FEDERAL RESERVE BANK OF SAN FRANCISCO

WORKING PAPER SERIES

Higher-Order Perturbation Solutions to Dynamic, Discrete-Time Rational Expectations Models

Eric Swanson
Federal Reserve Bank of San Francisco

Gary Anderson
Federal Reserve Board

Andrew Levin
Federal Reserve Board

January 2006

Working Paper 2006-01
http://www.frbsf.org/publications/economics/papers/2006/wp06-01bk.pdf

Higher-Order Perturbation Solutions to Dynamic, Discrete-Time Rational Expectations Models

Eric Swanson[1], Gary Anderson[2], and Andrew Levin[2]
[1]Federal Reserve Bank of San Francisco and [2]Federal Reserve Board
eric.swanson@sf.frb.org, ganderson@frb.gov, levina@frb.gov

Abstract

We present an algorithm and software routines for computing nth-order Taylor series approximate solutions to dynamic, discrete-time rational expectations models around a nonstochastic steady state. The primary advantage of higher-order (as opposed to first- or second-order) approximations is that they are valid not just locally, but often globally (i.e., over nonlocal, possibly very large compact sets) in a rigorous sense that we specify. We apply our routines to compute first- through seventh-order approximate solutions to two standard macroeconomic models, a stochastic growth model and a life-cycle consumption model, and discuss the quality and global properties of these solutions.

JEL Classification: C61, C63, E37

This Version: January 12, 2006
First Version: December 2002

The first version of this paper was prepared for the 2003 AEA Meetings in a session on perturbation methods and applications. We thank Ken Judd, Chris Sims, Michel Juillard, Jinill Kim, and seminar participants at the AEA Meetings, ES Meetings, SCE Meetings, SED Meetings, SITE, and the University of Maryland for helpful discussions, comments, and suggestions. The views expressed in this paper, and all errors and omissions, should be regarded as those solely of the authors, and do not necessarily represent those of the individuals or groups listed above, the Federal Reserve Board, the management of the Federal Reserve Bank of San Francisco, or any other individual within the Federal Reserve System.

1. Introduction

An increasing number of authors have found the standard log-linearization procedure in macroeconomics insufficient for solving interesting problems. For example, Kim and Kim (2002) and Kollmann (2002, 2003) show the importance of second-order approximations for measuring welfare gains from trade or from international monetary policy coordination. Gaspar and Judd (1997), Judd (1999), and Woodford (2003) show the importance of second-order approximations for the law of motion of the economy when one is calculating the optimal policy in many interesting problems that arise in practice.

In this paper, we present an algorithm and software routines that compute an nth-order Taylor series approximation to the solution of a dynamic, discrete-time set of rational expectations equations around a nonstochastic steady state. Such approximate solutions are referred to as "perturbation method" solutions by Judd (1999). Our routines represent an improvement over other authors' work in this area in that we can approximate the true solution to arbitrarily high order, we can consider models with arbitrary shock distributions, and we can compute the coefficients in the Taylor series solution to a given level of numerical precision, thereby ensuring that the coefficients we calculate at higher orders are accurate.

We will present examples below that show the usefulness of all of the above features, but the first deserves special emphasis. The primary advantage of nth-order (as opposed to first- or second-order) approximations is that, so long as the true solution is analytic, the nth-order approximation is guaranteed to converge to the true solution everywhere within the domain of convergence (the multidimensional analog of the radius of convergence) of the Taylor series. Moreover, given a compact set of any size within the domain of convergence, there exists a finite n such that the nth-order approximation achieves any given level of accuracy over the entire compact set. Thus, there is a very rigorous sense in which a higher-order approximation is *globally*—and not just locally—valid.[1]

To demonstrate some of the potential applications of our solution algorithm and routines, we compute the first- through seventh-order approximate solutions to two widely-

[1] The advantage of perturbation methods over alternative procedures (such as projection methods or discretization methods) is that they are generally much faster and capable of handling larger models, as discussed by Gaspar and Judd (1997) and Aruoba, Fernandez-Villaverde, and Rubio-Ramirez (2005). We will focus on higher-order perturbation solutions in this paper; readers who are interested in comparisons across methods are referred to the work by those authors.

studied macroeconomic models: a stochastic growth model and a life-cycle consumption model with non-normally-distributed shocks to income. We show that the solution to the first model is close to linear and extremely close to quadratic for standard parameterizations, while the solution to the second model has much more important nonlinearities due to the greater role played by uncertainty. We discuss the quality and global properties of our solutions to these models.

The remainder of the paper proceeds as follows. Section 2 presents our algorithm for computing nth-order approximate solutions to dynamic, discrete-time rational expectations models. Section 3 presents our "PerturbationAIM" software implementation of the algorithm and discusses its advantages. Section 4 works through the solution to the two examples described above. Section 5 concludes. Three appendices provide the mathematical proof of the global validity of our approximation, the source code for our PerturbationAIM software routine, and the third-order approximate solution to all the variables of the stochastic growth model as a benchmark.

2. The Algorithm

2.1 Model Setup and Notation

We consider a system of dynamic, discrete-time rational expectations equations of the form:

$$E_t\, F(x_{t-\underline{\theta}}, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_{t+\overline{\theta}}\,;\, \varepsilon_t, \varepsilon_{t+1}, \ldots, \varepsilon_{t+\overline{\phi}}) = 0 \tag{1}$$

where F, x_t, and ε_t are vectors of dimensions n_F, n_x, and n_ε, respectively, E_t denotes the mathematical expectation conditional on all variables dated t and earlier, $\{\varepsilon_t\}$ is an exogenous stochastic process, and it is understood that the system of equations (1) is satisfied at each time t, $t+1$, $t+2$, etc., but not necessarily at time $t-1$ or earlier. The parameters $\underline{\theta}$, $\overline{\theta}$, and $\overline{\phi}$ denote the maximum lag length and lead lengths required to describe the equations in system (1).[2]

[2] Stochastic shocks dated $t-1$ and earlier can be incorporated into (1) by defining an auxiliary variable in x—e.g., by setting $a_t = \varepsilon_t$, and considering a_{t-1}. Lagged expectation operators can be incorporated by setting $b_t = E_t x_{t+1}$ and then considering b_{t-1}. In most macroeconomic models, $\overline{\phi}$ is typically either 0 or 1—i.e., no shocks dated later than $t+1$—but our algorithm and software routines are valid for general $\overline{\phi}$.

We will assume that the stochastic shocks ε_t are i.i.d. across time and that the components of ε_t (denoted ε_{it}, $i = 1, \ldots, n_\varepsilon$) are mutually independent as well. In other words, any cross-sectional or intertemporal correlation of the $\{\varepsilon_t\}$ process must be explicitly specified by the modeler as sums of individual components ε_{it}. We assume that $E[\varepsilon_{it}] = 0$, and we let $\text{Mom}_n(\varepsilon_{it})$ denote $E[\varepsilon_{it}^n]$, the nth moment of ε_{it}. When an nth moment is specified, we require that it exists. We require no other distributional assumptions regarding the ε_t, and denote the distribution function for ε_t by $\Psi(z)$.

We look for time-invariant, analytic, ergodic solutions to (1) of the form:[3]

$$x_t \;=\; b(x_{t-\underline{\theta}}, \ldots, x_{t-1}\,;\, \varepsilon_t) \tag{2}$$

In other words, variables dated $t-1$ and earlier are regarded as having been observed, and the vector x_t is to be solved as a function of these lagged values and the observed value of the stochastic shock ε_t. Of course, the solution (2) also depends on the coefficients and parameters of the model (1), and in particular on the moments of the stochastic disturbances ε_t.

It is worth noting that we do not require the modeler to specify some components of x_t as being "state" or "predetermined" variables and the remaining components as being "co-state" or "jump" variables.[4]

Following Fleming (1971) and Judd (1999), we let $\sigma \in [0, 1]$ denote an auxiliary, "scaling" parameter for the distribution of the stochastic shocks ε_t in (1). In particular, we consider a continuum of auxiliary models $(1)'$, parameterized by $\sigma \in [0, 1]$, each identical to (1) in every respect except that the distribution function for ε_t in these auxiliary models is given by $\Psi(z/\sigma)$ instead of by $\Psi(z)$. Thus, $\sigma = 1$ corresponds to the original model (1), which is to be solved, while small values of σ correspond to versions of the model with relatively little uncertainty. The case $\sigma = 0$ is taken to mean $\varepsilon_t = 0$ with probability 1—i.e., a deterministic version of the model.

[3] By time-invariance of b, we require that $x_{t+k} = b(x_{t+k-\underline{\theta}}, \ldots, x_{t+k-1}; \varepsilon_{t+k})$ for all $k \geq 0$ (note that we do not require b to have held for $k < 0$). We require b to be analytic in order for the Taylor series to converge to b as $n \to \infty$. The ergodicity requirement rules out "bubbles" and solution functions b that are globally explosive. The use of the letter b to denote the solution function generalizes Anderson and Moore's (1985) AIM algorithm for linear models, which produces a solution matrix denoted by B.

[4] This is just as in the linear case: see Anderson and Moore (1985) and Sims (2000). Intuitively, the computer can figure out what linear combinations of variables are "predetermined" from the fact that variables dated $t-1$ or earlier are known. For example, a clearly "predetermined" variable, such as $K_t = (1-\delta)K_{t-1} + I_{t-1}$, falls out of the solution algorithm into the form (2) trivially.

Thus, we are considering a family of models of the form:

$$E_t\, F(x_{t-\underline{\theta}}, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_{t+\overline{\theta}}\,;\, \varepsilon_t, \varepsilon_{t+1}, \ldots, \varepsilon_{t+\overline{\phi}}\,;\, \sigma) = 0$$
$$\varepsilon_s \sim iid\ \Psi(z/\sigma),\ s > t$$

or, equivalently,

$$E_t\, F(x_{t-\underline{\theta}}, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_{t+\overline{\theta}}\,;\, \varepsilon_t, \sigma\varepsilon_{t+1}, \ldots, \sigma\varepsilon_{t+\overline{\phi}}\,;\, \sigma) = 0 \qquad (1)'$$
$$\varepsilon_s \sim iid\ \Psi(z),\ s > t$$

to which we are looking for a family of solutions indexed by σ:

$$x_t \;=\; b(x_{t-\underline{\theta}}, \ldots, x_{t-1}\,;\, \varepsilon_t\,;\, \sigma) \qquad (2)'$$

We have recycled the letters F and b here, but there is no risk of confusion as we will henceforth only refer to the generalized family of equations $(1)'$ and $(2)'$, and specify $\sigma = 1$ when we wish to refer to the original model (1) and solution (2).

Note in particular that we do *not* scale the time-t realization of the shock ε_t in $(1)'$ and $(2)'$ by σ, because ε_t is known at time t and because it is often the case in practice that the modeler wishes to shock a deterministic or "perfect foresight" model—i.e., a model for which $\sigma = 0$. Specifications $(1)'$ and $(2)'$ are the proper parameterizations that allow the researcher to perform this kind of counterfactual experiment in which agents are completely surprised by a shock that they did not think could occur.[5]

2.2 Approximate Solutions to the Model

Finding the nonlinear solution function b is difficult in general. As is standard practice, we assume that the modeler can solve $(1)'$ for a nonstochastic steady state $\overline{x}$:

$$F(\overline{x}, \ldots, \overline{x}, \ldots, \overline{x}\,;\, 0, \ldots, 0\,;\, 0) = 0 \qquad (3)$$

so that we have:

$$\overline{x} \;=\; b(\overline{x}, \ldots, \overline{x}\,;\, 0\,;\, 0) \qquad (4)$$

[5] There is an earlier literature on "perfect foresight" solutions of nonlinear rational expectations models (e.g., Anderson (1993), Fuhrer and Madigan (1997), and the Troll software package), which solve the model to numerical precision imposing the constraint that $\sigma = 0$. In these algorithms, the modeler can still see how the perfect foresight solution reacts to a shock to the system, graph impulse responses to a shock, and so on. Our perturbation approach nests this older literature very naturally.

