Numerically Stable Stochastic Simulation Approaches for Solving Dynamic Economic Models*

Kenneth Judd, Lilia Maliar and Serguei Maliar

August 26, 2009

Abstract

We develop numerically stable stochastic simulation approaches for solving dynamic economic models. We rely on standard simulation procedures to simultaneously compute an ergodic distribution of state variables, its support and the associated decision rules. We differ from existing methods, however, in how we use simulation data to approximate decision rules. Instead of the usual least-squares approximation methods, we examine a variety of alternatives, including the least-squares method using SVD, Tikhonov regularization, least-absolute deviation methods, principal components regression method, all of which are numerically stable and can handle ill-conditioned problems. These new methods enable us to compute high-order polynomial approximations without encountering numerical problems. Our approaches are especially well suitable for high-dimensional applications in which other methods are infeasible.

JEL classification : C63, C68

Key Words : neoclassical stochastic growth model; stochastic simulation; parameterized expectations algorithm (PEA); least absolute deviations (LAD); linear programming; regularization

*Professor Lilia Maliar acknowledges support from the Hoover Institution at Stanford University, the Generalitat Valenciana under the grant BEST/2008/090 and the Ministerio de Educación y Ciencia de España under the grant SEJ 2007-62656 and the José Castillejo program JC2008-224. Professor Serguei Maliar acknowledges support from the Hoover Institution at Stanford University and the Ministerio de Educación y Ciencia de España under the grant SEJ 2007-62656 and the Salvador Madariaga program PR2008-190.

1 Introduction

Dynamic stochastic economic models do not generally admit closed-form solutions and should be approached with numerical methods; see Taylor and Uhlig (1990), Judd (1998), Scott and Marimon (1999), and Santos (1999) for reviews of such methods. The majority of methods for solving dynamic models fall into three broad classes: projection methods, which approximate solutions on tensor-product grids using quadrature-integration; perturbation methods, which find solutions locally using high-order Taylor expansions of optimality conditions; and stochastic simulation methods, which simultaneously compute an ergodic distribution of state variables, its support and the associated decision rules. All three classes of methods have their relative advantages and drawbacks, and the optimal choice of a method depends on the details of the application in question. Projection methods are accurate and fast when applied to models with few state variables, however, their cost increases exponentially as the number of state variables increases.[1] Perturbation methods are generally cheap, however, the range within which they are accurate is uncertain. Stochastic simulation methods can compute global approximations in high-dimensional problems at a reasonable cost, however, they are generally less accurate and less numerically stable than other methods.[2]

In this paper, we focus on the stochastic simulation class of methods.[3] We specifically attempt to enhance numerical stability and accuracy of existing stochastic simulation methods for solving dynamic economic models. Our approaches distinguish themselves from existing methods in how we use sim-

[1] The cost of projection methods can be reduced using more efficient constructions of multi-dimensional grids (e.g., the Smolyak sparse grid described in Krueger and Kubler, 2004), however such constructions cannot prevent costs from nonetheless growing quickly with the dimension of the state space.

[2] The performance of projection and perturbation methods in the context of moderately large economic models is studied in Judd and Gaspar (1997).

[3] Stochastic simulations are widely used in economics and other fields; see Asmussen and Glynn (2007) for an up-to-date review of such methods. In macroeconomic literature, stochastic simulation methods have been used to approximate an economy's path (Fair and Taylor 1983), a conditional expectation function in the Euler equation (Den Haan and Marcet, 1990), a value function (Maliar and Maliar, 2005), an equilibrium interest rate (Aiyagari, 1994), and an aggregate law of motion of a heterogeneous-agent economy (Krusell and Smith, 1998), as well as to make inferences about the parameters of economic models (Smith, 1993) among others.

ulation data to approximate decision rules: we build an approximation step on a linear regression model, and we approximate decision rules using fast and efficient linear approximation methods. Such methods are designed to handle ill-conditioned problems including an extreme case of linearly dependent variables. The use of numerically stable approximation methods enables us to compute high-order polynomial approximations without encountering numerical problems. All the approximation methods described in the paper are simple to implement. We provide a Matlab routine that can be used in other economic applications.

A stochastic simulation approach is attractive for high-dimensional applications because it allows to compute solutions only in an area of the state space which is visited in simulations. In Figure 1, we plot an ergodic distribution of capital and technology shock for a one-sector growth model with a closed-form solution (for a detailed description of this model, see Section 2). The ergodic distribution takes the form of an oval and most of the rectangular area that sits outside of the oval's boundaries is never visited. In the two-dimensional case, a circle inscribed within a square occupies about 79% of the area of the square, and an oval inscribed in this way occupies even less of the original squares's area. Thus, in the model with two state variables, the state space is reduced by at least 21%, and quite possibly even more. In an n-dimensional case, the ratio of a hypershere's volume Ω_n^s to a hypercube's volume Ω_n^c is given by the following formula:

$$\frac{\Omega_n^s}{\Omega_n^c} = \begin{cases} \frac{(\pi/2)^{\frac{n-1}{2}}}{1 \cdot 3 \cdot \ldots \cdot n} \text{ for } n = 1, 3, 5... \\ \\ \frac{(\pi/2)^{\frac{n}{2}}}{2 \cdot 4 \cdot \ldots \cdot n} \text{ for } n = 2, 4, 6... \end{cases}. \tag{1}$$

The ratio $\frac{\Omega_n^s}{\Omega_n^c}$ declines very rapidly with the dimension of the state space; see Table 1 and Figure 2. For dimensions three, four, five, ten and thirty, this ratio is 0.52, 0.31, 0.16, $3 \cdot 10^{-3}$ and $2 \cdot 10^{-14}$, respectively. Hence, when computing a solution on an ergodic hypershere distribution, we face just a tiny fraction of cost we would have faced on a tensor-product hypercube grid, used in projection methods. In particular, restricting attention to ergodic distribution allows us to avoid computing a solution in an enormously large number of points on the hypercube's edges.

However, an ergodic distribution is unknown before a model is solved. Marcet (1988) and Den Haan and Marcet (1990) propose to compute both the unknown ergodic distribution and the associated decision rules using

the following stochastic simulation procedure: guess a solution, simulate the model, then use the simulation results to update the guess, and iterate until convergence. On each iteration, this procedure generates a set of points that can be viewed as an endogenous grid. Den Haan and Marcet's (1990) method is referred to in the literature as *parameterized expectation algorithm (PEA)* because of a specific parameterization used: the conditional expectation in the Euler equation. We will call this method *simulation-based PEA* to emphasize that it belongs to the stochastic simulation class of methods.[4] Den Haan and Marcet (1990) use simulated series not only to construct an ergodic distribution but also to evaluate conditional expectations via Monte Carlo integration. The last is not a very efficient integration technique because stochastic simulations oversample the center and undersample the tails of the ergodic distribution. However, in high-dimensional applications, Monte Carlo integration can be a useful alternative to expensive quadrature integration.[5]

The main shortcoming of the simulation-based PEA, however, is not inefficiency but numerical instability. First, polynomial terms in the approximating function are highly correlated even under low-order polynomials which can often lead to a failure of the approximation step; see Den Haan and Marcet (1990) and Christiano and Fisher (2000). Second, an exponentiated polynomial approximation used in the simulation-based PEA is estimated with non-linear least-squares (NLLS) methods and such methods require supplying an initial guess and involve computing costly Jacobian and Hessian matrices; moreover, on many occasions they simply fail to converge; see Christiano and Fisher (2000).[6] To reduce the likelihood of numerical problems in the approximation step, we propose to use a linear regression model and linear approximation methods instead of the exponentiated polynomial model and NLLS methods in Den Haan and Marcet (1990).

[4]There are PEAs that do not rely on stochastic simulations (see, e.g., Christiano and Fisher, 2000, for projection PEAs), and there are stochastic simulation algorithms that do not parameterize conditional expectation functions (see, e.g., Maliar and Maliar, 2005, for a stochastic simulation algorithm parameterizing value function). A stochastic simulation approach abstracts from the specific decision function being parameterized (conditional expectation, capital, consumption, etc.) and emphasizes the use of simulations.

[5]Creel (2008) develops a parallel computing toolbox which allows to reduce the cost of Monte Carlo integration in a version of the simulation-based PEA by running simulations on a cluster of computers.

[6]Maliar and Maliar (2003a) propose a simple modification that stabilizes the simulation-based PEA by restricting simulated series to be within certain bounds.

We start from diagnosing the exact causes of the simulation-based PEA's numerical instability: specifically, we ask whether this instability is caused by specific implementation of the algorithm (i.e. specific conditional-expectation parameterization, NLLS methods used, etc.) or the intrinsic instability of the iterative learning process in dynamic economic models. We find that the primary reason for the numerical instability of the stochastic simulation algorithm is ill-conditioning of the least-squares (LS) problem that is solved in the approximation step; however, we also detect cases in which the learning process itself appears unstable. Ill-conditioning arises as a consequence of multicollinearity and poor scaling of explanatory variables, and instability of learning occurs when the parameterized decision rules do not capture the essence of the agent's choice. We explore the following five strategies designed to enhance the numerical stability of the stochastic simulation algorithm: (1) to parameterize different decision rules (capital decision rule versus marginal-utility decision rule); (2) to use different polynomial families for the approximating functions (ordinary versus Hermite polynomial representations); (3) to normalize the variables in the regression; (4) to use LS methods that are more numerically stable than the standard OLS method; and (5) to replace the ill-conditioned LS problem with some other less ill-conditioned problem. Strategy (1) helps restore the stability of the iterative learning process, whereas strategies $(2) - (4)$ help mitigate the effect of ill-conditioning on the LS problem.

In the case of strategy (4), we consider two alternatives: one is a LS method using singular value decomposition (SVD) which can stand higher degrees of ill-conditioning than OLS; the other is Tikhonov regularization which reduces degrees of the ill-conditioning of the LS problem. In our pursuit of strategy (5), we propose to perform the approximation step using a least-absolute deviations (LAD) problem which is generally not so ill-conditioned as the LS problem. The LAD problem can be cast into the standard linear programming form and solved with linear programming methods. We formulate a primal and dual representations of the LAD problem that are suitable for our approximation step, and we subsequently develop regularized versions of the primal and dual problems, which are particularly robust and numerically stable. We next show that our numerically stable methods can be also used to stabilize non-linear regression models including the regression with exponentiated polynomials advocated in the previous literature. We finally merge the proposed approximation methods and the principal component analysis into a unified approach that can handle problems with any degree of

ill-conditioning including the case of perfectly collinear explanatory variables.

