)$

/* Symbolically creates a function that uses an initial Z_t path to generate a function of $(x_{t-1}, \epsilon_t, z_t)$ providing (potentially symbolically) the x_t component of inputs $(x_{t-1}, x_t, E_t(x_{t+1}), \epsilon_t)$,for model system equations $\mathbb{M}$ */

$x_t = Bx_{t-1} + (I - F)^{-1}\phi\psi_c + \phi.\psi_\epsilon.\epsilon_t + \phi\psi_z z_t + FfCon$

output: x_t

Algorithm 8: $genXtOfXtm1$

input : $(\mathcal{L}, x_{t-1}, fCon)$

/* Symbolically creates a function that uses an initial Z_t path to generate a function of $(x_{t-1}, \epsilon_t, z_t)$ providing (potentially symbolically)the x_{t+1} component of inputs $(x_{t-1}, x_t, E_t(x_{t+1}), \epsilon_t)$,for model system equations $\mathbb{M}$ */

$x_t = Bx_{t-1} + (I - F)^{-1}\phi\psi_c + \phi.\psi_\epsilon.\epsilon_t + \phi\psi_z z_t + fCon$

output: x_t

Algorithm 9: $genXtp1OfXt$

input : $(\mathcal{L}, \{Z_{t+1}, \ldots, Z_{t+k-1}\})$

/* Numerically sums the F contribution for the series. */

$S = \sum F^{\nu-1}\phi Z_{t+\nu}$

output: S

Algorithm 10: $fSumC$

input : $(\{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}, \mathbb{S})$

/* Solves model equations at collocation points producing interpolation data and subsequently returns the interpolating functions for the decision rule and decision rule conditional expectation. This routine also optionally replaces the approximations generated for all ‘‘backward looking’’ equations with user provided pre-computed functions. */

$interpData = genInterpData(\{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}, \mathbb{S})$

$\{\gamma, \mathbb{G}\} = interpDataToFunc(interData, \mathbb{S})$

output: $\{\gamma, \mathbb{G}\}$

Algorithm 11: $makeInterpFuncs$

input : $(\{\{\mathbb{C}_1, \ldots, \mathbb{C}_M\}, \mathbf{d}(x_{t-1}, \epsilon, x_t, \mathcal{E}[x_{t+1}])|(\mathbf{b}_\cdot, \mathbf{c}_\cdot)\}, \mathbb{S})$

/* Solves model equations at collocation points producing interpolation data and subsequently returns the interpolating functions for the decision rule and decision rule conditional expectation. This routine also optionally replaces the approximations generated for all ‘‘backward looking’’ equations with user provided pre-computed functions. */

output: $\{\gamma, \mathbb{G}\}$

Algorithm 12: $genInterpData$

input : $(\mathbb{C}_,(x_{t-1}, \epsilon_t))$

/* Evaluate the model equations obeying pre and post conditions */

output: x_t or **failed**

Algorithm 13: $evaluateTriple$

input : $(V, \mathbb{S})$

```
1 /* Computes a vector of Smolyak interpolating polynomials corresponding
     to the matrix of function evaluations.  Each row corresponds to a
     vector of values at a collocation point.                          */
```

output: $\{\gamma, \mathbb{G}\}$

Algorithm 14: *interpDataToFunc*

input : $(approxLevels, means, stdDev, minZs, maxZs, vvMat, distributions)$

```
1 /* Given approximation levels, PCA information, and probability
     distributions for errors compute the collocation points, the Psi
     Matrix, the polynomials and the integrals of the polynomials      */
```

output: $\{xPts, PsiMat, polys, intPolys\}$

Algorithm 15: *smolyakInterpolationPrep*

input : $(approxLevels, varRanges, distributions)$

```
1 /* Given approximation levels, variable ranges and probability
     distributions for errors compute the collocation points, the Psi
     Matrix, the polynomials and the integrals of the polynomials      */
```

output: $\{xPts, PsiMat, polys, intPolys\}$

Algorithm 16: *smolyakInterpolationPrep*

input : $(approxLevels, varRanges, distributions)$

```
1 /* Given approximation levels, variable ranges and probability
     distributions for errors compute the collocation points, the Psi
     Matrix, the polynomials and the integrals of the polynomials      */
```

output: $\{xPts, PsiMat, polys, intPolys\}$

Algorithm 17: *genSolutionErrXtEps*

input : $(approxLevels, varRanges, distributions)$

```
1 /* Given approximation levels, variable ranges and probability
     distributions for errors compute the collocation points, the Psi
     Matrix, the polynomials and the integrals of the polynomials      */
```

output: $\{xPts, PsiMat, polys, intPolys\}$

Algorithm 18: *genSolutionErrXtEpsZero*

input : $(approxLevels, varRanges, distributions)$

```
1 /* Given approximation levels, variable ranges and probability
     distributions for errors compute the collocation points, the Psi
     Matrix, the polynomials and the integrals of the polynomials      */
```

output: $\{xPts, PsiMat, polys, intPolys\}$

Algorithm 19: *genSolutionPath*