We then assume that there exists a neighborhood of this steady state for which the solution function b exists and is unique and on which F and b are sufficiently smooth. We substitute the relationships:

$$x_t = b(x_{t-\underline{\theta}}, \ldots, x_{t-1} \,;\, \varepsilon_t \,;\, \sigma) \tag{2}'$$

$$x_{t+1} = b\big(x_{t+1-\underline{\theta}}, \ldots, x_{t-1}, b(x_{t-\underline{\theta}}, \ldots, x_{t-1}; \varepsilon_t; \sigma) \,;\, \sigma\varepsilon_{t+1} \,;\, \sigma\big) \tag{2}''$$

etc.

into $(1)'$ and use the implicit function theorem to compute the second- and higher-order derivatives of the unknown function b with respect to $x_{t-\underline{\theta}}, \ldots, x_{t-1}$, ε_t, and σ at the known point $(\overline{x}, \ldots, \overline{x} \,;\, 0 \,;\, 0)$.

Like log-linearization, then, our approximate solution to $(1)'$ for any *fixed* order of approximation n (such as the now-popular quadratic approximation, $n = 2$) would only be valid in general when the lagged values $x_{t-\underline{\theta}}, \ldots, x_{t-1}$ lie within a sufficiently small neighborhood of $\overline{x}$, and ε_t and σ lie within a sufficiently small neighborhood of 0. It is crucial to note, however, that so long as F and b are analytic, the algorithm above is guaranteed to converge to b everywhere within the domain of convergence of the Taylor series expansion of b around the point $(\overline{x}, \ldots, \overline{x} \,;\, 0 \,;\, 0)$. This domain is potentially very large—in some models it can even be all of $\mathbb{R}^{\underline{\theta} n_x + n_\varepsilon + 1}$. Even better, the convergence is uniform on compact subsets.

The importance of these global properties of the algorithm are generally underappreciated in the profession. We thus formally state and prove the key point as a proposition:

Proposition 1: *Let $\Omega \subseteq \mathbb{R}^m$ be a domain and $b : \Omega \to \mathbb{R}^h$ be analytic at the point $\overline{x} \in \Omega$. Let b_n denote the nth-order Taylor series expansion of b about the point $\overline{x}$ and let $D \subseteq \Omega$ be the domain of convergence of the Taylor series expansion. Then for any given $K \subseteq D$ compact and $\epsilon \in \mathbb{R}^+$, there exists an integer N such that $|b_n(x) - b(x)| < \epsilon$ uniformly for all $x \in K$ and all $n \geq N$.*

PROOF: See Appendix A for technical discussion and proof. □

The proposition emphasizes that, for any given problem with an analytic solution b and any (potentially very large) compact set K within the domain of convergence of the Taylor series for b about the point $\overline{x}$, there exists a *finite* N such that the Taylor series approximation is as accurate as desired over the *entire* set K. Thus, there is a very rigorous sense in which our algorithm is *globally*—and not just locally—valid.

In contrast to other researchers (Schmitt-Grohe and Uribe (2004), Kim, Kim, Schaumberg and Sims (2002)), our algorithm and software routines allow for approximation of the function b to arbitrarily high order n, as opposed to simply the second order. We are indebted to Judd (1999) for making the essential point that, once the first derivatives of b have been calculated, one needs only to solve a straightforward, recursive linear problem to calculate the derivatives of b to each successive order. To solve for the first derivatives of b, we use AIM, a generalization of the Blanchard-Kahn (1980) algorithm developed by Anderson and Moore (1985).[6]

2.3 Limitations of Perturbation Methods

Perturbation methods are in general significantly faster than alternatives such as projection methods or discretization methods, as discussed by Gaspar and Judd (1997) and Aruoba, Fernandez-Villaverde, and Rubio-Ramirez (2005). As emphasized by Proposition 1 in the previous section, they are also globally valid for many interesting problems. Nonetheless, there are many models which are not amenable to this type of solution procedure. For example, the system of equations F must be free of inequality constraints and must be globally analytic over the domain of interest.[7] To the extent that the user is interested in inequality or other non-smooth constraints F, and is not willing to approximate them with an analytic penalty function or analytic approximation to the non-smooth constraints, then alternative procedures, such as projection or discretization methods, must be employed.

3. Software Implementation of the Algorithm: PerturbationAIM

We implemented the solution procedure described above in Mathematica and refer to the software implementation as "PerturbationAIM."

[6] Anderson (2001) shows that AIM is faster and more numerically robust than the numerous other alternatives available. Like all of these first-order procedures, AIM requires that the solution function b be first-order stable, or at least not too explosive, local to the nonstochastic steady state. AIM treats unit roots as stable by default, so unit roots in the first-order approximation satisfy this stability requirement. Users can also modify the Blanchard-Kahn stability requirement in AIM to be satisfied by eigenvalues less than λ in absolute value, where the user may choose $\lambda < 1$ or $\lambda > 1$ as well as the standard $\lambda = 1$.

[7] If F is analytic locally but not globally, then the nth-order approximations will generally converge to b locally (including σ very close to zero), but this near-perfect-foresight solution is typically of much less interest than the true stochastic solution, which corresponds to the nonlocal case $\sigma = 1$.

3.1 General Advantages of PerturbationAIM

While second-order approximations can be done fairly easily in any computer language, generalization to higher orders is dramatically simpler in Mathematica, owing to its large library of built-in routines for symbolic differentiation, symbolic manipulation, and solution of large-scale systems of linear equations. In particular, the use of Mathematica:

A. Allows for much simpler and intuitive code (less than 200 lines), making the implementation of the algorithm more transparent and more likely to be free of human coding error (bugs).

B. Allows the algorithm to manipulate symbolic expressions and thus proceed in the same way and produce the same output as if the user were performing it by hand, making the implementation of the algorithm and the generated output more intuitive.

C. Eliminates the need for user-written differentiation or linear solution routines, which are likely to be more amateur than built-in Mathematica routines and hence more likely to suffer from human coding error and numerical inaccuracies.

D. Allows the use of symbolic differentiation, improving numerical accuracy, particularly at higher orders.

E. Allows the computation of all coefficients to arbitrarily high precision, eliminating inaccuracies from machine-precision computations that cumulate with each recursive step of the algorithm.

Our experience has shown that point E is surprisingly relevant in practice as well as in principle. Errors in machine-precision arithmetic were observed quite frequently at higher orders or for larger models, probably due to the fact that the recursive nature of the algorithm implies that a few digits of accuracy are lost with each successive iteration. We will return to this issue in the specific context of the stochastic growth model, below.

3.2 Modeling Advantages of PerturbationAIM

In addition to the general, internal advantages of PerturbationAIM discussed above, the software has a number of advantages from the economic modeler's point of view, as follows:

1. No need to categorize variables as "predetermined" or "non-predetermined." Knowing that all variables dated $t-1$ or earlier are observed and all variables dated t are to be solved is sufficient for the computer to determine which linear combinations of variables are "predetermined" and which "non-predetermined."

Anderson and Moore (1985), Sims (2000), and Collard and Juillard (2001) also make this observation. This feature of the code becomes increasingly convenient as the number of variables in the model increases.

2. No need for the modeler to substitute out identities. The computer is completely capable of making these substitutions.

3. Ability to solve models with arbitrarily long lags and leads (including shocks dated later than t or $t+1$), with no need for the modeler to transform the model by hand. Note also that the usual trick for linear models of defining an auxiliary variable $z_t \equiv E_t x_{t+1}$, and then considering z_{t+1}, fails for nonlinear models because $F(E_t x_{t+1}) \neq E_t F(x_{t+1})$ in general. PerturbationAIM handles multiple leads correctly in the nonlinear as well as the linear case.

4. Ability to solve models with non-normally distributed shocks. PerturbationAIM allows the modeler to substitute in any set of moments for the stochastic shocks of the model, allowing consideration of models with any shock distribution for which the moments are finite.

3.3 Outline of PerturbationAIM

The software implementation of the algorithm proceeds as follows:

1. Find the nonstochastic steady state solution $(\overline{x}, \ldots, \overline{x}, \ldots, \overline{x}\,;\, 0, \ldots, 0\,;\, 0)$ to $(1)'$ using a standard nonlinear equation solver.

2. Define the (unknown) solution function b^{x_i} for each element x_i of x, $i = 1, \ldots, n_x$, with arguments $(x_{t-\underline{\theta}}, \ldots, x_{t-1}\,;\, \varepsilon_t\,;\, \sigma)$. Let b denote $(b^{x_1}, \ldots, b^{x_{n_x}})'$.

3. Compute the first derivatives of b at the point $(\overline{x}, \ldots, \overline{x}\,;\, 0\,;\, 0)$ using AIM, as follows: Differentiate the function F with respect to each of its arguments and evaluate these derivatives at $(\overline{x}, \ldots, \overline{x}, \ldots, \overline{x}\,;\, 0, \ldots, 0\,;\, 0)$. Write out the first-order Taylor series approximation to F as a linear model in its arguments and find the solution to this linear model using AIM. The AIM solution matrix yields the desired first derivatives of b.

In preparation for computing the second- and higher-order derivatives of b:

4. Substitute out for x_{t+k}, $k > 0$, in $(1)'$ using the relationships $(2)'$, $(2)''$, etc. For notational simplicity, denote the result by:

$$E_t\,(F \circ b) = 0 \tag{5}$$

Note that $F \circ b$ has arguments $(x_{t-\underline{\theta}}, \ldots, x_{t-1}\,;\, \varepsilon_t, \ldots, \varepsilon_{t+\max(\overline{\theta},\overline{\phi})}\,;\, \sigma)$.

Now, for each order $n \geq 2$, calculate the nth derivatives of b as follows:

5. Calculate all the nth-order derivatives of $F \circ b$ with respect to its arguments, as follows: The derivatives of F are known (and are symbolic), the derivatives of b up through degree $n-1$ are unknown in general (and are symbolic) but are known at the steady state (and are numerical). The nth-order derivatives of b are unknown in general (and are symbolic) and are unknown at the steady state (and are undetermined numbers).

6. Compute the nth-order Taylor series approximation to $F \circ b$ at the steady state, as follows: The (known) derivatives of F evaluated at steady state yield numerical values, the (known) derivatives of b up through degree $n-1$ at steady state yield numerical values, and the (unknown) nth derivatives of b at steady state are denoted by a set of (unknown) symbolic constants. Write out the nth-order Taylor series approximation to $F \circ b$ with the nth derivatives of b left as undetermined coefficients.

7. Take the expectation implied by equation (5) for the nth-order Taylor series approximation to $F \circ b$ by sending terms involving ε_{t+k}, $k \geq 1$, into the (known) corresponding moments.

8. Solve for the nth-order derivatives of b using the method of undetermined coefficients: Set each coefficient of the nth-order Taylor series of $F \circ b$ equal to zero (as implied by equation (5)) and solve the resulting linear system of equations for the undertermined nth derivatives of b.

Repeat steps 5–8 for each successive order n, as desired.

3.4 Algorithmic Enhancements

The PerturbationAIM source code closely follows the outline above, but also contains a few algorithmic enhancements that considerably increase computational efficiency and accuracy. Two of these in particular are worth discussing in greater detail.

3.4.1 Parsimonious Definition of the Economic State

Recall our definition of the solution function for x_t in equation $(2)'$ as:

$$x_t \;=\; b(x_{t-\underline{\varrho}}, \ldots, x_{t-1} \,;\, \varepsilon_t \,;\, \sigma)$$

Note that in this definition, we have implicitly defined the economic state vector to be all of $(x_{t-\underline{\varrho}}, \ldots, x_{t-1} \,;\, \varepsilon_t)$, as well as the auxiliary scaling parameter σ. In fact, the state

of the model at time t can typically be defined much more parsimoniously than this, as follows.