Most of the approximation methods described in the paper are drawn from the fields of applied mathematics and statistics. We apply those approximation methods in a novel way by bringing them into an iterative process that solves dynamic economic models. In this context, we are the first to use the LAD approach and to show that a dual formulation of the LAD problem saves cost and memory relative to the primal formulation. Our findings suggest that the LAD approach can be a useful alternative to the standard LS approach in economic applications.

Some of the methods we present in the paper are new to the literature. First, we develop primal and dual LAD regularization methods that have advantages compared to the existing LAD regularization method by Wang, Gordon and Zhu (2006). Second, we propose a non-linear LAD method that can be used as an alternative to NLLS methods, and we show how to cast such a method into a linear programming form. Finally, we combine the principal component analysis with the linear and non-linear regularization methods into a unified numerically stable approach.

We test the successfulness of the proposed strategies $(1)-(5)$ by applying them to a one-sector neoclassical stochastic growth model. For the version of the model with a closed-form solution, we observe that when the OLS method is used with unnormalized ordinary polynomials, the stochastic simulation algorithm cannot go beyond a second-order polynomial approximation, and the Euler equation errors are of order 10^{-6}. Normalization of variables alone allows us to move to the third-order polynomial approximation and to reach the accuracy of order 10^{-7}. Under the dual formulation of the LAD problem and the normalized variables, we can obtain the fourth-order polynomial approximation and achieve a level of accuracy that is of order 10^{-8}. The introduction of the Hermite polynomials allow us to compute the polynomial approximation of all five orders and to achieve the accuracy of order 10^{-9}, as do all other considered methods, including LS using SVD, Tikhonov regularization, principal component method, and primal and dual LAD regularization methods. In more general versions of the model without closed-form solutions, we do not achieve such a remarkable accuracy because the solutions are close to linear and high-order polynomial terms do little to improve accuracy. However, our stochastic simulation algorithm is still stable and is able to compute polynomial approximations up to the fifth order, except in the model with a highly risk averse agent in which the parameterized capital decision rule leads to an unstable learning process. We finally compare the

polynomial and exponentiated polynomial specifications, and we find that both specifications lead to very similar results. Thus, replacing Den Haan and Marcet's (1990) exponentiated polynomials with simple polynomials has no detrimental effect on a method's accuracy.

Although we develop the numerically stable approximation methods in the context of a specific stochastic simulation algorithm, such methods can improve stability and accuracy of any stochastic simulation algorithm that recovers unknown functions from the data. For example, our approximation methods can be useful for implementing the regression step in Krusell and Smith's (1998) algorithm if the LS problem is ill-conditioned (this can be the case if an aggregate law of motion is approximated not only in terms of the first and second but also in terms of higher moments of wealth distribution).

The rest of the paper is organized as follows: In Section 2, we describe stochastic simulation algorithm in the context of the standard neoclassical stochastic growth model. In Section 3, we compare linear and non-linear regression models for approximating decision functions. In Section 4, we discuss the origins and consequences of the ill-conditioning of the LS problem. In Section 5, we elaborate on potential strategies for enhancing the numerical stability of stochastic simulation algorithms. In Section 6, we merge the previously described approximation methods with the principal component analysis. In Section 7, we present quantitative results. Finally, in Section 8, we conclude.

2 Stochastic simulation algorithm

We apply the stochastic simulation algorithm for finding a solution to the standard one-sector neoclassical stochastic growth model. The social planner in this model solves

$$\max_{\{k_{t+1},c_t\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \delta^t u\left(c_t\right) \tag{2}$$

subject to the budget constraint,

$$c_t + k_{t+1} = \left(1-d\right) k_t + \theta_t f\left(k_t\right), \tag{3}$$

and to the process for technology shocks,

$$\ln \theta_t = \rho \ln \theta_{t-1} + \epsilon_t, \qquad \epsilon_t \sim N\left(0, \sigma^2\right), \tag{4}$$

where initial condition (k_0, θ_0) is given. Here, E_t is the operator of conditional expectation; c_t, k_t and θ_t are, respectively, consumption, capital and technology shock; u and f are, respectively, the utility and production functions, both of which are assumed to be strictly increasing and concave; $\delta \in (0,1)$ is the discount factor; $d \in (0,1]$ is the depreciation rate of capital; and $\rho \in (-1,1)$ and $\sigma \geq 0$ are the autocorrelation coefficient and the standard deviation of the technology shock, respectively.

Under the assumption of an interior solution, the Euler equation of the problem $(2)-(4)$ is

$$u'(c_t) = \delta E_t \left\{ u'(c_{t+1}) \left[1 - d + \theta_{t+1} f'(k_{t+1}) \right] \right\}. \tag{5}$$

We restrict attention to a first-order (Markov) equilibrium where decision rules in period t are functions of current state (k_t, θ_t). Our objective is to find decision rules for capital, $k_{t+1} = K(k_t, \theta_t)$, and consumption, $c_t = C(k_t, \theta_t)$, such that for any sequence of shocks $\{\theta_t\}_{t=0}^{\infty}$ generated by the process (4), these rules yield time series $\{k_{t+1}, c_t\}_{t=0}^{\infty}$ that satisfy the Euler equation (5) and the budget constraint (3).

In the benchmark case, we parameterize the capital decision rule $k_{t+1} = K(k_t, \theta_t)$. We specifically pre-multiply both sides of the Euler equation (5) by k_{t+1} to obtain

$$k_{t+1} = E_t \left\{ \delta \frac{u'(c_{t+1})}{u'(c_t)} \left[1 - d + \theta_{t+1} f'(k_{t+1}) \right] k_{t+1} \right\} \simeq \Psi(k_t, \theta_t; \beta), \tag{6}$$

where $\Psi(k_t, \theta_t; \beta)$ is a flexible functional form that depends on a vector of coefficients β. Our objective is to find β such that $\Psi(k_t, \theta_t; \beta)$ is the best possible approximation of the capital decision rule for the given functional form Ψ.

We compute β by using the following stochastic simulation algorithm.

- Step 1. Choose a length of simulations T. Fix initial condition (k_0, θ_0). Draw a sequence of shocks $\{\theta_t\}_{t=1}^{T}$ using the process given in (4) and fix this sequence for all simulations.

- Step 2. On iteration j, fix some vector of coefficients $\beta^{(j)}$ and use the assumed decision rule $\Psi\left(k_t, \theta_t; \beta^{(j)}\right)$ from (6) to simulate forward $\{k_{t+1}\}_{t=0}^{T}$ for a given sequence of shocks $\{\theta_t\}_{t=0}^{T}$.

- Step 3. Restore the sequence of consumption $\{c_t\}_{t=0}^{T}$ from the budget constraint (3) and compute a variable inside the conditional expectation, such that

$$y_t \equiv \delta \frac{u'(c_{t+1})}{u'(c_t)} \left[1 - d + \theta_{t+1} f'(k_{t+1})\right] k_{t+1} \tag{7}$$

  for $t = 0, ..., T-1$.

- Step 4. Approximate the conditional expectation function in (6) by running a regression of a response variable y_t on $\Psi(k_t, \theta_t; \beta)$. Designate the estimated vector of coefficients as $\widehat{\beta}$.

- Step 5. Compute the next-iteration input $\beta^{(j+1)}$ as

$$\beta^{(j+1)} = (1 - \mu)\beta^{(j)} + \mu\widehat{\beta}, \tag{8}$$

  where $\mu \in (0, 1]$ is a dampening parameter.

  Iterate on these steps until convergence. Our convergence criterion requires the average relative difference between the capital series obtained on two subsequent iterations to be smaller than $10^{-\omega}\mu$ with $\omega > 0$ and $\mu \in (0, 1]$, i.e.

$$\frac{1}{T}\sum_{t=1}^{T}\left|\frac{k_{t+1}^{(j)} - k_{t+1}^{(j+1)}}{k_{t+1}^{(j)}}\right| < 10^{-\omega}\mu, \tag{9}$$

  where $\left\{k_{t+1}^{(j)}\right\}_{t=1}^{T}$ and $\left\{k_{t+1}^{(j+1)}\right\}_{t=1}^{T}$ are the capital series obtained on iterations j and $j+1$, respectively.[7]

As indicated in (8), the dampening parameter μ controls the amount by which coefficients are updated on each iteration. The effect of μ on the performance of the algorithm is twofold: A larger value of μ increases the speed of convergence, but it also increases the likelihood that the simulated series become too explosive or implosive and causes the algorithm to crash.

[7] The described stochastic simulation algorithm parameterizes the conditional expectation function and hence is a PEA.

Typically, one must run a number of experiments to find the value of μ that is most suitable for a given application.

The convergence criterion (9) depends on μ because the difference between the capital series obtained on two subsequent iterations depends on μ. In particular, if μ is equal to zero, the series do not change from one iteration to the next (i.e. $k_{t+1}^{(j)} = k_{t+1}^{(j+1)}$); if μ is small, the difference between successive capital series is small as well. Adjusting our convergence criterion to μ allows us to ensure that different values of μ lead to roughly the same accuracy in the simulated series. Since the simulated series are our true object of interest, the convergence criterion (9) based on the distance between successive series is more appealing than the criterion used in the previous literature (e.g., Marcet and Lorenzoni, 1999, Maliar and Maliar, 2003), which is based on the distance between the assumed and re-estimated vectors of coefficients, $\beta^{(j)} - \widehat{\beta}$. In the remainder of the paper, we focus on approximation methods to be used in Step 4, as well as on other related choices such as a choice of a decision rule to parameterize, a choice of a polynomial family, etc.