Let x_t^i, $i = 1, \ldots, n_x$, denote the n_x components of the vector x_t. For each i, let $\underline{\theta}^i$ denote the longest lag with which the ith component of x appears in the original system of equations F in (1)—i.e., the maximum k for which x_{t-k}^i appears in F. Note that $\underline{\theta}^i$ may be zero for some i (if x^i does not appear in F with a lag at all). Then, when we consider the system F at dates t, $t+1$, $t+2$, ..., the variables $x_{t-\underline{\theta}^i}^i$, $x_{t-\underline{\theta}^i+1}^i$, $x_{t-\underline{\theta}^i+2}^i$, ..., will all appear, but x_{t-k}^i for any $k > \underline{\theta}^i$ will not appear. Thus, we may vary x_{t-k}^i, $k > \underline{\theta}^i$, freely without affecting any of the constraints on the vector x_t (and x_{t+1}, x_{t+2}, ...). It follows that the inherited economic state of the model at date t is completely described by the set of variables $\bigcup_{i=1}^{n_x} \{x_{t-k}^i\}_{k=1}^{\underline{\theta}^i}$ which do appear in F with a lag (where if $\underline{\theta}^i = 0$ the quantity in braces is taken to be the empty set). Note that this set of variables is in general smaller than $\{x_{t-\underline{\theta}}, \ldots, x_{t-1}\}$, with equality if and only if $\underline{\theta}^i = \underline{\theta}$ for each i; in practice, $\bigcup_i \{x_{t-k}^i\}_{k=1}^{\underline{\theta}^i}$ is typically much smaller than $\{x_{t-\underline{\theta}}, \ldots, x_{t-1}\}$.

Characterizing the economic state more succinctly as $\bigcup_i \{x_{t-k}^i\}_{k=1}^{\underline{\theta}^i}$ typically yields large gains in computational speed and accuracy, since any reduction in the number of arguments of b dramatically reduces the number of derivatives of $F \circ b$ that must be computed and the number of derivatives of b that must be solved at each step.

3.4.2 Intermediate Computation of a Certainty-Equivalent Solution

In solving for the nth-order derivatives of b, the problem is separable into two stages: a certainty-equivalent stage and a stochastic stage. In particular, we may consider a certainty-equivalent version of model $(1)'$ by setting $\sigma = 0$ in $(1)'$. The solution to this model is given by :

$$x_t^{CE} \;=\; b(x_{t-\underline{\theta}}, \ldots, x_{t-1} \,;\, \varepsilon_t \,;\, 0)$$

Thus, it is possible to solve for all of the nth-order certainty-equivalent derivatives (i.e, those that do not involve σ) of the *true*, stochastic solution function b by restricting attention first to the case of $F \circ b$ with $\sigma = 0$. Once we have computed the nonstochastic derivatives of b by implicit differentiation of $F \circ b$ evaluated at $\sigma = 0$ and solving, we may then return to the computation of the remaining derivatives of b (i.e., those involving σ) by computing the derivatives of $F \circ b$ that involve σ and considering the case $\sigma > 0$.

It turns out that this two-step procedure is more computationally efficient than solving for all of the nth-order derivatives of b simultaneously. No extra computation is performed, because all of the derivatives of $F \circ b$ are computed once and only once by either method, and efficiency is gained because it is faster to solve the resulting linear system of equations in the block-recursive manner that is imposed by the two-step procedure.

3.5 Source Code

The complete Mathematica source code for PerturbationAIM is very succinct (less than 200 lines, excluding comments) and is included in Appendix B for reference. Copies can also be freely downloaded from http://www.ericswanson.pro, or are available from the authors upon request. All that is required of the user is to specify a "model file" with the equations of F written out in standard Mathematica notation, using time indices t, $t-1$, $t+1$, etc. to convey to the software the correct intertemporal relationship of the variables in the model. Simple examples and model file templates are available for download along with the PerturbationAIM algorithm.

4. Examples

We illustrate some of the applications and potential benefits of PerturbationAIM by means of two examples that are standard in the macroeconomics literature: a stochastic growth model, and a life-cycle consumption model.

4.1 Stochastic Growth Model

A natural example that illustrates the features of the algorithm is a standard stochastic growth model:

$$Y_t = A_t K_{t-1}^{\alpha} \tag{6}$$

$$\log A_t = \rho \log A_{t-1} + \varepsilon_t \tag{7}$$

$$K_t = (1-\delta) K_{t-1} + I_t \tag{8}$$

$$Y_t = C_t + I_t \tag{9}$$

$$C_t^{-\gamma} = \beta E_t\left[(1+r_{t+1})\, C_{t+1}^{-\gamma}\right] \tag{10}$$

$$r_t = \alpha A_t K_{t-1}^{\alpha-1} - \delta \tag{11}$$

$$\text{Welf}_t = \frac{C_t^{1-\gamma}}{(1-\gamma)} + \beta E_t \text{Welf}_{t+1} \tag{12}$$

We desire a solution to (6)–(12) for A_t, C_t, I_t, K_t, r_t, Y_t, and Welf_t as functions of variables dated $t-1$, ε_t, and σ.[8]

Equations (6)–(12) can be entered into a model file essentially exactly as they are written above—as discussed in section 3.2, the researcher does not need to categorize variables as being "predetermined" or "non-predetermined," does not need to substitute out identities such as (9), and is free to tack on auxiliary equations such as (12), which do not affect the solution to equations (6)–(11) but nonetheless produce a result that is of interest—in this case, the expected present discounted value of representative-agent utility flows conditional on all information at time t.

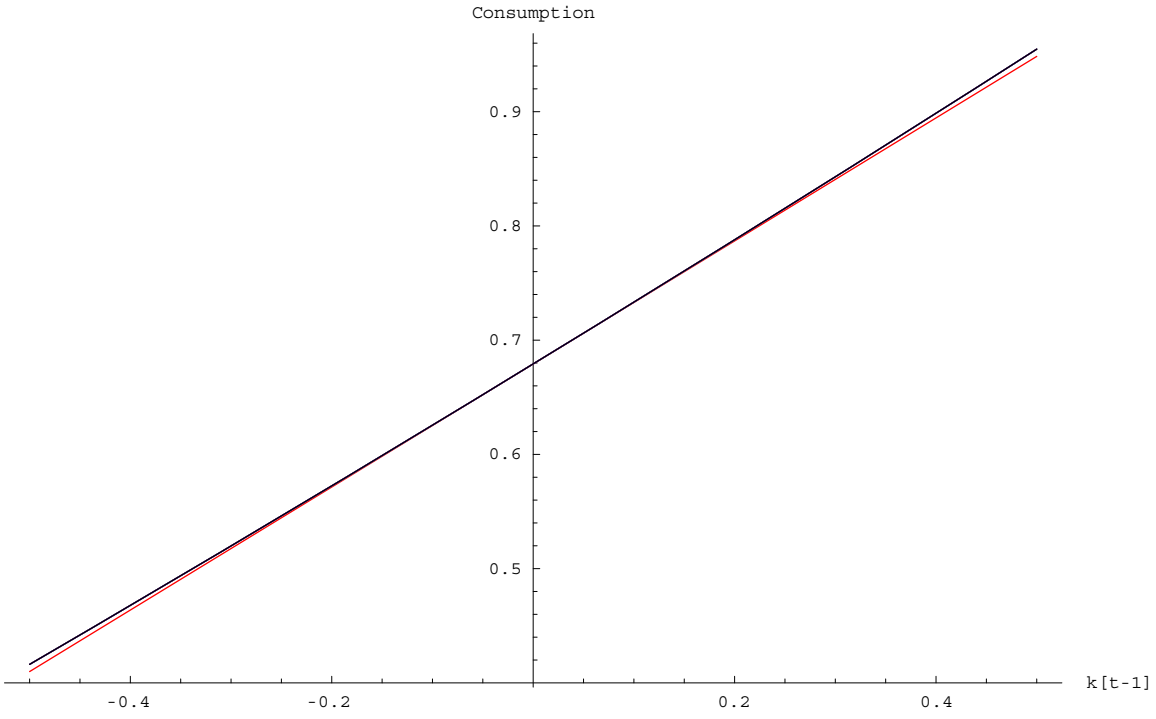
4.1.1 Results

We apply PerturbationAIM to the stochastic growth model above and compute the first- through seventh-order approximate solutions to each of the above variables. In computing these numerical results, we set $\alpha = .3$, $\beta = .99$, $\gamma = 1.1$, $\delta = .025$, and $\rho = .8$, in line with post-war quarterly U.S. data. We assume that the shocks ε_t are normally distributed with a standard deviation of .01. Since there is some reason to think that several variables in this model will be better approximated in logartihms than in levels, we have the software transform the variables A, C, K, and Y into logarithms, but leave investment I, the net interest rate r, and Welfare in levels, since I may be negative at times, r is close to zero, and Welf is negative in steady state.

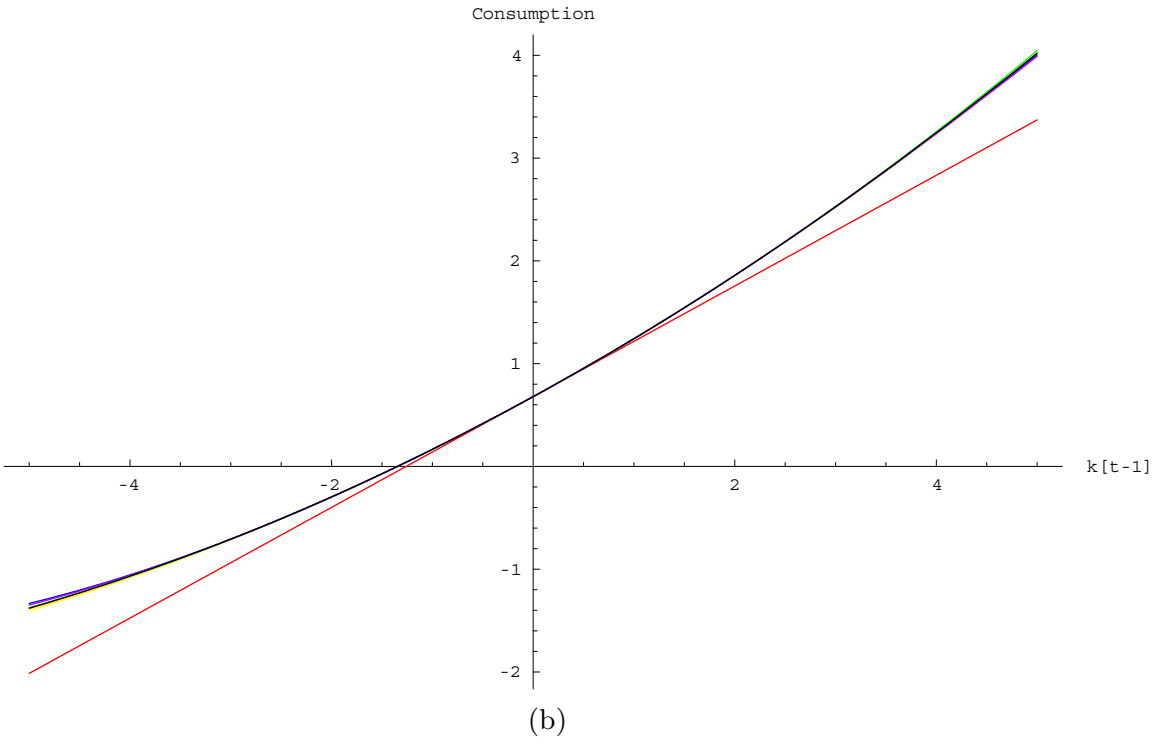
We compute the first- through seventh-order solutions for all of these variables around the nonstochastic steady state of the model. Figure 1 provides a sense of these results by graphing the solution for (the logarithm of) C_t as a function of (log) K_{t-1}, the primary state variable of interest. To keep this graph simple, all of the other state variables in the model are taken to be at their nonstochastic steady-state values, except for the auxiliary scaling paremeter σ, which we set equal to 1 (corresponding to the original model of interest). The horizontal axis of figure 1 denotes the deviation of $k_{t-1} \equiv \log K_{t-1}$ from its

[8] In fact, the economic state of this model can be characterized more parsimoniously by A_{t-1}, K_{t-1}, ε_t and σ. See section 3.4 for a detailed discussion.