3 Linear versus non-linear regression models

To implement the regression given in Step 4 of the stochastic simulation algorithm, we need to choose a functional form $\Psi(k_t, \theta_t; \beta)$. One possible choice for $\Psi(k_t, \theta_t; \beta)$ is suggested by the version of the model (2) – (4) with a closed-form solution. Under the assumptions of full depreciation of capital, $d = 1$, the Cobb-Douglas production function, $f(k) = k^{\alpha}$, and the logarithmic utility function, $u(c) = \ln c$, the capital and consumption decision rules are given by $k_{t+1} = \alpha\delta\theta_t k_t^{\alpha}$ and $c_t = (1 - \alpha\delta)\theta_t k_t^{\alpha}$, respectively. The substitution of the closed-form solution for c_t, c_{t+1} and k_{t+1} into the definition of y_t given in (7) implies that $y_t = k_{t+1}$. To match this result, one can assume that $\Psi(k_t, \theta_t; \beta)$ is given by the following exponentiated polynomial

$$\Psi(k_t, \theta_t; \beta) = \exp\left[\beta_0 + \beta_1 \ln k_t + \beta_2 \ln \theta_t\right]. \tag{10}$$

A regression of k_{t+1} on $\Psi(k_t, \theta_t; \beta)$ should give us $\beta_0 = \ln \alpha\delta$, $\beta_1 = \alpha$ and $\beta_2 = 1$.

The log-linearity of the closed-form solution, in conjunction with the fact that the distribution for shock θ_t is log-normal, suggests that an exponentiated polynomial can be an adequate choice for the functional form $\Psi(k_t, \theta_t; \beta)$ not only in the model with the closed-form solution but also under general

assumptions about u, $f$and d when no closed-form solution exists. A general exponentiated polynomial specification for Ψ is

$$\Psi(k, \theta; \beta) = \exp(X\beta). \tag{11}$$

Here, $\Psi(k, \theta; \beta) \equiv (\Psi(k_0, \theta_0; \beta), ..., \Psi(k_{T-1}, \theta_{T-1}; \beta))' \in \mathbb{R}^T$ is a vector of observations on Ψ; $X \equiv [1_T, X_1, ..., X_n] \in \mathbb{R}^{T\times(n+1)}$ is a matrix of explanatory variables, in which 1_T is a $T \times 1$ vector whose entries are equal to 1, and $X_i \in \mathbb{R}^T$ is a vector of observations on the i-th polynomial term, which is a function of capital k_t and/or shock θ_t; and $\beta \equiv (\beta_0, \beta_1, ..., \beta_n)' \in \mathbb{R}^{n+1}$ is a vector of regression coefficients. In particular, for the first-order exponentiated polynomial approximation given in (10), $X = [1_T, \ln k, \ln \theta]$, in which $k \equiv (k_0, ..., k_{T-1})' \in \mathbb{R}^T$ and $\theta \equiv (\theta_0, ..., \theta_{T-1})' \in \mathbb{R}^T$ are vectors of observations on capital and shock, respectively. The exponentiated polynomial specifications have been commonly used in the literature relying on stochastic simulations; see Den Haan and Marcet (1990), Marcet and Lorenzoni (1999), Christiano and Fisher (2000), Maliar and Maliar (2003a), etc.

The exponentiated polynomial specification for Ψ in (11) leads to the following regression model:

$$Y = \exp(X\beta) + \varepsilon, \tag{12}$$

where $Y \equiv (y_0, y_1, ..., y_{T-1})' \in \mathbb{R}^T$, and $\varepsilon \in \mathbb{R}^T$ is a vector of residuals. An undesirable feature of the regression model (12) is its non-linearity, which means that it is estimated with non-linear estimation methods. Such methods typically require an initial guess for the coefficients, may require computing Jacobian and Hessian matrices and can be slow and numerically unstable. Furthermore, non-linear estimation methods become increasingly costly and cumbersome when we move to models with many state variables.

As an alternative to the exponentiated polynomial specification, we propose to use a simple polynomial specification for the approximating function Ψ,

$$\Psi(k, \theta; \beta) = X\beta. \tag{13}$$

In particular, for the first-degree ordinary polynomial approximation, $X = [1_T, k, \theta]$. The polynomial specification for Ψ in (13) leads to the standard linear regression model

$$Y = X\beta + \varepsilon. \tag{14}$$

The advantages of the linear regression model (14) over the non-linear regression model (12) are both that no initial guess of coefficient value is required

and that the estimation of coefficients can be performed with linear regression methods, which are faster and more numerically stable.

In the model with the closed-form solution, the polynomial specification for Ψ given in (13) will not lead to as an accurate solution as will the exponentiated polynomial specification given in (11), which matches the closed-form solution exactly. However, under general assumptions about u, f and d, the distribution for y_t is not known, and there is no a priori reason to think that the exponentiated polynomial specification will deliver a more accurate solution than will the polynomial specification.[8] We compare the properties of the solutions under the polynomial and exponentiated polynomial specifications in Section 7.3.

4 Ill-conditioned LS problem

To estimate β in both the linear and non-linear regression models, we can use the least-squares (LS) technique which minimizes the squared sum of residuals between the response variable Y and the approximation $\Psi(k, \theta; \beta)$, such that

$$\min_{\beta} \|Y - \Psi(k, \theta; \beta)\|_2^2 = \min_{\beta} [Y - \Psi(k, \theta; \beta)]' [Y - \Psi(k, \theta; \beta)]. \quad (15)$$

In this case, $\|\cdot\|_2$ denotes the L_2 (Euclidean) vector norm, i.e. for a vector $z = (z_1, ..., z_T)'$, we have $\|z\|_2 \equiv \left(\sum_{t=1}^{T} z_t^2\right)^{1/2}$. Under the linear regression model (14), a solution to the LS problem (15) is given by the standard OLS estimator:

$$\widehat{\beta} = (X'X)^{-1} X'Y. \quad (16)$$

For the non-linear regression model (12), the LS estimator generally cannot be written explicitly and should be computed with non-linear LS (NLLS) methods; we discuss these methods in Section 5.6.

[8] For some algorithms, we can discriminate between the two types of polynomial specifications based on the properties of the solution. In particular, for algorithms that linearize the optimality conditions around a steady state, Den Haan and Marcet (1994) show that parameterizing a decision rule for $\ln k_{t+1}$ in terms of $\ln k_t$ and $\ln \theta_t$ is preferable to parameterizing a decision rule for k_{t+1} in terms of k_t and $\ln \theta_t$ so long as the solution satisfies the property that an increase in the volatility of shock σ causes the agent to save more capital.

It turns out that estimating the linear regression model (14) using OLS can often cause the matrix $X'X$ to be ill-conditioned. The degree of ill-conditioning of $X'X$ can be measured in terms of a condition number which is defined as a ratio of the matrix's largest eigenvalue, λ_1, to its smallest eigenvalue, λ_n, i.e. $\kappa(X'X) \equiv \lambda_1/\lambda_n$. The eigenvalues of $X'X$ are defined by the standard eigenvalue decomposition of $X'X$,

$$X'X = V\Lambda V', \tag{17}$$

for which Λ is an $n \times n$ diagonal matrix with ordered eigenvalues of $X'X$ on its diagonal, $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$, and V is an $n \times n$ orthogonal matrix of eigenvectors of $X'X$. The larger the condition number, the smaller the determinant $\det(X'X) = \det(\Lambda) = \lambda_1\lambda_2...\lambda_n$ and the closer is $X'X$ to being singular (not invertible). Thus, a matrix is *ill-conditioned* if it has a large condition number.

An ill-conditioned $X'X$ matrix has a dramatic impact on the outcome of the stochastic simulation algorithm: First, the computer may fail to deliver the OLS estimator if the degree of ill-conditioning is so high that computing the inverse of $X'X$ leads to an overflow problem. Second, the OLS coefficients may change drastically from one iteration to another which causes cycling and leads to non-convergence. Finally, even if convergence is achieved, the resulting approximation may be inaccurate.

Two causes of an ill-conditioned $X'X$ matrix are multicollinearity and poor scaling of the variables constituting X. Multicollinearity occurs because high-order polynomial terms forming the matrix X are significantly correlated.[9] The following example illustrates the effects of multicollinearity on the LS solution under $T = n = 2$.[10]

Example 1 *Let* $Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ *and* $X = [X_1, X_2] = \begin{bmatrix} x_{11} & x_{11}+\phi \\ x_{12} & x_{12} \end{bmatrix}$, *with* $x_{12} \neq 0$ *and* $\phi \neq 0$. *In this case, the OLS solution* (16) *is*

$$\widehat{\beta}_1 = \frac{y_2}{x_{12}} - \widehat{\beta}_2 \text{ and } \widehat{\beta}_2 = \frac{y_1}{\phi} - \frac{y_2 x_{11}}{\phi x_{12}}. \tag{18}$$

[9] The multicollinearity problem also occurs in the non-linear regression model with the exponentiated polynomial specification (12). In particular, Den Haan and Marcet (1990) find that even under a second-degree ordinary polynomial approximation, a cross term is highly correlated with the other terms and should be removed from the regression.

[10] If X is square ($T = n$) and has a full rank, the system $Y = X\beta$ has a unique solution $\beta = X^{-1}Y$ which coincides with the OLS solution.

If $\phi \to 0$, *we have* $\det(X'X) = x_{12}^2\phi^2 \to 0$, $\kappa(X'X) = \frac{x_{11}+x_{12}}{\phi} \to \infty$, $\widehat{\beta}_1, \widehat{\beta}_2 \to \pm\infty$, *and* $\widehat{\beta}_1 \approx -\widehat{\beta}_2$, *i.e. a large positive coefficient on one variable is canceled by a similarly large negative coefficient on its correlated counterpart.*

The scaling problem arises when polynomial terms in X have dramatically different means and variances due to differential scaling among either the state variables (k_t and θ_t) or polynomial terms of different orders (e.g., k_t and k_t^5) have different scales. Columns with entries that are excessively small in absolute value have the same effect on the LS solution as columns of zeros. We illustrate the effect of the scaling problem on the LS solution with the following example.

Example 2 *Let* $Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ *and* $X = [X_1, X_2] = \begin{bmatrix} x_{11} & \phi \\ x_{12} & 0 \end{bmatrix}$ *with* $x_{12} \neq 0$ *and* $\phi \neq 0$. *In this case, the OLS solution* (16) *is*

$$\widehat{\beta}_1 = \frac{y_2}{x_{12}} \text{ and } \widehat{\beta}_2 = \frac{y_1}{\phi} - \frac{y_2 x_{11}}{\phi x_{12}}. \tag{19}$$

If $\phi \to 0$, *we have* $\det(X'X) = x_{12}^2\phi^2 \to 0$, $\kappa(X'X) = \frac{x_{11}}{\phi} \to \infty$, *and* $\widehat{\beta}_2 \to \pm\infty$, *i.e. entries of* X_2 *that are excessively small in absolute value result in the assignation of an excessively large absolute value to coefficient* $\widehat{\beta}_2$.