FIGURE 1: CONSUMPTION IN THE STOCHASTIC GROWTH MODEL, $c_t(k_{t-1})$
FIRST- THROUGH SEVENTH-ORDER APPROXIMATE SOLUTIONS



(a)



(b)

Log consumption at time t as a function of the inherited log capital stock. Other state variables are assumed to be at their steady-state values and the auxiliary scaling parameter σ is set equal to 1. Red, orange, yellow, green, blue, violet, and black lines depict, respectively, the first- through seventh-order solutions. Horizontal axis is in (log) deviations from steady state; panel (b) considers a much wider range of inherited capital stocks than panel (a).

nonstochastic steady state value, and the vertical axis denotes the level of $c_t \equiv \log C_t$ (i.e., not as a deviation from steady state). In figure 1(a), we consider a range for k_{t-1} from about 50 percent below to about 50 percent above the nonstochastic steady-state level; in figure 1(b), we effectively "zoom out" to consider a much wider range of values for k_{t-1} of ± 5 natural logarithms from steady state, a factor of approximately 100 both above and below the steady-state level. In both panels, the first- through seventh-order solutions are graphed, respectively, in the colors red, orange, yellow, green, blue, violet, and black.

The most striking feature of figure 1 is just how well the second- and even first-order approximate solutions do at capturing c_t as a function of k_{t-1}. The second- through seventh-order solutions all lie virtually on top of one another and are essentially indistinguishable over the range of values in figure 1(a), and differ only very slightly even over the vast range of capital stocks considered in figure 1(b). Evidently, the solution for c_t is very close to log-quadratic or even log-linear over plausible ranges for the state variables.

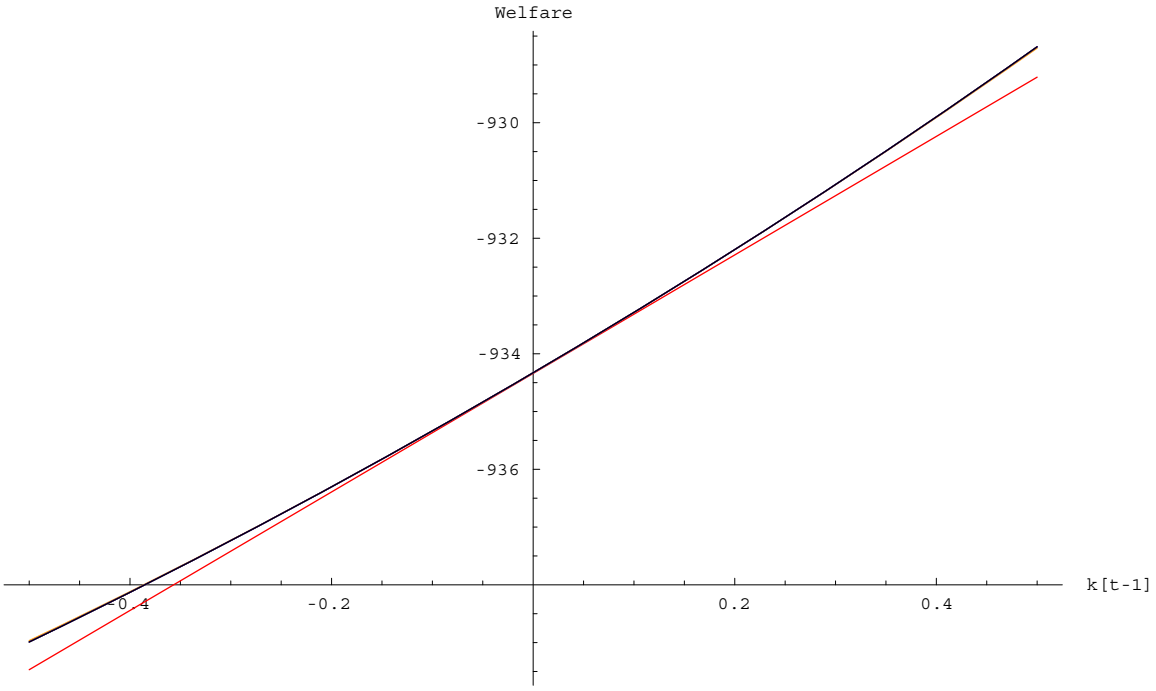
This qualitative conclusion holds not just for consumption, but for all of the possible choices of dependent and independent variables one could consider. Figure 2 presents the analogous graphs for the most nonlinear of the variables in the model, conditional welfare Welf_t. Even for that variable, the second- through seventh-order approximations are indistinguishable in panel (a), and the third- through seventh-order approximations are difficult to distinguish in panel (b) despite the vast range of capital stocks considered.

Given the results in figures 1 and 2, it is somewhat surprising that the stochastic growth model is so often used as a benchmark for computing nonlinear approximate solutions. Indeed, the log-linear approximation does quite well, and the quadratic approximation to the solution of this model is excellent, even over extremely large ranges for the inherited state variables. This is not true in general, as the life-cycle consumption model below will demonstrate.

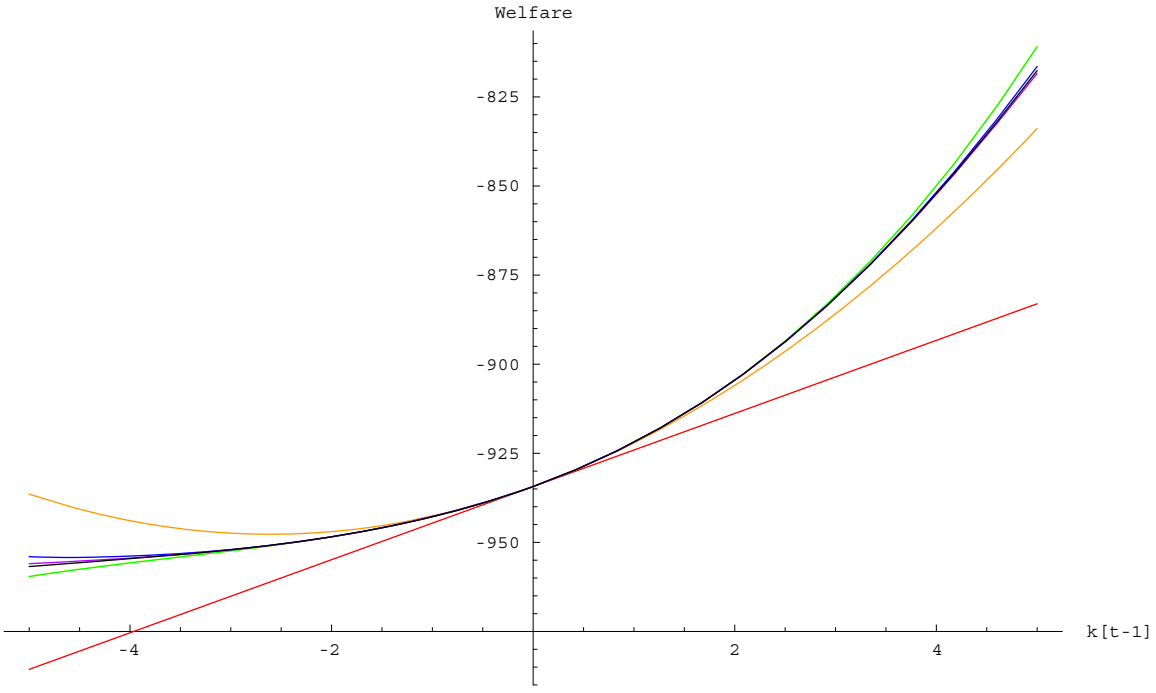
4.1.2 Potential Pitfalls of Machine-Precision Arithmetic

In theory, the coefficients of a Taylor series approximation can be computed to arbitrarily high order. In practice, however, as one proceeds to higher and higher orders of approximation, the recursive computation of derivatives of each successive order from the previous order compounds potential numerical inaccuracies in the computation at each step. An

FIGURE 2: WELFARE IN THE STOCHASTIC GROWTH MODEL, $\text{Welf}_t(k_{t-1})$
FIRST- THROUGH SEVENTH-ORDER APPROXIMATE SOLUTIONS



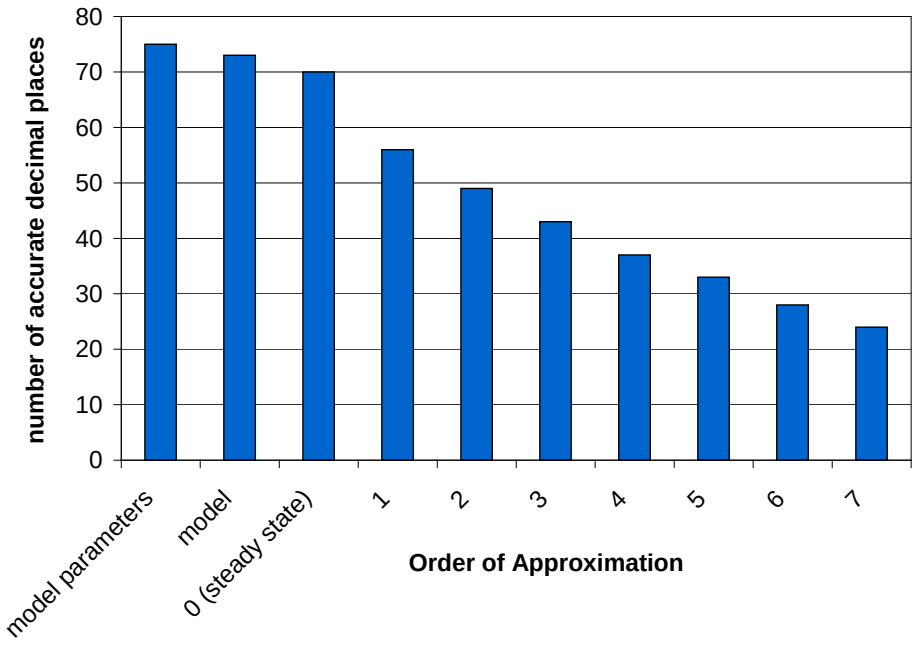
(a)



(b)

Expected present discounted value of period utility conditional on information at time t as a function of the inherited log capital stock. Other state variables are assumed to be at their steady-state values and the auxiliary scaling parameter σ is set equal to 1. Red, orange, yellow, green, blue, violet, and black lines depict, respectively, the first- through seventh-order solutions. Horizontal axis is in (log) deviations from steady state; panel (b) considers a much wider range of inherited capital stocks than panel (a).

Figure 3: Decimal Places of Taylor Series Coefficient Accuracy
stochastic growth model, first- through seventh-order approximate solutions



advantage of implementing PerturbationAIM in Mathematica is that it there are built-in procedures for reporting the accuracy of the computed results, and for increasing that accuracy.

Figure 3 illustrates the issue by graphing the number of decimal places of accuracy in the coefficients of the Taylor series approximation for each successive order of approximation to the stochastic growth model solution. The accuracy of the original model and of the steady state are also reported. For this example, we specified that the parameters of the model (α, β, etc.) were accurate to 75 decimal places. The model itself then has about 73 decimal places of accuracy—this is due to the presence of $1-\gamma$ in the denominator of equation (12), which reduces the accuracy of the model relative to the parameters because $(1-\gamma)^{-1}$ has lower accuracy than γ, particulary since γ is not too different from unity. The steady state of the model is successfully computed to about 70 decimal places (that is, a minimum of 70 decimal places; some variables may have been computed to higher accuracy than others, but all of them are accurate to 70 decimal places or more).