A comparison of Examples 1 and 2 shows that the effects of multicollinearity and poor scaling on the OLS solution are similar. In both examples, when $\phi \to 0$, the matrix $X'X$ becomes ill-conditioned, and an information loss results due to the rounding up of excessively large entries of $(X'X)^{-1}$. Furthermore, since ϕ appears in the denominator of the OLS solutions, the OLS coefficients become excessively large in size and oversensitive to changes in ϕ. These effects are the cause of the previously discussed problems of low accuracy and non-convergence of the stochastic simulation algorithm.

On the basis of our examples, we can propose several strategies for enhancing the numerical stability of the stochastic simulation algorithm, namely,

1. To consider alternative variants of the Euler equation (6); this leads to different decision rules and affects the response variable Y.

2. To consider alternative polynomial representations of the approximating function Ψ; this affects the explanatory variables $X_1, ..., X_n$.

3. To re-scale the variables to comparable units.

4. To develop LS estimation methods that are suitable for handling ill-conditioned problems.

5. To replace the ill-conditioned LS problem with a non-LS problem that avoids computing the inverse of the ill-conditioned matrix $X'X$.

5 Enhancing numerical stability

In this section, we elaborate on the previously proposed strategies for enhancing the numerical stability of the stochastic simulation algorithm. In Sections 5.1-5.3, we discuss the choice of a decision rule to parameterize, the choice of a polynomial representation and the issue of data normalization, respectively. In Sections 5.4 and 5.5, we describe numerically stable methods for estimating the linear regression model, and in Section 5.6, we generalize such methods to the case of the non-linear regression model.

5.1 Choosing a decision rule to parameterize

In the benchmark version of the stochastic simulation algorithm, we parameterize the decision rule for the next-period capital stock, $k_{t+1} = K(k_t, \theta_t)$. As an alternative, we consider a parameterization of the marginal utility of consumption that is standard in PEA literature:

$$u'(c_t) = \delta E_t \left\{ u'(c_{t+1}) \left[1 - d + \theta_{t+1} f'(k_{t+1}) \right] \right\} \simeq \delta \Psi^u(k_t, \theta_t; \beta). \tag{20}$$

Under the parameterization (20), the third step of the stochastic simulation algorithm, described in Section 2, should be modified such that the assumed decision rule (20) is used to restore consumption and to compute the next-period capital stock k_{t+1} from the budget constraint (3). In this case, the variable under the expectation is modified as $y_t^u \equiv u'(c_{t+1}) \left[1 - d + \theta_{t+1} f'(k_{t+1}) \right]$. The remainder of the algorithm, including the dampening procedure and the convergence criterion, is the same as in the benchmark case. We discuss other parameterizations of the Euler equation in Section 7.4.

5.2 Choosing a polynomial representation

To approximate Ψ, we restrict our attention to the polynomial space of functions. We consider two alternative representations: an ordinary polynomial representation and a Hermite polynomial representation. The Hermite (probabilists') polynomials can be defined by

$$H_m(x) = (-1)^m e^{x^2/2} \frac{D^m}{Dx^m} e^{-x^2/2}, \tag{21}$$

where $\frac{D^m}{Dx^m}$ is the m-th order derivative of $e^{-x^2/2}$ with respect to $x \in (-\infty, \infty)$.[11] It is also possible to define the Hermite polynomials with a simple recursive formula: $H_0(x) = 1$, $H_1(x) = x$ and $H_m(x) = xH_m(x) - mH_{m-1}(x)$. Below, we compare the Hermite polynomials, $H_m(x)$, to the ordinary polynomials, $P_m(x)$, up to degree five:

$$\begin{array}{ll} P_0(x) = 1 & H_0(x) = 1 \\ P_1(x) = x & H_1(x) = x \\ P_2(x) = x^2 & H_2(x) = x^2 - 1 \\ P_3(x) = x^3 & H_3(x) = x^3 - 3x \\ P_4(x) = x^4 & H_4(x) = x^4 - 6x^2 + 3 \\ P_5(x) = x^5 & H_5(x) = x^5 - 10x^3 + 15x. \end{array} \tag{22}$$

The Hermite polynomials in (21) are orthogonal with respect to the standard normal probability distribution whose density function is $\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$, where x has a zero expected value and unit variance. Since $\ln \theta_t$ is distributed normally, the Hermite polynomials are a natural choice for the case of a decision rule expressed in terms of logarithms. Additionally, the Hermite polynomials can have advantages over the ordinary polynomials even if the state variables are expressed in levels. This point is illustrated by a comparison of the ordinary and Hermite polynomials shown in Figures 3 and 4, respectively. In the case of the ordinary polynomials, the basis functions $P_m(x)$, $m = 1, ..., 5$ appear very similar (namely, $P_2(x) = x^2$ looks similar to $P_4(x) = x^4$, and $P_3(x) = x^3$ looks similar to $P_5(x) = x^5$). As a result,

[11] The probabilists' definition of the Hermite polynomials builds on the probability density function for normal distributions. There is an alternative physicists' definition for the Hermite polynomials, and this alternative definition is used to model periodic oscillations. The two types of Hermite polynomials can be interconverted via scaling.

the explanatory variables for the regression that appears in Step 4 of the stochastic simulation algorithm are likely to be correlated (i.e. the LS problem is ill-conditioned) and estimation methods (e.g., OLS) are likely to fail because they cannot distinguish between similarly shaped polynomial terms. In contrast, the Hermite polynomials case is characterized by difference in the shapes of the basis functions $H_m(x)$, $m = 1, ..., 5$, hence, the multicollinearity problem manifests to a much lesser degree, if at all.

We approximate the function Ψ by a complete set of polynomials in k_t and θ_t. For example, the complete set of Hermite polynomials of degree three is given in levels by

$$\begin{aligned}\Psi(k_t, \theta_t; \beta) = \beta_0 + \beta_1 k_t + \beta_2 \theta_t + \beta_3 \left(k_t^2 - 1\right) + \beta_4 k_t \theta_t + \beta_5 \left(\theta_t^2 - 1\right) + \\ + \beta_6 \left(3k_t^3 - 3k_t\right) + \beta_7 \left(k_t^2 - 1\right) \theta_t + \beta_8 k_t \left(\theta_t^2 - 1\right) + \beta_9 \left(3\theta_t^3 - 3\theta_t\right),\end{aligned}$$

where variables k_t and θ_t are previously normalized to have zero means and unit variances. The polynomial approximations of orders one, two, three, four and five have 3, 6, 10, 15 and 21 coefficients, respectively.

5.3 Normalizing the variables

To reduce the likelihood of the scaling problem highlighted in Example 2, we must normalize (i.e. center and scale) the variables. Centering consists of subtracting the sample mean from each observation , and scaling consists of dividing each observation by the sample standard deviation. By construction, a centered variable has a zero mean, and a scaled variable has a unit standard deviation.

In the case of the linear regression model (14), we first center and scale both the response variable Y and the explanatory variables of X. We then estimate a regression model without a constant term (i.e. intercept) to obtain the vector of coefficients $\left(\widehat{\beta}_1^*, ..., \widehat{\beta}_n^*\right)$. We finally restore the coefficients $\widehat{\beta}_1, ..., \widehat{\beta}_n$ and the intercept $\widehat{\beta}_0$ in the original (unnormalized) regression model according to

$$\widehat{\beta}_i = (\sigma_Y / \sigma_{X_i}) \widehat{\beta}_i^*, \quad i = 1, ..., n, \quad \text{and} \quad \widehat{\beta}_0 = \overline{Y} - \sum_{i=1}^{n} \widehat{\beta}_i^* \overline{X}_i, \tag{23}$$

where $\overline{Y}$ and $\overline{X}_i$ are the sample means, and σ_Y and σ_{X_i} are the sample

standard deviations of the original unnormalized variables Y and X_i, respectively.[12]

In addition to reducing the impact of the scaling problem, the normalization of variables is also useful for the regularization methods, considered in Sections 5.4.2 and 5.5.2. These methods adjust for the effects of ill-conditioning by penalizing large values of the regression coefficients. Since the effect of a penalty depends on the size of the coefficients (i.e., on the scales of the variables), centering and scaling all explanatory variables to zero means and unit standard deviations allows us to use the same penalty for all coefficients. Furthermore, centering of the response variable Y allows us to estimate a no-intercept model and to impose a penalty on the coefficients $\beta_1, ..., \beta_n$ without penalizing the intercept β_0 (recall that we recover the intercept according to (23) after all other coefficients are computed).

In the case of the non-linear regression model (12), we can both center and scale the explanatory variables of X, however, we cannot center the response variable, Y, because the exponentiated polynomial specification (11) cannot match negative deviations of Y from its mean. In this case, we shall scale but not center the response variable Y and estimate the regression equation (12) with a constant term included.

5.4 LS approaches to the linear regression model

In this section, we present two LS approaches that are more numerically stable and more suitable for dealing with ill-conditioning than the standard OLS approach. The first approach, called *LS using SVD (LS-SVD)*, infers $(X'X)^{-1}$ matrix included in the OLS formula (16) from a singular value decomposition (SVD) of the matrix X. The second approach, called *regularized LS using Tikhonov regularization (RLS-Tikhonov)*, relies on a specific (Tikhonov) regularization of the ill-conditioned LS problem that imposes penalties based on the size of the regression coefficients. In essence, the LS-SVD approach finds a solution to the original ill-conditioned LS problem, while the RLS-Tikhonov approach modifies (regularizes) the original ill-conditioned LS problem into a well-conditioned problem.

[12]To maintain a relatively simple system of notation, we shall not introduce separate notation for normalized and unnormalized variables. Instead, we shall remember that when the linear regression model (14) is estimated with normalized variables, the vector of coefficients β is n-dimensional; when (14) is estimated with unnormalized variables, it is $(n+1)$-dimensional.

5.4.1 LS-SVD

We can use an SVD of the matrix X to re-write the OLS formula (16) in a way that does not require the explicit computation of $(X'X)^{-1}$. For a matrix $X \in \mathbb{R}^{T\times n}$ with $T > n$, we compute a thin SVD,

$$X = USV', \tag{24}$$

where $U \in \mathbb{R}^{T\times n}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and $S \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonal entries $s_1 \geq s_2 \geq ... \geq s_n \geq 0$, known as *singular values* of X.[13] There is a close relation between the SVD of X and the eigenvalue decomposition of $X'X$ defined in (17):

$$V\Lambda V' = X'X = (USV')' \times (USV') = VS'SV'. \tag{25}$$

As this implies, $S'S = \Lambda$ and thus, the singular values of X and the eigenvalues of $X'X$ are connected by $s_i = \sqrt{\lambda_i}$. In conjunction with formula (24), this last result implies that the condition number of X is given by $\kappa(X) = \kappa(S) = \sqrt{\kappa(X'X)}$. Therefore, X and S are likely to be less ill-conditioned than $X'X$.

We rewrite the OLS estimator $\widehat{\beta} = (X'X)^{-1} X'Y$ in terms of the SVD (24) as follows:

$$\widehat{\beta} = (VS'SV')^{-1} VS'U'Y = VS^{-1}U'Y. \tag{26}$$

If $X'X$ is well-conditioned, the OLS formula (16) and the LS-SVD formula (26) give identical estimates of β. However, if $X'X$ is ill-conditioned and the standard OLS estimator cannot be computed, it is still possible that matrices X and S are sufficiently well-conditioned for the successful computation of the LS-SVD estimator to occur.[14]

We illustrate the advantage of LS-SVD over the standard OLS by way of example.

[13] This version of SVD is called *thin* to emphasize that it is a reduced version of the full SVD with $U \in \mathbb{R}^{T\times T}$ and $S \in \mathbb{R}^{T\times n}$. Under $T \gg n$, using the thin SVD instead of the full SVD allows us to save memory and time.

[14] Another decomposition of X that leads to a numerically stable LS approach is a QR factorization. A (thin) QR factorization of a $T \times n$ matrix X is given by $X = QR$, where $Q \in \mathbb{R}^{T\times n}$ is an orthogonal matrix, and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix. The LS solution based on the QR factorization is given by $\widehat{\beta} = R^{-1}Q'Y$.

Example 3 *Let* $Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ *and* $X = [X_1, X_2] = \begin{bmatrix} \phi & 0 \\ 0 & 1 \end{bmatrix}$ *with* $\phi \neq 0$. *In* (24), *we have* $S = X$ *and* $U = V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. *The condition numbers of* S *and* $X'X$ *are related by* $\kappa(S) = \sqrt{\kappa(X'X)} = \phi$. *The OLS and the LS-SVD estimators coincide such that*

$$\widehat{\beta} = (X'X)^{-1} X'Y = \begin{bmatrix} 1/\phi^2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \phi y_1 \\ y_2 \end{bmatrix} =$$

$$= VS^{-1}U'Y = \begin{bmatrix} 1/\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_1/\phi \\ y_2 \end{bmatrix}. \tag{27}$$

If $\phi < 1$, *the largest elements of* $(X'X)^{-1}$ *and* S^{-1} *are* $1/\phi^2$ *and* $1/\phi$, *respectively. When* $\phi \approx 0$, *the computer has a better chance to compute* $1/\phi$ *than* $1/\phi^2$ *since* $1/\phi \ll 1/\phi^2$.

In practice, the SVD of a matrix is typically computed in two steps. First, the matrix is reduced to a bidiagonal matrix; afterwards, the singular values of the resulting bidiagonal matrix are computed iteratively with QR or Jacobi methods; for a description of these methods see, e.g., Bjorck (1996), pp. 81-98. Routines that compute the SVD of matrices are readily available in modern programming languages.

5.4.2 RLS-Tikhonov

A regularization is a process of re-formulating an ill-conditioned problem for numerical treatment by imposing additional restrictions on the solution. Tikhonov regularization is the most commonly used regularization method for dealing with ill-conditioned problems in approximation theory.[15] In statistics, this method is known as ridge regression and is classified as a shrinkage method to emphasize that the length of the vector of estimated coefficients shrinks relative to that of the non-regularized solution.[16] The key idea behind Tikhonov regularization is the mitigation of the effects of ill-conditioning of

[15] Tikhonov regularization is named for Andrey Tikhonov who developed this method and applied it for solving integral equations; see Tikhonov (1963).

[16] The term *ridge regression* was introduced by Hoerl (1959) to emphasize its similarity to ridge analysis in his previous study of second-order response surfaces. Hoerl and Kennard (1970) popularized ridge regressions by providing theoretical and practical insights into benefits of the method.

the LS problem by shrinking the excessively large LS coefficients that are responsible for the numerical problems experienced by the stochastic simulation algorithm (see detailed discussion in Section 4).

Formally, Tikhonov regularization consists of imposing an L_2 penalty on the size of the regression coefficients; i.e. for a given regularization parameter $\eta \geq 0$, we search for $\beta(\eta)$ that solves

$$\min_{\beta} \|Y - X\beta\|_2^2 + \eta \|\beta\|_2^2 = \min_{\beta} (Y - X\beta)'(Y - X\beta) + \eta\beta'\beta, \qquad (28)$$

where $Y \in \mathbb{R}^T$ and $X \in \mathbb{R}^{T \times n}$ are centered and scaled, and $\beta \in \mathbb{R}^n$. The parameter η controls the amount by which the regression coefficients are shrunk, with larger values of η leading to greater shrinkage. As outlined in Section 5.3, centering and scaling allow us to use the same penalty for all explanatory variables and running a no-intercept regression allows us to penalize all regression coefficients except of the intercept.

Finding the first-order condition of (28) with respect to β gives us the following estimator

$$\widehat{\beta}(\eta) = (X'X + \eta I_n)^{-1} X'Y, \qquad (29)$$

where I_n is an identity matrix of order n. Note that Tikhonov regularization adds a positive constant to $X'X$ prior to inverting this matrix. Thus, even if $X'X$ is singular, the matrix $X'X + \eta I_n$ is non-singular, which allows its inverse to be computed. The estimator $\widehat{\beta}(\eta)$ in (29) is biased but it tends to have a smaller total mean square error than the OLS estimator; see Hoerl and Kennard (1970) for a proof.

5.5 LAD approaches to the linear regression model

In this section, we replace the ill-conditioned LS problem with a least-absolute deviations (LAD) or L_1 problem, which consists of minimizing the sum of absolute deviations. Since the LAD problem does not involve minimizing the squared sum of residuals, it does not require computing $(X'X)^{-1}$. In Section 5.5.1, we develop primal and dual linear programming formulations of the LAD problem, and in Section 5.5.2, we regularize the primal and dual formulations of the LAD problem by imposing a penalty on the size of the regression coefficients.

5.5.1 LAD

As applied to the linear regression model (14), the LAD problem is

$$\min_{\beta} \|Y - X\beta\|_1 = \min_{\beta} 1_T' |Y - X\beta| . \tag{30}$$

where $\|\cdot\|_1$ denotes L_1 vector norm, i.e. for a vector $z = (z_1, ..., z_T)'$, we have $\|z\|_1 \equiv \sum_{t=1}^{T} |z_t|$, and $|\cdot|$ denotes the absolute value.[17] Without a loss of generality, we assume that X and Y are centered and scaled.

The LAD approach is known to be more robust to outliers than the LS approach because it minimizes errors without squaring them and thus, places comparatively less weight on distant observations than the LS approach does. The statistical literature suggests that the LAD approach is preferable to the LS approach in many applications; see Narula and Wellington (1982), and Dielman (2005) for surveys of the literature on the LAD estimation of econometric models. An example of economic application where the LAD approach was critical for solving an approximation problem is Michelangeli (2008).

There is no explicit solution to the LAD problem (30) similar to the explicit OLS solution (16) to the LS problem. However, we can re-formulate the LAD problem to consist of a linear objective function bound by a set of linear constraints, and we can solve this reformulated problem with standard linear programming techniques. By substituting the absolute value term $|Y - X\beta|$ in (30) with a vector $p \in \mathbb{R}^T$, we obtain

$$\min_{\beta,\, p} 1_T' p \tag{31}$$

$$\text{s.t.} \quad -p \leq Y - X\beta \leq p. \tag{32}$$

This problem has $n + T$ unknowns. To find a solution to (31) and (32), linear programming methods convert the $2T$ inequality restrictions imposed by (32) into equalities by adding non-negative slack variables; this effectively

[17] LAD regression is a particular case of quantile regressions introduced by Koenker and Bassett (1978); see also Koenker and Hallock (2001) for many examples. The central idea behind quantile regressions is the assignation of differing weights to positive versus negative residuals, $Y - X\beta$. A θ-th regression quantile, $\theta \in (0, 1)$, is defined as a solution to the problem of minimizing a weighted sum of residuals, where θ is a weight on positive residuals. The LAD estimator is the regression median, i.e. the regression quantile for $\theta = 1/2$.

creates $2T$ additional lower bounds on the slack variables. Although this formulation of the LAD problem is intuitive, we argue below that it is not the most suitable for a numerical analysis.

LAD: primal problem (LAD-PP) Charnes et al. (1955) show that a linear LAD problem can be transformed into the standard linear programming form by representing each deviation as the difference between two non-negative variables. We use this result to elaborate an alternative formulation of the LAD problem (30).

We express the deviation for each observation as a difference between two non-negative variables u_t and v_t, such that

$$y_t - \sum_{i=0}^{n} \beta_i x_{it} = u_t - v_t, \tag{33}$$

where x_{it} is the t-th element of the vector X_i. The variables u_t and v_t can be interpreted as non-negative vertical deviations above and below the fitted line, $\widehat{y}_t = X_t\widehat{\beta}$, respectively; the sum $u_t + v_t$ is the absolute deviation between the fit $\widehat{y}_t$ and the observation y_t. Thus, the LAD problem is to minimize the total sum of absolute deviations subject to the system of equations (33). In vector notation, this problem is

$$\min_{\beta,u,v} \; 1_T'u + 1_T'v \tag{34}$$

$$\text{s.t. } u - v + X\beta = Y, \tag{35}$$

$$u \geq 0, \quad v \geq 0, \tag{36}$$

where $u,\ v \in \mathbb{R}^T$. We refer to the above linear programming problem as *primal problem*. A noteworthy property of its solution is that u_t or v_t cannot both be strictly positive; if so, we can subtract the same positive number from both u_t and v_t, which will reduce the value of the objective function without affecting the constraint (35). The advantage of the formulation given by (34) – (36) compared to that which is shown in (31) and (32) is that the former has T equality constraints while the latter had $2T$ constraints.