As is clear in figure 3, the accuracy of the computed Taylor series coefficients for each order of approximation falls steadily as we compute each successive order, down to about 24 decimal places by the time we reach the seventh order. While the fall is greatest from the

Table 1: Number of Taylor Series Coefficients with Errors Arising from Machine-Precision Arithmetic, Stochastic Growth Model

Order of Approx.	# nonzero coeffs in Taylor series for Consumption	# of machine-precision coeffs with error > 10^{-4}	# of machine-precision coeffs with error > 10^{-2}	# of machine-precision coeffs with relative error > 1%	# of machine-precision coeffs with relative error > 10%
0	1	0	0	0	0
1	4	0	0	0	0
2	11	1	0	1	0
3	25	5	1	5	1
4	50	14	4	14	5
5	91	29	9	29	18
6	154	59	18	56	34
7	246	103	21	104	68

zeroth to the first order, due to the repeated swapping of adjacent Schur blocks that must be performed to group the large eigenvalues together,[9] the essential feature of figure 3—that accuracy of the computed Taylor series coefficients falls steadily with each successive order—is an unavoidable consequence of the recursive nature of the algorithm. For this particular model, we seem to lose about five decimal places of accuracy in the computed Taylor series coefficients as we progress to each successive order of approximation.[10]

In Table 1, we investigate to what extent a researcher could go wrong by working entirely with machine-precision numbers—which have about 16 decimal places of accuracy—at each step of the above computation. We focus on the Taylor series solution for (log) consumption in particular, although the results are very similar for all of the non-predetermined endogenous variables in the model. The table reports both absolute errors—the raw difference between the machine-precision coefficients and the "true" values of those coefficients computed out to 23 decimal places or more—and the relative errors, which are the absolute errors divided by the true values.

While the errors from using machine-precision arithmetic are very small for the first few orders, they contaminate a larger and larger number of coefficients and become progressively larger as we proceed to each higher order. Coefficient errors as large as 10%

[9] The accuracy of this part of the algorithm could be improved—it is not built-in to Mathematica but was instead coded by us without great attention to numerical precision, since the arbitrary-precision capabilities of Mathematica make this less necessary.

[10] Dropping welfare equation (12)—the most badly scaled equation—from the model improves the accuracy of our first-order solution by a few decimal places relative to the version of the model with welfare included; nonetheless, the loss of about five decimal places of accuracy with each successive order of approximation remains true even when we drop welfare from the model.

become quite common by about the fourth or fifth order and coefficient errors of about 1% are present even in the second-order solution.

Machine-precision arithmetic varies significantly from machine to machine, and even from software package to software package. Thus, there is no guarantee that the results above will be representative of other researchers' experiences using different hardware or different software packages. Nonetheless, the qualitative observations of table 1 are unavoidable, and there are a number of reasons to think that they will be representative of other researchers' experiences: first, they have been computed on a standard PC with 32-bit Intel hardware, by far the most common platform in use in the profession; and second, Mathematica's internal numerical linear algebra procedures are all taken directly from LAPACK, which is by far the most common underlying set of numerical linear algebra routines in use today (these are the same procedures underlying Matlab, for example).

For future reference, the third-order solution to the stochastic growth model is provided in Appendix C to six significant digits, as a benchmark.

4.2 Life-Cycle Consumption Model

The stochastic growth model is a common benchmark in macroeconomics, but the relatively small shocks and near-linearity of the solution to the model make it less interesting for evaluating nonlinear numerical solution methods. We thus turn attention to a more nonlinear—and thus more interesting for our purposes—model of life-cycle consumption.

Equations of the model are given by:

$$\log Y_t = \log Y_{t-1} + \alpha + \varepsilon_t^y + \varepsilon_t^x \tag{13}$$

$$C_t^{-\gamma} = \beta(1+r)E_t\, C_{t+1}^{-\gamma} \tag{14}$$

$$A_t = (1+r)A_{t-1} + Y_t - C_t \tag{15}$$

where Y_t denotes the agent's exogenous income stream, C_t the agent's choice of consumption, A_t the agent's end-of-period asset stock, α the expected growth rate of income from one period to the next, r the risk-free real rate at which the agent can borrow and save, β the agent's discount rate, γ the inverse of the elasticity of substitution, and ε^y and ε^x exogenous shocks to income, with:

$$\varepsilon_t^y \sim N(0, \sigma_y^2)$$

and

$$\varepsilon_t^x \sim \begin{cases} -X & \text{with probability } p \\ X\,p/(1-p) & \text{with probability } 1-p \end{cases}$$

We interpret ε_t^y as year-to-year variation in income conditional on staying in the same job, and ε_t^x as the probability of the agent losing her job, resulting in a loss of annual income of about X percent. The shock ε_t^x is thus substantially non-normally distributed, skewed to the downside, and potentially very large, all of which make the implications of uncertainty in the model more interesting than was the case for the stochastic growth model.

We choose parameter values to roughly correspond to annual data for a U.S. consumer: $r = .05$, $\beta = .97$, $\gamma = 1.1$, $\sigma_y = .03$, and $p = .05$. For illustrative purposes, we will consider values of X equal to .05 and to .3. The latter value is more realistic, we would argue, but the former value offers insight by acting as an intermediate case that is closer to normally distributed. We calibrate α in such a way as to ensure that the model has a nonstochastic steady state, as follows.

Equations (13)–(15), as written, will typically not imply a nonstochastic steady state for A, C, and Y since these variables will trend over time. In order to apply our perturbation algorithm to the model, we require the model to have a nonstochastic steady state, so we define the normalized variables $y_t \equiv Y_t/Y_{t-1}$, $c_t \equiv C_t/Y_t$, and $a_t \equiv A_t/Y_t$ and rewrite the system (13)–(15) as:

$$\log y_t = \alpha + \varepsilon_t^y + \varepsilon_t^x \tag{16}$$

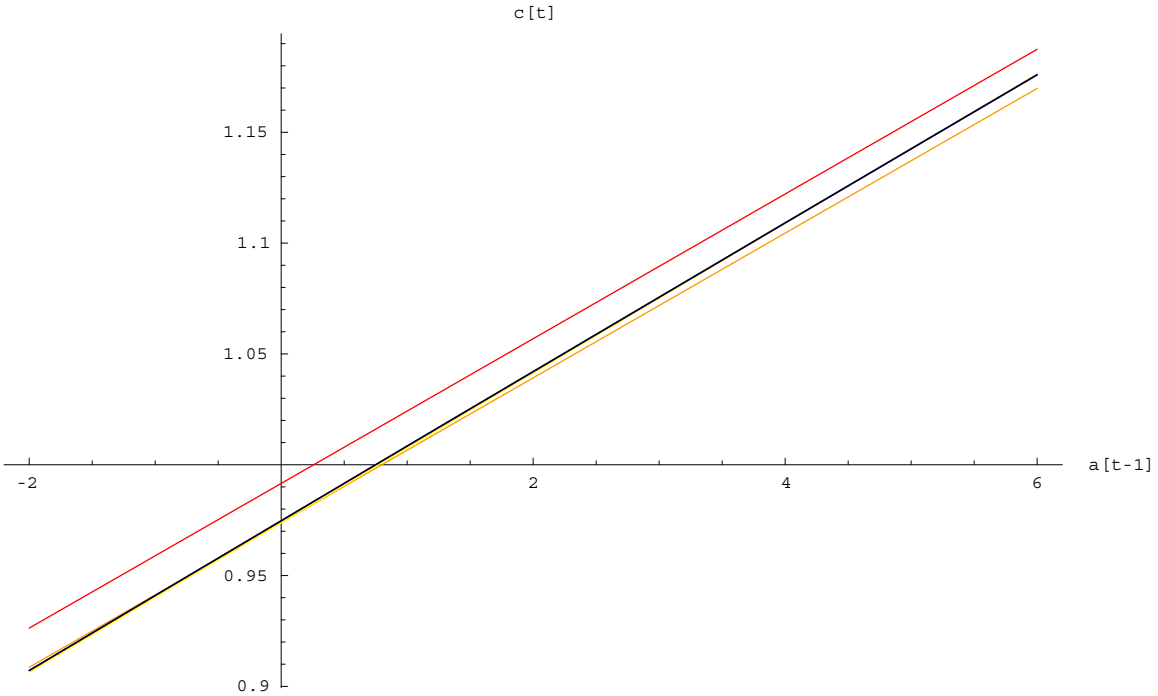
$$c_t^{-\gamma} = \beta(1+r)E_t\, c_{t+1}^{-\gamma} y_{t+1}^{-\gamma} \tag{17}$$

$$a_t y_t = (1+r)a_{t-1} + y_t - c_t y_t \tag{18}$$

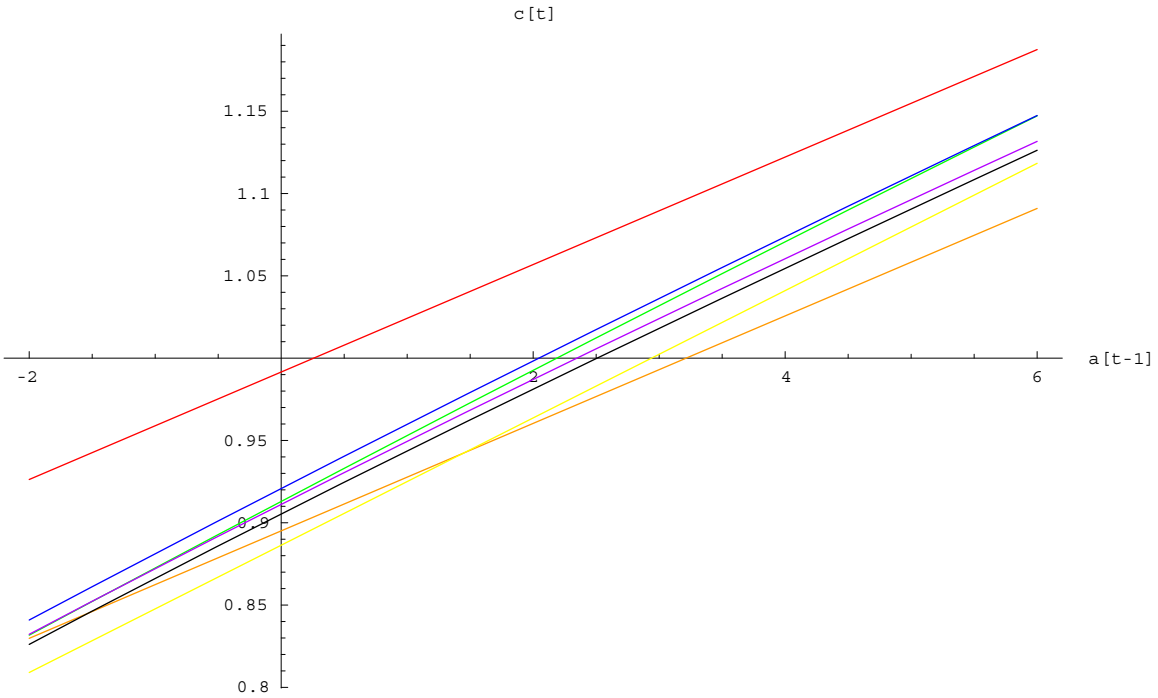
Finally, we set $\alpha = \gamma^{-1}\log[\beta(1+r)]$ to ensure that the model has a nonstochastic steady state.