LAD: dual problem (LAD-DP) A useful result of linear programming is that every primal problem can be converted into a dual problem.[18] The

[18] See Ferris, Mangasarian, Wright (2007) for duality theory and examples.

dual problem corresponding to the primal problem (34) − (36) is

$$\max_{q} Y'q \tag{37}$$

$$\text{s.t. } X'q = 0, \tag{38}$$

$$-1_T \leq q \leq 1_T, \tag{39}$$

where $q \in \mathbb{R}^T$ is a vector of unknowns. Wagner (1959) argues that if the number of observations, T, is sizable (i.e. $T \gg n$), the dual problem (37) − (39) is less computationally cumbersome than the primal problem (34)−(36). Indeed, the dual problem contains only n equality restrictions, and the primal problem has contained T equality restrictions, while the number of lower and upper bounds on unknowns is equal to $2T$ in both problems. Note that the vector of coefficients β, which is a primary object of our estimation, does not enter into the dual problem explicitly. However, as noted in Wagner (1959), we need not make any extra computations to find the coefficients of the vector β since these coefficients are equal to the Lagrange multipliers associated with the equality restrictions given in (38).

5.5.2 Regularized LAD (RLAD)

In this section, we modify the original LAD problem (30) to incorporate an L_1 penalty on the vector of coefficients β.[19] We refer to the resulting problem as *regularized LAD* (RLAD). Like Tikhonov regularization, our regularization of the LAD problem reduces the impact of ill-conditioning on the solution by shrinking the values of the coefficients toward zero. Introducing an L_1 penalty in place of the L_2 penalty used in Tikhonov regularization, has an important advantage in the case of LAD: It preserves the linearity of the objective function, which allows the RLAD problem to be cast into a linear programming form. Formally, for a given regularization parameter $\eta \geq 0$, the RLAD problem attempts to find $\beta(\eta)$ that solves

$$\min_{\beta} \|Y - X\beta\|_1 + \eta \|\beta\|_1 = \min_{\beta} 1_T' |Y - X\beta| + \eta 1_n' |\beta|, \tag{40}$$

where $Y \in \mathbb{R}^T$ and $X \in \mathbb{R}^{T \times n}$ are centered and scaled, and $\beta \in \mathbb{R}^n$. As in the case of Tikhonov regularization, centering and scaling of X and Y in

[19] In statistics, the L_1 penalty was originally applied to the LS estimation of linear models by Tibshirani (1996) and is known as *lasso* (least absolute shrinkage and selection operator); see Hastie, Tibshirani and Friedman (2009), pp. 68-79.

the RLAD problem (40) allows us to use the same penalty parameter for all explanatory variables and to avoid penalizing an intercept.

There is another paper in the literature that considers a regularized version of the LAD problem, namely, Wang, Gordon and Zhu (2006) reformulate the RLAD problem (40) as a linear programming problem of type (31), (32) by representing the absolute value terms $|\beta_i|$ as $sign\,(\beta_i)\,\beta_i$. Here, we develop a linear programming formulation of the RLAD problem that is parallel to the primal problem $(34)-(36)$, i.e. we replace the absolute value terms $|\beta_i|$ with two variables as discussed in Section 5.5.1.

RLAD: primal problem (RLAD-PP) To cast the RLAD problem (40) into a linear programming form, we represent the coefficients of the vector β as differences between two non-negative variables: i.e. $\beta_i = a_i - b_i$, with $a_i \geq 0$, $b_i \geq 0$ for $i = 1, ..., n$. We then impose a linear penalty on each a_i and b_i. The resulting regularized version of the primal problem $(34)-(36)$ is

$$\min_{a,b,u,v} \; 1_T' u + 1_T' v + \eta 1_n' a + \eta 1_n' b \tag{41}$$

$$\text{s.t.} \quad u - v + Xa - Xb = Y, \tag{42}$$

$$u \geq 0, \quad v \geq 0, \tag{43}$$

$$a \geq 0, \quad b \geq 0, \tag{44}$$

where a, $b \in \mathbb{R}^n$ are vectors that define β. The above problem has $2T + 2n$ unknowns, as well as T equality restrictions as given in (42) and $2T + 2n$ lower bounds as given in (43) and (44).

RLAD: dual problem (RLAD-DP) The dual problem corresponding to the RLAD-PP $(41)-(44)$ is

$$\max_{q} \; Y'q \tag{45}$$

$$\text{s.t. } X'q \leqslant \eta \cdot 1_n, \tag{46}$$

$$-X'q \leqslant \eta \cdot 1_n, \tag{47}$$

$$-1_T \leq q \leq 1_T, \tag{48}$$

where $q \in \mathbb{R}^T$ is a vector of unknowns. Here, $2n$ linear inequality restrictions are imposed by (46) and (47), and $2T$ lower and upper bounds on T unknown

components of q are given in (48). By solving the dual problem, we obtain the coefficients of the vectors a and b as the Lagrange multipliers associated with the restrictions (46) and (47), respectively; we can then restore the RLAD estimator using $\beta = a - b$.

5.6 LS and LAD approaches to the non-linear regression model

In this section, we extend the approaches to the linear regression model (14) that are developed in Sections 5.4 and 5.5 to the case of the non-linear regression model,

$$Y = \Psi(k, \theta; \beta) + \varepsilon, \tag{49}$$

where $\beta \in \mathbb{R}^{n+1}$. The regression model with exponentiated polynomial (12) is a particular case of (49). For the non-linear regression model (49), we first consider the non-linear LS problem (15), and we then formulate the corresponding LAD problem.

The typical NLLS estimation method linearizes the NLLS problem (15) around a given initial guess, β, by using a first-order Taylor expansion of $\Psi(k_t, \theta_t; \beta)$ and makes a step $\Delta\beta$ toward a solution, $\widehat{\beta}$,

$$\widehat{\beta} \simeq \beta + \Delta\beta. \tag{50}$$

Using the linearity of the differential operator, we can derive an explicit expression for the step $\Delta\beta$. This step is given by a solution to the system of normal equations,

$$J'J\Delta\beta = J'\Delta Y, \tag{51}$$

where $J \equiv \begin{pmatrix} \frac{\partial\Psi(k_1,\theta_1;\beta)}{\partial\beta_0} & ... & \frac{\partial\Psi(k_1,\theta_1;\beta)}{\partial\beta_n} \\ ... & ... & ... \\ \frac{\partial\Psi(k_T,\theta_T;\beta)}{\partial\beta_0} & ... & \frac{\partial\Psi(k_T,\theta_T;\beta)}{\partial\beta_n} \end{pmatrix}$ is a Jacobian matrix of Ψ and $\Delta Y \equiv (y_1 - \Psi(k_1, \theta_1; \beta), ..., y_T - \Psi(k_T, \theta_T; \beta))'$; see Judd (1998), pp. 117-119. Typically, the NLLS estimation method does not give an accurate solution $\widehat{\beta}$ in a single step $\Delta\beta$, and must instead iterate on the step (50) until convergence.[20]

[20] Instead of the first-order Taylor expansion of $\Psi(k, \theta; \beta)$, we can consider a second-order Taylor expansion, which leads us to another class of non-linear optimization methods in which the step $\Delta\beta$ depends on a Hessian matrix; see Judd (1992), pp. 103-117, for a review.

A direct way to compute the step $\Delta\beta$ from (51) is to invert the matrix $J'J$, which yields the well-known *Gauss-Newton method*,

$$\Delta\beta = (J'J)^{-1} J'\Delta Y. \tag{52}$$

This formula (52) has a striking resemblance to the OLS formula (16): The terms X, Y and β, which appear in (16), are replaced in (52) by J, ΔY and $\Delta\beta$, respectively. If $J'J$ is ill-conditioned, as is often the case in applications, the Gauss-Newton method experiences the same difficulties in computing $(J'J)^{-1}$ and $\Delta\beta$ as the OLS method does in computing $(X'X)^{-1}$ and β.

To deal with the ill-conditioned matrix $J'J$ in the Gauss-Newton method (52), we can employ the LS approaches developed for the linear regression model in Sections 5.4.1 and 5.4.2. Specifically, we can compute an inverse of the ill-conditioned matrix $J'J$ by using LS methods based on SVD or QR factorization of J. We can also use the Tikhonov type of regularization, which leads to the *Levenberg-Marquart* method,

$$\Delta\beta = (J'J + \eta I_{n+1})^{-1} J'\Delta Y, \tag{53}$$

where $\eta \geq 0$ is a regularization parameter.[21]

Furthermore, we can replace the ill-conditioned NLLS problem with a non-linear LAD (NLLAD) problem,

$$\min_{\beta} \|Y - \Psi(k, \theta; \beta)\|_1 = \min_{\beta} 1_T' |Y - \Psi(k, \theta; \beta)|. \tag{54}$$

As in the NLLS case, we can proceed by linearizing the non-linear problem (54) around a given initial guess β. The linearized version of the NLLAD problem (54) is

$$\min_{\Delta\beta} 1_T' |\Delta Y - J\Delta\beta|. \tag{55}$$

This problem (55) can be formulated as a linear programming problem: specifically, we can set up non-regularized primal and dual problems, as well as regularized primal and dual problems, analogous to those considered in Sections 5.5.1 and 5.5.2.

Example 4 *Let us formulate a regularized primal problem for* (55) *that is parallel to* (41) − (44)*. Fix some initial a and b (which determine initial*

[21] This method was proposed independently by Levenberg (1944) and Marquart (1963).

$\beta = a - b$) and solve for Δa and Δb from the following linear programming problem:

$$\min_{\Delta a, \Delta b, u, v} \; 1_T' u + 1_T' v + \eta 1_n' \Delta a + \eta 1_n' \Delta b \tag{56}$$

$$s.t. \quad u - v + J\Delta a - J\Delta b = \Delta Y, \tag{57}$$

$$u \geq 0, \quad v \geq 0, \tag{58}$$

$$\Delta a \geq 0, \quad \Delta b \geq 0. \tag{59}$$

Compute $\widehat{a} \simeq a + \Delta a$ and $\widehat{b} \simeq b + \Delta b$, and restore the RLAD estimator $\widehat{\beta} \simeq (a + \Delta a) - (b + \Delta b)$. As in the case of NLLS methods, we will not typically obtain an accurate solution $\widehat{\beta}$ in a single step, but must instead solve the problem (56) − (59) *iteratively until convergence.*

To set up a regularized dual problem for (55), which is analogous to (45) − (48), we shall replace X and Y with J and ΔY, respectively.

We should finally notice that the NLLS and NLLAD regularization methods described in this section penalize all coefficients equally, including a constant term. Therefore, prior to applying these methods, we need to appropriately normalize the explanatory variables and set the penalty on the constant term to zero.

6 Unified principal component method

The methods developed in Sections 5.4 - 5.6 are more suitable for dealing with ill-conditioned problems than the standard OLS or Gauss-Newton methods; nonetheless, they are still likely to fail when the degrees of ill-conditioning are extremely high. In particular, these methods will not perform appropriately when the matrix X contains linearly dependent columns (i.e. X is singular). In this case, the solution is not uniquely determined, and all the studied methods will either fail to deliver a solution or deliver one of many possible solutions, which leads to cycling of the stochastic simulation algorithm.

In this section, we describe a numerically stable method that can handle any degrees of data ill-conditioning including the case of linearly dependent columns of X. Specifically, we combine the methods of Sections 5.4 - 5.6 with the principal component analysis, used in statistics to deal with multicollinearity. The key idea of the proposed method is to reduce the degrees

of ill-conditioning to a reasonable level prior to applying the methods of Sections 5.4 - 5.6.

Let $X \in \mathbb{R}^{T \times n}$ be a matrix of centered and scaled explanatory variables and consider the SVD of X defined in (24). Let us make the following linear transformation of the explanatory variables: $Z \equiv XV$, where $Z \in \mathbb{R}^{T \times n}$. The variables $Z_1, ..., Z_n$ are called *principal components* of X and are orthogonal, $Z_i'Z_i = s_i^2$ and $Z_j'Z_i = 0$ for any $j \neq i$. Principal components have two noteworthy properties. First, the sample mean of each principal component Z_i is equal to zero, since it is given by a linear combination of centered variables $X_1, ..., X_n$, each of which has a zero mean; additionally, the variance of each principal component is equal to s_i^2/T, because we have $Z_i'Z_i = s_i^2$.

Since the SVD orders the singular values from the largest, the first principal component Z_1 has the largest sample variance among all the principal components, while the last principal component Z_n has the smallest sample variance. In particular, if Z_i has a zero variance (equivalently, a zero singular value, $s_i = 0$), then all entries of Z_i are equal to zero, $Z_i = (0, ..., 0)'$, which implies that the variables $X_1, ..., X_n$ constituting this particular principal component are linearly dependent. Therefore, we can reduce the degrees of ill-conditioning of X to a "desired" level by excluding low variance principle components corresponding to small singular values.

To formalize the above idea, let $\overline{\kappa}$ represent the largest condition number of the matrix X that we are willing to accept. Let us compute the ratios of the largest singular value to all other singular values, $\frac{s_1}{s_2}, ..., \frac{s_1}{s_n}$. (Recall that the last ratio is the actual condition number of the matrix X; $\kappa(X) = \kappa(S) = \frac{s_1}{s_n}$). Let $Z^r \equiv (Z_1, ..., Z_r) \in \mathbb{R}^{T \times r}$ be the first r principal components for which $\frac{s_1}{s_i} \leq \overline{\kappa}$, and let us remove the last $n - r$ principal components for which $\frac{s_1}{s_i} > \overline{\kappa}$. By construction, the matrix Z^r has a condition number which is smaller than or equal to $\overline{\kappa}$.

We re-write the linear regression model (14) in terms of the principal components Z^r,

$$Y = Z^r \vartheta + \varepsilon, \tag{60}$$

where Y is centered and scaled, and $\vartheta \in \mathbb{R}^r$ is a vector of coefficients. To estimate the regression equation (60), we can use any of the LS and LAD methods described in Sections 5.4 and 5.5. Once the estimate $\widehat{\vartheta}$ is computed, we can find the vector of coefficients for the no-intercept regression, $\widehat{\beta} = V^r\widehat{\vartheta} \in \mathbb{R}^n$, where $V^r = (V_1, ..., V_r) \in \mathbb{R}^{n \times r}$ contains the first r right singular vectors of X.

We can also arrive at the regression equation (60) using a truncated SVD method from the field of inverse problems in applied mathematics. Let the matrix $X^r \in \mathbb{R}^{T\times r}$ be defined by a truncated SVD of the matrix X, such that $X^r \equiv U^r S^r (V^r)'$; under this definition, $U^r \in \mathbb{R}^{T\times r}$ and $V^r \in \mathbb{R}^{n\times r}$ are the first r columns of U and V, respectively, and $S^r \in \mathbb{R}^{r\times r}$ is a diagonal matrix whose entries are the r largest singular values of X. As follows from the theorem of Eckart and Young (1936), the matrix X^r is the closest rank r approximation of the matrix $X \in \mathbb{R}^{T\times n}$. To obtain the regression equation (60), we can use the fact that Z^r and X^r are related by $Z^r = X^r V^r$.

We can use the truncated SVD to represent the LS estimator $\widehat{\vartheta}$ in the regression (60), specifically by setting $\widehat{\vartheta} = (S^r)^{-1} (U^r)' Y$, which implies

$$\widehat{\beta} = V^r (S^r)^{-1} (U^r)' Y. \tag{61}$$

We call the estimator (61) *regularized LS using truncated SVD (RLS-TSVD)*. If $r = n$, then RLS-TSVD coincides with LS-SVD described in Section 5.4.1.[22]

Finally, we shall make three remarks. First, we can extend the results of this section to the non-linear case by re-formulating the non-linear regression model (49) in terms of the principal components Z^r and by estimating the model with the numerically stable methods such as those described in Section 5.6. Second, the principal component regression (60) is well suited to the shrinkage type of regularization methods without additional scaling: the lower the variance of a principal component, the larger is the corresponding regression coefficient and the more heavily such a coefficient is penalized by a regularization method. Finally, we should be careful with removing low variance principal components since they may contain important pieces of information. Hadi and Ling (1998) construct an artificial regression example with four principal components, for which the removal of the lowest variance principal component reduces the explanatory power of the regression dramatically: R^2 drops from 1.00 to 0.00. A safe strategy is to set $\overline{\kappa}$ to a very large number in order to rule out only cases of extremely collinear variables.

[22] A possible alternative to the truncated SVD is a truncated QR factorization method with pivoting of columns; see Eldén (2007), pp. 72-74. The latter method is used in Matlab to construct a powerful back-slash operator for solving linear systems of equations.

7 Numerical results

We now investigate the successfulness of the proposed strategies in enhancing the numerical stability of the stochastic simulation algorithm. In Section 7.1, we describe the methodology of our numerical analysis. In Sections 7.2 and 7.3, we present the results for the linear and non-linear regression methods, respectively. In Section 7.4, we discuss the sensitivity experiments.

7.1 Methodology

We have discussed a variety of approaches for enhancing the numerical stability of the stochastic simulation algorithm. To assess the above approaches, we distinguish the following representative methods. For the linear regression model, we consider four non-regularization methods (OLS, LS-SVD, LAD-PP, and LAD-DP) and four corresponding regularization methods (RLS-Tikhonov, RLS-TSVD, RLAD-PP, and RLAD-DP). The RLS-TSVD method is also a representative of the principal component approach. For the non-linear regression model, we consider one representative method, namely, the Levenberg-Marquart method. We use the above methods to compute the solutions in cases involving both unnormalized and normalized data, as well as both the ordinary and Hermite polynomial representations.

We proceed by eliminating the least successful methods along the numerical experiments. Specifically, we begin with a version of the model with the closed-form solution under a relatively short simulation length, $T = 3,000$. After solving this model using all methods, we discard any and all methods that proved to be too costly in terms of memory and time. We subsequently apply the remaining methods to solving the model with partial depreciation of capital under a longer simulation length $T = 10,000$.

We parameterize the model $(2) - (4)$ in the way which is standard for macroeconomic literature. We assume the Cobb-Douglas production function, $f(k_t) = k_t^{\alpha}$ with $\alpha \in (0,1)$, and the CRRA utility function, $u(c_t) = \frac{c_t^{1-\gamma}-1}{1-\gamma}$ with $\gamma \in (0,\infty)$. The share of capital in production is equal to $\alpha = 0.36$, the parameters in the process for shocks (4) are set at $\rho = 0.95$ and $\sigma = 0.01$, and the discount factor is equal to $\delta = 0.99$. We consider two values of the depreciation rate, namely, $d = 1$ and $d = 0.02$. Under the former value, we set the coefficient of relative risk aversion at $\gamma = 1$, which leads to the model with the closed-form solution, and under the latter value, we consider three alternative values for the coefficient of relative risk aversion,

namely, $\gamma \in \{0.1, 1, 10\}$. All the experiments are performed with an identical sequence of shocks: we fix a sequence of shocks of length $T = 10,000$, and we use the first $3,000$ observations of the sequence when computing experiments with $T = 3,000$.[23]

In all experiments, we intended to compute polynomial approximations up to the fifth order; however, the algorithm failed to deliver high-order polynomial approximations in some experiments. To compute the polynomial approximation of order $m = 1$, we start iterations from a low-accuracy solution; to compute the polynomial approximations of order $m \geq 2$, we start from the polynomial approximation of order $m - 1$. We set the dampening parameter at $\mu = 0.1$ when the capital decision rule is parameterized, and we set it at $\mu = 0.5$ when the marginal utility of consumption is parameterized. In the model with the closed-form solution, we use the convergence criterion (9) with $\omega = 9$. In the model with partial depreciation of capital, a tight criterion does not help improve solution accuracy but does increase the computational costs, so we use the less strict convergence criterion of $\omega = 6$.