Figure 4 reports the first- through seventh-order solutions for the cases of small negative income shocks arising from job loss ($X = .05$) in panel (a) and large income shocks from job loss ($X = .3$) in panel (b). For the small shocks to income (panel (a)), the linear approximation clearly misses the precautionary savings motive that is evident in the second- and higher-order solutions, which is not surprising given that the linearized model is certainty-equivalent. There is some discernible difference between the second-order and the higher-order approximate solutions, but nonetheless, the second-order approximation

FIGURE 4: CONSUMPTION IN THE LIFE-CYCLE MODEL, $c_t(a_{t-1})$
FIRST- THROUGH SEVENTH-ORDER APPROXIMATE SOLUTIONS



(a) $X = .05$



(b) $X = .3$

Normalized consumption at time t as a function of the inherited normalized asset stock. Other state variables are assumed to be at their steady-state values and the auxiliary scaling parameter σ is set equal to 1. Red, orange, yellow, green, blue, violet, and black lines depict, respectively, the first- through seventh-order solutions. Horizontal axis is in deviations from steady state. Panel (b) considers the case of larger and more negatively skewed income shocks in the event of job loss ($X = .3$). See text for details.

performs quite well, even over the relatively large range of asset stocks considered in the figure (ranging from two times annual income below steady state up to six times annual income above steady state).[11] The third- through seventh-order solutions are all essentially indistinguishable from one another, suggesting that convergence to the true solution for this version of the model has virtually been achieved by the third or fourth order.

The advantages of the higher-order approximations become apparent when we begin to consider less trivial, more realistic shocks to income in panel (b) ($X = .3$). In this panel, a clear difference emerges between each order of approximation, and only by the sixth or seventh order do we appear to be settling down to an extent that would suggest we are close to the true solution. Compared to the first- or second-order solutions, the third- and higher-order solutions are substantially steeper and more concave, indicating that it is optimal for the agent to save more at the margin when assets are lower, all else equal. The first-order solution, of course, misses all aspects of the precautionary savings motive—a huge oversight in this model—and the second-order solution, while doing better at low levels of assets, significantly overstates both the optimal level of savings and the marginal propensity to save as the agent's buffer stock of assets increases.

5. Conclusions

We have presented a specific algorithm and software routines for computing arbitrarily high-order Taylor series approximations to the true solution for a set of user-specified dynamic, discrete-time rational expectations equations.

Higher-order perturbation solutions may be of interest to a researcher for two reasons. First, in many problems the higher-order features of the model are themselves of interest, such as risk premia and time-varying risk premia in the asset pricing literature, and precautionary savings and "prudence" in the consumption literature. Second, there is a rigorous sense in which an nth-order solution is globally valid, which we specify and discuss. Given the computational advantages of perturbation methods over alternatives such as projection or discretization methods, the global validity of the procedure and avail-

[11] The steady-state level of the asset stock in the model is close to zero, so the deviations from steady state reported on the horizontal axis are also essentially equal to the levels.

ability of an off-the-shelf set of software routines may make these methods attractive to researchers working on a variety of problems.

An additional advantage of our software implementation is the ability to compute the Taylor series coefficients of the approximate solution to arbitrarily high numerical precision. Our experience has shown that numerical errors in these coefficient computations cumulate relatively quickly as one progresses to the computation of higher-order derivatives. Thus, our software routines should serve as a useful benchmark for other researchers going forward.

It is also our hope that researchers will adapt, modify, and apply our software routines in new and surprising ways to a much wider variety of models and problems than we have considered here. The relative simplicity and robustness of these routines should, we hope, make them amenable to such development.

Appendix A: Proof of Proposition 1

Proposition 1: *Let $\Omega \subseteq \mathbb{R}^m$ be a domain and $b : \Omega \to \mathbb{R}^h$ be analytic at the point $\overline{x} \in \Omega$. Let b_n denote the nth-order Taylor series expansion of b about the point $\overline{x}$ and let $D \subseteq \Omega$ be the domain of convergence of the Taylor series expansion. Then for any given $K \subseteq D$ compact and $\epsilon \in \mathbb{R}^+$, there exists an integer N such that $|b_n(x) - b(x)| < \epsilon$ uniformly for all $x \in K$ and all $n \geq N$.*

See Krantz (2001) for the definition of domain of convergence of a power series in the multivariable context. Note that the requirement that b be defined on an open connected set Ω is not in conflict with our assumption that the auxiliary scaling parameter $\sigma \in [0, 1]$, since it is trivial to extend the definition of the family of auxiliary equations F to the case $\sigma > 1$ and even to $\sigma < 0$ (where the latter case performs a reflection as well as a scaling of the shocks ε).

Proposition 1 is a trivial application of the results in Krantz (2001), or those in Ahlfors (1979) for the univariate case. To prove Proposition 1, we will make use of the following lemma:

Lemma 1: *The family of functions $\{b_n\}$ is equicontinuous on K. That is, given $\epsilon \in \mathbb{R}^+$, there exists a $\delta \in \mathbb{R}^+$ such that $|b_n(x_1) - b_n(x_2)| < \epsilon$ for all n and all $x_1, x_2 \in K$ with $|x_1 - x_2| < \delta$.*

Proof of Lemma 1: The analyticity of b implies that the partial derivatives $\partial b_n/\partial x^i \to \partial b/\partial x^i$ on D (see, e.g., Krantz (2001)), where x^i denotes the ith component of x. Moreover, $\partial b/\partial x^i$ and each $\partial b_n/\partial x^i$ are continuous, hence bounded on K. It follows that the derivatives of the $\{b_n\}$ are uniformly bounded on K; that is, there exists an $M \in \mathbb{R}$ such that $|\partial b_n/\partial x^i(x)| < M$ for all n, all i, and all $x \in K$.

Then $|b_n(x_1) - b_n(x_2)| < M|x_1 - x_2|$ for all n and all $x_1, x_2 \in K$. Choosing $\delta < \epsilon/M$ completes the proof. □

Proof of Proposition 1: By equicontinuity of the $\{b_n\}$ and uniform continuity of b on K, we may choose $\delta > 0$ such that $|x_1 - x_2| < \delta$ implies $|b(x_1) - b(x_2)| < \epsilon/3$ and $|b_n(x_1) - b_n(x_2)| < \epsilon/3$ for all n and all $x_1, x_2 \in K$.

For every $x \in K$, let $B(x, \delta)$ denote the δ-ball $\{y \in D : |y - x| < \delta\}$. The set of δ-balls $B(x, \delta)$, $x \in K$, form an open cover of K; choose a finite sub-cover given by $B(x_i, \delta)$, $i = 1, \ldots, M$. For each $i = 1, \ldots, M$, there exists an integer N_i such that $|b_n(x_i) - b(x_i)| < \epsilon/3$ for all $n \geq N_i$. Define $N = \max\{N_i\}$.

Now, consider $x \in K$. The point x lies in $B(x_i, \delta)$ for some i, call it i_0. Then, for $n \geq N$, we have $|b_n(x) - b(x)| \leq |b_n(x) - b_n(x_{i_0})| + |b_n(x_{i_0}) - b(x_{i_0})| + |b(x_{i_0}) - b(x)| < \epsilon$. □

Appendix B: PerturbationAIM Source Code

```
Print["This is Perturbation AIM Simple Version 2.4ets"] ;
(*                                                                      *)
(* Credits: original algorithm designed by Gary Anderson, 2002-3;       *)
(*   corrected, optimized, and recoded by Eric Swanson, 2004.           *)
(* Code is currently maintained by Eric Swanson, see                    *)
(*   http://www.ericswanson.pro for most recent version and for         *)
(*   background, instructions, capabilities, examples, and tips.        *)
(* This code will only work with Mathematica version 5.  While it is not *)
(*   too hard to modify the code to work in earlier versions of         *)
(*   Mathematica, those earlier versions are *much* slower at solving   *)
(*   linear systems of equations, and a few of the function calls are also *)
(*   a bit more awkward.                                                *)
(* This is the "simple" version of the code.  A more complicated but    *)
(*   faster and more numerically stable version of the code is available *)
(*   for download from the web site above.  This "simple" version should *)
(*   be completely adequate for most purposes, however, and will be much *)
(*   easier for the user to understand and even tailor to his/her       *)
(*   individual needs.                                                  *)
(*                                                                      *)

SetOptions["stdout",PageWidth->79] ;             (* adjust this as you like *)

(* Load two standard Mathematica packages: *)

<<DiscreteMath`Combinatorica`        (* we'll use the Compositions function *)
<<LinearAlgebra`MatrixManipulation`   (* we'll use the BlockMatrix function *)

(* The AIMZeroTol parameter should be thought of as an "economic zero."  *)
(*   In other words, AIMZeroTol should be set to the smallest possible   *)
(*   value that a coefficient in the model (either in inputs or outputs) *)
(*   could take on and still have economic meaning (as opposed to being  *)
(*   simply an artifact of finite-precision machine arithmetic).         *)
(* The default value of AIMZeroTol is 10^-10, but you can change it and  *)
(*   in fact should play around with it for any given model to test the  *)
(*   numerical stability of the solution to the model.  Note that when   *)
(*   working with arbitrary-precision numbers in Mathematica, you have   *)
(*   more freedom to set this parameter to smaller values, but you should *)
(*   still think of it as being an "economic zero" rather than a "machine *)
(*   numerical zero" (the latter is left to the internals of Mathematica *)
(*   to handle appropriately).  Thus, values like 10^-50 are probably    *)
(*   unnecessarily small and will only require you to input and compute  *)
(*   more digits of numerical precision than is really worthwhile.       *)

AIMZeroTol=10^-10 ;

(* Useful utilities that we will repeatedly call.  The syntax           *)
(*  FunctionName[vars]:= FunctionName[vars] = ...                       *)
(*  tells Mathematica to remember the answer, so that it does not have  *)
(*  to be recomputed every time the function is called.                 *)

AIMGenericArgs[eqns_]:= AIMGenericArgs[eqns] = Table[Unique["arg"],{AIMNArgs[eqns]}] ;

AIMLagVars[eqns_]:= AIMLagVars[eqns] =
Flatten[Map[Table[#[t+i],{i,Min[Cases[eqns,#[t+j_.]->j,Infinity]],-1}]&, AIMVarNames[eqns]]] ;

AIMMaxLag[eqns_]:= AIMMaxLag[eqns] = -Min[Cases[eqns,x_Symbol[t+i_.]->i,Infinity]] ;

AIMMaxLead[eqns_]:= AIMMaxLead[eqns] = Max[Cases[eqns,x_Symbol[t+i_.]->i,Infinity]] ;

AIMNArgs[eqns_]:= AIMNArgs[eqns] = Length[AIMStateVars[eqns]] ;

AIMNEqns[eqns_]:= AIMNEqns[eqns] = Length[Flatten[{eqns}]] ;

AIMNVars[eqns_]:= AIMNVars[eqns] = Length[AIMVarNames[eqns]] ;

AIMShocks[eqns_]:= AIMShocks[eqns] = Union[Cases[eqns,eps[x_][t],Infinity]] ;

AIMSSSubs={Sigma->0, eps[_][_]->0, x_[t+_.]:>Symbol[SymbolName[x]<>"AIMSS"]}
```

```
AIMSSVars[eqns_]:= AIMSSVars[eqns] = Map[Symbol[SymbolName[#]<>"AIMSS"]&, AIMVarNames[eqns]] ;

AIMStateVars[eqns_]:= AIMStateVars[eqns] = Join[AIMLagVars[eqns], AIMShocks[eqns], {Sigma}] ;

AIMVarNames[eqns_]:= AIMVarNames[eqns] = Union[Cases[eqns,x_Symbol[t+i_.]->x,Infinity]] ;

(* Calculate the steady state *)

AIMSS[eqns_,___,opts___Rule]:= AIMSS[eqns,AIMZeroTol] =
Module[{ssguess,precision,sseqns,symbols,ss},
 ssguess = AIMSSGuess /.{opts} /.AIMSSGuess->Table[0,{AIMNVars[eqns]}] ;
 precision = AIMPrecision /.{opts} /.AIMPrecision->MachinePrecision ;
 sseqns = Thread[(eqns /.Equal->Subtract /.AIMSSSubs)==0] ;
 AIMModelDiagnostics[eqns] ;
 Print["Finding steady state, AIMSSGuess->",InputForm[ssguess]] ;
 symbols = Complement[Union[Cases[sseqns,_Symbol,Infinity]], Join[AIMSSVars[eqns],{E}]] ;
 If[Length[symbols]>0,Print["Warning: found symbols ",symbols," in equations"]];
 ss = Chop[FindRoot[sseqns, Transpose[{AIMSSVars[eqns],ssguess}],
 MaxIterations->1000, WorkingPrecision->precision], AIMZeroTol] ;
 If[Head[ss]===FindRoot, $Failed, ss]
] ;

(* Front end that does error-checking and formats the output *)

AIMSeries[eqns_,deg_Integer]:=
Module[{soln,const,argsubs},
 If[AIMSS[eqns,AIMZeroTol] === $Failed || (soln=AIMSoln[eqns,deg,AIMZeroTol]) === $Failed, $Failed,
 Print["Formatting output, time is ", Date[]] ;
 const = AIMSSVars[eqns] /.AIMSS[eqns,AIMZeroTol] ;
 argsubs = Thread[AIMGenericArgs[eqns]->AIMStateVars[eqns]] ;
 Thread[Through[AIMVarNames[eqns][t]] == const + (soln /.argsubs)]
]] ;

(* Front end for Linear AIM *)

AIMSoln[eqns_,1,zerotol_]:= AIMSoln[eqns,1,zerotol] =
Module[{eqnseq0,allvars,hmat,epsmat,soln,cofb,s0inv,alllagvars},
 eqnseq0 = Flatten[{eqns}] /.Equal->Subtract ;
 allvars = Flatten[Map[Through[AIMVarNames[eqns] [t+#]]&,
  Range[-Max[AIMMaxLag[eqns],1], AIMMaxLead[eqns]]]] ;
 hmat = Outer[D,eqnseq0,allvars] /.AIMSSSubs /.AIMSS[eqns,zerotol] ;
 epsmat = Outer[D,eqnseq0,AIMShocks[eqns]] /.AIMSSSubs /.AIMSS[eqns,zerotol] ;
 If[(soln=AIMLinearSoln[hmat,AIMMaxLead[eqns]]) === $Failed, $Failed,
 {cofb,s0inv} = soln ;
 alllagvars = allvars[[Range[Max[AIMMaxLag[eqns],1]*AIMNVars[eqns]]]] ;
 Chop[cofb .alllagvars - s0inv .epsmat .AIMShocks[eqns] /.
 Thread[AIMStateVars[eqns]->AIMGenericArgs[eqns]], zerotol]
]] ;

(* This is the heart of the program.  Derivatives are evaluated at steady   *)
(*  state, the expectation is taken, and coefficients are solved using      *)
(*  the method of undetermined coefficients.                                *)
(* The following two tricks are not strictly necessary and bloat the code   *)
(*  a bit, but increase speed and, more importantly, seem to reduce the     *)
(*  likelihood of numerical inaccuracies:                                   *)
(* 1. Solve a certainty-equivalent version of the problem first (Sigma=0).  *)
(* 2. Impose constraint that all terms linear in Sigma (even if nonlinear   *)
(*     in other variables) have zero coefficients, which is fairly easy to  *)
(*     show mathematically.                                                 *)

AIMSoln[eqns_,deg_Integer,zerotol_]:= AIMSoln[eqns,deg,zerotol] =
Module[{args,drvindxs,cedrvindxs,stdrvindxs,cecoeffs,stcoeffs,nextceTerms,
 nextstTerms,bsubs,ssderivs,cesystem,cesoln,dum,dropce,stsystem,stsoln},
 args = AIMGenericArgs[eqns] ;
 drvindxs = Compositions[deg,AIMNArgs[eqns]] ;
  cedrvindxs = Select[drvindxs, #[[-1]]==0 &] ;
  stdrvindxs = Select[drvindxs, #[[-1]] >1 &] ;
 cecoeffs = Table[Unique[],{AIMNVars[eqns]},{Length[cedrvindxs]}] ;
  stcoeffs = Table[Unique[],{AIMNVars[eqns]},{Length[stdrvindxs]}] ;
 nextceTerms = cecoeffs .Map[Apply[Times,Power[args,#]]&, cedrvindxs] ;
```

```
  nextstTerms = stcoeffs .Map[Apply[Times,Power[args,#]]&, stdrvindxs] ;
 bsubs = Thread[Map[bFunc,AIMVarNames[eqns]] -> Map[Apply[Function,
  {args,#}]&, AIMSoln[eqns,deg-1,zerotol] + nextceTerms + nextstTerms]] ;
 Print["Differentiating eqns, time is ", Date[]] ;
 ssderivs = Chop[AIMDerivatives[eqns,deg][0] /.bsubs, zerotol] ;
 Print["Undetermined coefficients to solve: ", AIMNVars[eqns] *(Length[cedrvindxs]+Length[stdrvindxs])] ;
 Print["Calculating CE solution, time is ", Date[]] ;
 cesoln = If[AIMNArgs[eqns]===1, PrependTo[args,dum]; {},
   cesystem = Flatten[Chop[Take[CoefficientArrays[ssderivs /.args[[-1]]->0, Drop[args,-1]],-1],zerotol]];
   Chop[Flatten[NSolve[cesystem, Flatten[cecoeffs]]], zerotol]] ;
 Print["Calculating Stoch solution, time is ", Date[]] ;
 dropce = Flatten[Drop[CoefficientArrays[ssderivs, args[[-1]]], 2]] ;
 stsystem = Chop[Expand[Flatten[CoefficientArrays[dropce, Drop[args,-1]]]] /.
  eps[x_][_]^n_->mom[x,n] /.eps[_][_]->0 /.cesoln, zerotol] ;
 stsoln = Chop[Flatten[NSolve[stsystem, Flatten[stcoeffs]]], zerotol] ;
 AIMSoln[eqns,deg-1,zerotol] + (nextceTerms /.cesoln) + (nextstTerms /.stsoln)
] ;

(* That's essentially it.  The following routine calculates derivatives     *)
(*   of the equations composed with the (unknown) solution functions b.     *)
(* The trick of using univariate differentiation to calculate all the       *)
(*   multivariate derivatives of Fob speeds up the code considerably; in    *)
(*   particular, variations on the obvious Outer[D,eqns,vars] are *much*    *)
(*   slower.                                                                *)

AIMDerivatives[eqns_,0]:=
Function[tAIM, Evaluate[AIMSubBFuncsIntoEqns[eqns] /.
 Thread[AIMStateVars[eqns]->tAIM*AIMGenericArgs[eqns]]]] ;

AIMDerivatives[eqns_,deg_Integer]:= AIMDerivatives[eqns,deg] =
AIMDerivatives[eqns,deg-1]' ;

AIMSubBFuncsIntoEqns[eqns_]:= AIMSubBFuncsIntoEqns[eqns] =
With[{deveqns = eqns /.x_Symbol[t+i_.]:>Symbol[SymbolName[x]<>"AIMSS"] +x[t+i]
 /.Equal->Subtract /.AIMSS[eqns,AIMZeroTol]},
 AIMSubOutLeadVars[eqns,deveqns,AIMMaxLead[eqns]]
] ;

AIMSubOutLeadVars[origeqns_,eqnssofar_,0]:=
eqnssofar /.x_Symbol[t]->Apply[bFunc[x],AIMStateVars[origeqns]] /.eps[x_][t+i_]->Sigma*eps[x][t+i]

AIMSubOutLeadVars[origeqns_,eqnssofar_,lead_Integer]:=
With[{fwdsv=AIMStateVars[origeqns] /.t->t+lead},
With[{reducedeqns=eqnssofar /.x_Symbol[t+lead]->Apply[bFunc[x],fwdsv]},
 AIMSubOutLeadVars[origeqns, reducedeqns, lead-1]
]] ;  (* note how the use of recursion makes handling multiple leads simple *)

(* Print out model diagnostics (obviously) *)

AIMModelDiagnostics[eqns_] := (
Print["\nModel Diagnostics:"] ;
Print["Number of equations: ",AIMNEqns[eqns]] ;
Print["Number of variables: ",AIMNVars[eqns]] ;
Print["Number of shocks:    ",Length[AIMShocks[eqns]]] ;
Print["Maximum lag:         ",AIMMaxLag[eqns]] ;
Print["Maximum lead:        ",AIMMaxLead[eqns]] ;
Print["Lagged variables: ",AIMLagVars[eqns]] ;
Print["Shocks: ",AIMShocks[eqns]] ;
Print[" together with Sigma, these yield ",AIMNArgs[eqns]," state variables\n"];
Print["List of all variables: ",AIMVarNames[eqns]] ;
Print["Treating steady state and final coeff values < ", N[AIMZeroTol], " as zero (AIMZeroTol)\n"] ;
) ;

(* Everything that follows is Linear AIM.  See Anderson and Moore (1985)    *)
(*   for a description of the algorithm.  There are ways to improve the     *)
(*   speed and numerical stability of this implementation for larger        *)
(*   models, at the cost of complicating the code.  In particular, one can: *)
(* 1. Implement "exact" ShiftRights to reduce the number of calls to the    *)
(*     singular value decomposition.                                        *)
(* 2. Use a Schur decomposition instead of eigenvectors to compute the      *)
```

```
(*     left invariant subspace corresponding to the large eigenvalues of    *)
(*     AR1Form[shiftedh].  (This requires a user-written routine, because   *)
(*     Mathematica's Schur implementation does not sort eigenvalues into    *)
(*     any order.)                                                           *)
(* These improvements are implemented in the more advanced version of the   *)
(*   code.                                                                   *)

AIMLinearSoln[hmat_,nleads_Integer] :=
Module[{hrows,hcols,shiftedh,qmat,ar1eigvals,stabconds,bmat,smat},
 {hrows,hcols} = Dimensions[hmat] ;
 {shiftedh,qmat} = NestWhile[AIMShiftRight, {hmat, {}},
 MatrixRank[AIMLastBlock[#[[1]]]] <hrows &, 1, nleads*hrows] ;
 Print["Lead matrix is full rank; computing stability conditions"] ;
 ar1eigvals = Eigenvalues[AIMAR1Form[shiftedh]] ;
 stabconds = Chop[Eigenvectors[Transpose[AIMAR1Form[shiftedh]],
 Length[Cases[Abs[ar1eigvals],i_/;i>1+AIMZeroTol]]], AIMZeroTol];
 Print["Model has ",Length[stabconds], " unstable roots"] ;
 qmat = Join[qmat,stabconds] ;
 If[Length[qmat]<hrows*nleads, Print["Multiple Linear Solutions"]; $Failed,
 If[Length[qmat]>hrows*nleads || MatrixRank[AIMLastBlock[qmat]]<hrows*nleads,
                                    Print["No Linear Solutions"]; $Failed,
 bmat = LinearSolve[AIMLastBlock[qmat],-AIMFirstBlocks[qmat]] ;
 smat = hmat .BlockMatrix[{{IdentityMatrix[hcols-hrows*nleads]},
 {ZeroMatrix[hrows*nleads,hrows],bmat}}] ;
 Chop[{Take[bmat,hrows], Inverse[AIMLastBlock[smat]]}, AIMZeroTol]
]]] ;

AIMLinearSoln[hmat_,0] :=
With[{cofb=LinearSolve[AIMLastBlock[hmat],-AIMFirstBlocks[hmat]]},
 Chop[{cofb, Inverse[AIMLastBlock[hmat]]}, AIMZeroTol]
] ;

AIMShiftRight[{hmatold_,qmatsofar_}]:=
Module[{hrows=Length[hmatold],svdu,svdsig,hmatnew,zerorows,firstblocks},
 {svdu,svdsig} = Take[SingularValueDecomposition[AIMLastBlock[hmatold]],2] ;
 hmatnew = Transpose[svdu] .hmatold ;
 zerorows = Map[List,Range[MatrixRank[svdsig]+1,hrows]] ;
 Print["Shifting ",Length[zerorows]," linear combinations of equations forward"];
 firstblocks = AIMFirstBlocks[hmatnew] ;
 Chop[{ReplacePart[hmatnew,BlockMatrix[{{ZeroMatrix[hrows,hrows],firstblocks}}],zerorows,zerorows],
   Join[qmatsofar,firstblocks[[Flatten[zerorows]]]]}, AIMZeroTol]
] ;

AIMAR1Form[shiftedh_]:= AIMAR1Form[shiftedh] =
With[{hrows=Length[shiftedh],hcols=Length[shiftedh[[1]]]},
 Chop[BlockMatrix[{{ZeroMatrix[hcols-2*hrows,hrows], IdentityMatrix[hcols-2*hrows]},
 {LinearSolve[AIMLastBlock[shiftedh],-AIMFirstBlocks[shiftedh]]}}],
 AIMZeroTol]
] ;

AIMLastBlock[matrix_]:=
With[{dims=Dimensions[matrix]},
 SubMatrix[matrix,{1,dims[[2]]-dims[[1]]+1},{dims[[1]],dims[[1]]}]
] ;

AIMFirstBlocks[matrix_]:=
With[{dims=Dimensions[matrix]},
 SubMatrix[matrix,{1,1},{dims[[1]],dims[[2]]-dims[[1]]}]
] ;
```

Appendix C: Detailed Solution to the Stochastic Growth Model

The third-order solution to equations (6)–(12) in the text is provided below. "Inv" is used instead of "I" (to avoid confusion with the imaginary number i) and "eps[a]" instead of ε. The variables A, C, K, and Y are transformed into natural logarithms before the approximation is computed. Parameter values are $\alpha = .3$, $\beta = .99$, $\gamma = 1.1$, $\delta = .025$, and $\rho = .8$. Variables on the right-hand side of each equation denote deviations from steady state (A, C, K, and Y denote log-deviations from steady state). Variables on the left-hand side of each equation are in levels (log-levels for A, C, K, and Y). The term mom[a,n] refers to the (known) nth moment of ε. The output below is in standard PerturbationAIM output format. Coefficients are accurate to at least 42 decimal places (see table 1), but have only been reported to 6 significant digits to save space.

```
A[t] == 0.800000 A[-1 + t] + 1.00000 eps[a][t],

C[t] == 0.679145 + 0.128223 A[-1 + t] + 0.0242775 A[-1 + t]^2 + 0.00323546 A[-1 + t]^3 +
    0.538516 K[-1 + t] - 0.0563800 A[-1 + t] K[-1 + t] - 0.00802382 A[-1 + t]^2 K[-1 + t] +
    0.0252054 K[-1 + t]^2 + 0.00902121 A[-1 + t] K[-1 + t]^2 + 0.000147704 K[-1 + t]^3 +
    0.263256 Sigma^2 mom[a,2] + 0.0298985 Sigma^2 A[-1 + t] mom[a,2] +
    0.0997791 Sigma^2 K[-1 + t] mom[a,2] - 0.0230988 Sigma^3 mom[a,3] + 0.160279 eps[a][t] +
    0.0606937 A[-1 + t] eps[a][t] + 0.0121330 A[-1 + t]^2 eps[a][t] -
    0.0704750 K[-1 + t] eps[a][t] - 0.0200595 A[-1 + t] K[-1 + t] eps[a][t] +
    0.0112765 K[-1 + t]^2 eps[a][t] + 0.0373731 Sigma^2 mom[a,2] eps[a][t] +
    0.0379335 eps[a][t]^2 + 0.0151662 A[-1 + t] eps[a][t]^2 - 0.0125372 K[-1 + t] eps[a][t]^2 +
    0.00631925 eps[a][t]^3,

Inv[t] == 0.535902 + 1.75359 A[-1 + t] + 0.738497 A[-1 + t]^2 + 0.200811 A[-1 + t]^3 -
    0.309629 K[-1 + t] + 0.576954 A[-1 + t] K[-1 + t] + 0.236344 A[-1 + t]^2 K[-1 + t] -
    0.222813 K[-1 + t]^2 + 0.0893370 A[-1 + t] K[-1 + t]^2 - 0.0671071 K[-1 + t]^3 -
    0.519191 Sigma^2 mom[a,2] - 0.125538 Sigma^2 A[-1 + t] mom[a,2] -
    0.476376 Sigma^2 K[-1 + t] mom[a,2] + 0.0455553 Sigma^3 mom[a,3] + 2.19199 eps[a][t] +
    1.84624 A[-1 + t] eps[a][t] + 0.753040 A[-1 + t]^2 eps[a][t] +
    0.721193 K[-1 + t] eps[a][t] + 0.590860 A[-1 + t] K[-1 + t] eps[a][t] +
    0.111671 K[-1 + t]^2 eps[a][t] - 0.156922 Sigma^2 mom[a,2] eps[a][t] +
    1.15390 eps[a][t]^2 + 0.941300 A[-1 + t] eps[a][t]^2 + 0.369288 K[-1 + t] eps[a][t]^2 +
    0.392208 eps[a][t]^3,

K[t] == 3.06508 + 0.0818058 A[-1 + t] + 0.0311051 A[-1 + t]^2 + 0.00673207 A[-1 + t]^3 +
    0.960556 K[-1 + t] - 0.0516639 A[-1 + t] K[-1 + t] - 0.0178403 A[-1 + t]^2 K[-1 + t] +
    0.0157721 K[-1 + t]^2 + 0.0147636 A[-1 + t] K[-1 + t]^2 - 0.00349273 K[-1 + t]^3 -
    0.0242205 Sigma^2 mom[a,2] - 0.00387500 Sigma^2 A[-1 + t] mom[a,2] +
    0.00104197 Sigma^2 K[-1 + t] mom[a,2] + 0.00212517 Sigma^3 mom[a,3] + 0.102257 eps[a][t] +
    0.0777626 A[-1 + t] eps[a][t] + 0.0252453 A[-1 + t]^2 eps[a][t] -
    0.0645798 K[-1 + t] eps[a][t] - 0.0446008 A[-1 + t] K[-1 + t] eps[a][t] +
    0.0184545 K[-1 + t]^2 eps[a][t] - 0.00484375 Sigma^2 mom[a,2] eps[a][t] +
    0.0486017 eps[a][t]^2 + 0.0315566 A[-1 + t] eps[a][t]^2 - 0.0278755 K[-1 + t] eps[a][t]^2 +
    0.0131486 eps[a][t]^3,

r[t] == 0.0101010 + 0.0280808 A[-1 + t] + 0.0112323 A[-1 + t]^2 + 0.00299529 A[-1 + t]^3 -
   0.0245707 K[-1 + t] - 0.0196566 A[-1 + t] K[-1 + t] - 0.00786263 A[-1 + t]^2 K[-1 + t] +
   0.00859975 K[-1 + t]^2 + 0.00687980 A[-1 + t] K[-1 + t]^2 - 0.00200661 K[-1 + t]^3 +
   0.0351010 eps[a][t] + 0.0280808 A[-1 + t] eps[a][t] + 0.0112323 A[-1 + t]^2 eps[a][t] -
   0.0245707 K[-1 + t] eps[a][t] - 0.0196566 A[-1 + t] K[-1 + t] eps[a][t] +
   0.00859975 K[-1 + t]^2 eps[a][t] + 0.0175505 eps[a][t]^2 + 0.0140404 A[-1 + t] eps[a][t]^2 -
   0.0122854 K[-1 + t] eps[a][t]^2 + 0.00585017 eps[a][t]^3,

Welf[t] == -934.340 + 4.57011 A[-1 + t] + 0.752082 A[-1 + t]^2 + 0.0961280 A[-1 + t]^3 +
   10.2581 K[-1 + t] - 1.16168 A[-1 + t] K[-1 + t] - 0.0980612 A[-1 + t]^2 K[-1 + t] +
   1.96600 K[-1 + t]^2 - 0.000889657 A[-1 + t] K[-1 + t]^2 + 0.184489 K[-1 + t]^3 +
   116.338 Sigma^2 mom[a,2] + 0.499460 Sigma^2 A[-1 + t] mom[a,2] -
   2.97055 Sigma^2 K[-1 + t] mom[a,2] + 18.5872 Sigma^3 mom[a,3] + 5.71263 eps[a][t] +
   1.88020 A[-1 + t] eps[a][t] + 0.360480 A[-1 + t]^2 eps[a][t] -
   1.45210 K[-1 + t] eps[a][t] - 0.245153 A[-1 + t] K[-1 + t] eps[a][t] -
   0.00111207 K[-1 + t]^2 eps[a][t] + 0.624325 Sigma^2 mom[a,2] eps[a][t] +
   1.17513 eps[a][t]^2 + 0.450600 A[-1 + t] eps[a][t]^2 - 0.153221 K[-1 + t] eps[a][t]^2 +
   0.187750 eps[a][t]^3,

Y[t] == 0.919523 + 0.800000 A[-1 + t] + 0.300000 K[-1 + t] + 1.000000 eps[a][t]
```

References

Ahlfors, Lars. *Complex Analysis* (New York: McGraw-Hill, 1979).

Anderson, Gary. "Symbolic Algebra Programming for Analyzing the Long Run Dynamics of Economic Models" in *Economic and Financial Modeling with Mathematica*, ed. Hal Varian (Springer-Verlag: New York, 1993), 124–47.

Anderson, Gary. "A Systematic Comparison of Linear Rational Expectation Model Solution Algorithms," unpublished manuscript, Federal Reserve Board (2001).

Anderson, Gary and George Moore. "A Linear Algebraic Procedure for Solving Linear Perfect Foresight Models," *Economics Letters* 17 (1985), 247–52.

Aruoba, S. Boragan, Jesus Fernandez-Villaverde, and Juan Rubio-Ramirez. "Comparing Solution Methods for Dynamic Equilibrium Economies," unpublished manuscript, University of Pennsylvania (2005).

Blanchard, Olivier and Charles Kahn. "The Solution of Linear Difference Models under Rational Expectations," *Econometrica* 48 (1980), 1305–11.

Collard, Fabrice and Michel Juillard. "Perturbation Methods for Rational Expectations Models," unpublished manuscript, CEPREMAP (2001).

Fleming, Wendell. "Stochastic Control for Small Noise Intensities," *SIAM Journal of Control* 9 (1971), 473–517.

Fuhrer, Jeffrey and Brian Madigan. "Monetary Policy When Interest Rates Are Bounded at Zero," *Review of Economics and Statistics* 79 (1997), 573–85.

Gaspar, Jess and Kenneth Judd. "Solving Large-Scale Rational-Expectations Models," *Macroeconomic Dynamics* 1 (1997), 45–75.

Jin, He-Hui and Kenneth Judd. "Perturbation Methods for General Dynamic Stochastic Models," unpublished manuscript, Hoover Institute (2002).

Judd, Kenneth. *Numerical Methods for Economists* (MIT Press: Cambridge, MA, 1999).

Kim, Jinill, and Sunghyun Kim. "Spurious Welfare Reversals in International Business Cycle Models," *Journal of International Economics* (forthcoming).

Kim, Jinill, Sunghyun Kim, Ernst Schaumberg, and Christopher Sims. "Second Order Accurate Solution of Discrete Time Dynamic Equilibrium Models," unpublished manuscript, Princeton Unviersity (2002).

Kollmann, Robert. "Monetary Policy Rules in the Open Economy: Effects on Welfare and Business Cycles," *Journal of Monetary Economics* 49 (2002), 989–1015.

Kollmann, Robert. "Monetary Policy Rules in an Interdependent World," *CEPR Discussion Paper* 4012 (2003).

Krantz, Steven. *Function Theory of Several Complex Variables* (Providence, Rhode Island: AMS Chelsea, 2001).

Schmitt-Grohe, Stephanie and Martin Uribe. "Solving Dynamic General Equilibrium Models Using a Second-Order Approximation of the Policy Function," *Journal of Economic Dynamics and Control* 28 (2004), 755–75.

Sims, Christopher. "Solving Linear Rational Expectations Models," unpublished manuscript, Princeton University (2000).

Woodford, Michael. *Interest and Prices* (Princeton: Princeton University Press, 2003).