For each computational experiment, we report the time (in seconds) required to compute a solution, CPU. To compute the Hermite polynomial terms, we use the explicit formulas shown in (22). As a result, a construction of X from the ordinary and Hermite polynomials takes us approximately the same time, and the differences in computational time between the two polynomial representations reflect essentially the differences in the number of iterations.

To evaluate the accuracy of the obtained solutions, we use the Euler equation error test. To this purpose, we re-write the Euler equation (5) in a unit-free form,

$$e\left(k_t, \theta_t\right) \equiv E_t\left[\frac{c_{t+1}^{-\gamma}}{c_t^{-\gamma}}\left(1 - d + \alpha\theta_{t+1}k_{t+1}^{\alpha-1}\right)\right] - 1, \tag{62}$$

where $e\left(k_t, \theta_t\right)$ is the Euler equation error in (k_t, θ_t). Given that $e\left(k_t, \theta_t\right) = 0$ for all (k_t, θ_t) in the true solution, we can measure the accuracy of a candidate solution by examining the degree to which $e\left(k_t, \theta_t\right)$ differs from zero.

[23] In our experiments, the length of simulations was not a determinant factor of accuracy. In some model economies, Euler equation errors were of order 10^{-9} under $T = 3,000$ and in other model economies, they were of order 10^{-4} independently of whether we use $T = 3,000$ or $T = 100,000$. To increase accuracy of Monte Carlo integration by one order of magnitude, we should increase T by a factor of 100; see Judd (1998), pp. 292, for a discussion.

To implement the test, we draw a new sequence of shocks with length $T = 1,000$, and we use the decision rules to simulate the time series $\{k_{t+1}, c_t\}_{t=0}^{1000}$. For each (k_t, θ_t), we compute $e(k_t, \theta_t)$ by evaluating the conditional expectation in (62) with a ten-point Gauss-Hermite quadrature rule; see Judd (1998), pp. 261-263 for details. We then find the average absolute and maximum absolute Euler equation errors, $e_{mean} = \frac{1}{T}\sum_{t=1}^{1,000} |e(k_t, \theta_t)|$, and $e_{max} = \max\left(\{|e(k_t, \theta_t)|\}_{t=1}^{1,000}\right)$, respectively.

We run the computational experiments on a desktop computer ASUS with Intel(R) Core(TM)2 Quad CPU Q9400 (2.66 GHz). Our programs are written in Matlab. We compute SVD using the LAPACK package which is built in Matlab. To solve the linear programming problems, we use the routine "linprog" under the option of an interior-point method.[24] For the non-linear regression model, we use the NLLS routine "nlinfit," which implements the Levenberg-Marquart method.

7.2 Linear regression methods

In this section, we present the results of the various linear regression approaches. We first study the model with the closed-form solution, then turn our attention to the model with partial depreciation of capital.

7.2.1 Model with the closed-form solution

We describe the results for the model with the closed-form solution in Tables 2 and 3. In Table 2, we compare the performance of four non-regularization methods (OLS, LS-SVD, LAD-PP and LAD-DP) using both unnormalized and normalized data as well as both ordinary and Hermite polynomial representations.

OLS With unnormalized data and ordinary polynomials, OLS is able to compute the solutions only up to the second-order polynomials (OLS case I); under higher orders of polynomials, the stochastic simulation algorithm

[24] A possible alternative to the interior-point method is a simplex method. Our experiments indicated that the simplex method, incorporated in Matlab, was slower than the interior-point method; occasionally, it was also unable to find an initial guess. See Portnoy and Koenker (1997) for a comparison of interior-point and simplex-based algorithms in computing LAD estimates.

either breaks down or cycles. The failure of OLS to compute the solutions under high-order polynomials is the consequence of $X'X$ being ill-conditioned. The data normalization improves the performance of the OLS method, however, we still cannot calculate more than a third-order polynomial approximation (OLS, case III). The introduction of Hermite polynomials completely resolves the ill-conditioning of the LS problem (OLS, cases II and IV): After Hermite polynomials are introduced, OLS can compute all five degrees of the polynomial approximations, and the accuracy of these approximations improves systematically as we move from the first- to the fifth-order polynomials. For example, the average Euler equation error e_{mean} decreases from 10^{-4} to 10^{-9}. Data normalization has no visible effect on the outcome of the Hermite polynomials case; this is because the Hermite polynomial terms are, by construction, already expressed in comparable units (compare OLS, cases II and IV).

LS-SVD This method produces virtually the same solutions, regardless of whether the variables are normalized or not and whether ordinary or Hermite polynomial representations are used. The LS-SVD method is capable of computing the solutions for all five degrees of polynomials with a very high degree of accuracy and with very low computational time. While the case of LS-SVD with unnormalized data and ordinary polynomials suffers a bit from the ill-conditioning of the LS problem - namely, its fifth-order polynomial approximation is slightly less accurate than forth-order polynomial approximation (LS-SVD, case I) - the remaining LS-SVD experiments do not suffer from the effects of ill-conditioning at all.

LAD-PP LAD-PP with unnormalized data and ordinary polynomials (LAD-PP, case I) can deliver solutions for all five degrees of polynomials; hence, it performs better than its OLS counterpart (OLS, case I). The solution for the fifth-order polynomial, however, is slightly worse than that for the forth-order polynomial; this shows that LAD-PP with unnormalized data and ordinary polynomials still suffers from the ill-conditioning problem, specifically, from the scaling problem. Indeed, when the variables were normalized, the accuracy of the solution for the fifth-order polynomial approximation becomes better than that of the fifth-order polynomial approximation. Using Hermite polynomials, instead of ordinary polynomials, allows us to visibly improve the accuracy of the solutions (compare LAD-PP, cases I and II and also,

cases III and IV).

LAD-DP In the ordinary polynomials cases, LAD-DP is somewhat less numerically stable than LAD-PP: It can only compute the approximations up to the third degree when presented with unnormalized data (LAD-DP, case I), and up to the fourth degree when presented with normalized data (LAD-DP, case III). In the Hermite polynomials case, LAD-DP does not seem to suffer from the above problems and performs very similarly to LAD-PP. Nonetheless, an advantage of LAD-DP over LAD-PP is its considerably smaller computational time.

Regularization methods In Table 3, we present the results obtained under four regularization methods: RLS-Tikhonov, RLAD-PP, RLAD-DP, and RLS-TSVD. Normalized data was used for all four regularization methods. To illustrate on the degree to which solution accuracy depends on the size of the regularization parameter for each method, we consider two values of the regularization parameter: In RLS-TSVD, the largest condition number allowed, $\overline{\kappa}$, is set equal to 10^8 and 10^4, and in the other three methods, the penalty on the size of the regression coefficients, η, is set equal to 10^{-4} and 10^{-2}. We chose these values arbitrary and hence cannot draw a rigorous comparison of accuracy across the methods. For each of our methods, we can find a value of the regularization parameter that leads to a more accurate solution than the one reported in the table.[25]

As we can see from Table 3, the regularization methods under consideration can compute the solutions under all five degrees of polynomials and deliver a level of accuracy that is comparable or superior to that of the non-regularization methods shown in Table 2. For the shrinkage methods (RLS-Tikhonov, RLAD-PP, and RLAD-DP) an increase in η from 10^{-4} to 10^{-2} leads to the following changes: RLS-Tikhonov produces less accurate solutions under both the ordinary and Hermite polynomials, while RLAD-PP and RLAD-DP deliver less accuracy under the ordinary polynomials and almost the same accuracy under the Hermite polynomials. For RLS-TSVD, a decrease in $\overline{\kappa}$ from 10^8 to 10^4 results in the removal of relevant information

[25] To choose appropriate values of regularization parameters, one can employ a generalized cross-validation procedure, which is commonly used in statistics; see, e.g., Brown (1993), pp. 62-71. In our case, one can choose values that minimize the mean or maximum absolute Euler equation errors on a simulated path.

from the regression, which causes the accuracy of the solutions to decrease.

Lessons On the basis of our results for the model with the closed-form solution, we can make the following conclusions. First, in the case of unnormalized data and ordinary polynomials, all of the proposed methods produce more accurate and numerically stable solutions than the standard OLS method. Second, data normalization consistently improves the accuracy of the solutions in cases of ordinary polynomials while leaving solution accuracy unaffected when Hermite polynomials are used; in the subsequent experiments on the linear regression model, we thus use only normalized data. Finally, our primal-problem formulations LAD-PP and RLAD-PP require operations on matrices with $(2T + n) \times T$ and $(2T + 2n) \times T$ dimensions, respectively, which makes these formulations both memory intensive and slow to compute. In particular, when T exceeds $3,000$, our computer ran out of memory. One can potentially reduce the cost of these methods by a more efficient management of computer memory but we do not pursue this direction. When we move to the model with partial depreciation of capital, we thus take LAD-PP and RLAD-PP out of consideration and concentrate on the dual formulations LAD-DP and RLAD-DP which are not so costly in terms of memory and time.

7.2.2 Model with partial depreciation of capital

Table 4 contains the results for the model with partial depreciation of capital; such model does not admit a closed-form solution. We compare four methods: OLS, RLS-Tikhonov, RLAD-DP and RLS-TSVD. When using the RLS-Tikhonov and RLAD-DP methods, we set the regularization parameter η equal to 10^{-3}; and in RLS-TSVD method, we set the regularization parameter $\overline{\kappa}$ equal to 10^6. For each method, we report the results under both ordinary and Hermite polynomial representations, as well as under three different values of the coefficient of relative risk aversion $\gamma \in \{0.1, 1, 10\}$.

Low and medium degrees of risk aversion In the case of $\gamma = 1$, we can distinguish the following regularities. Under the ordinary polynomial representation, OLS is able to compute the solutions up to the forth-order polynomial approximation ($\gamma = 1$, case I), while all the other methods are able to do so for all five polynomial approximations considered ($\gamma = 1$, cases

III, V and VII); solution accuracy is very similar across all four methods.[26] Under the Hermite polynomial representation, OLS is capable of computing the solutions for all five polynomial approximations ($\gamma = 1$, case II); furthermore, OLS, RLS-Tikhonov and RLS-TSVD are indistinguishable in accuracy and computational time ($\gamma = 1$, cases II, IV and VIII), while RLAD-DP is generally somewhat slower ($\gamma = 1$, case VI). The LS regularization methods (RLS-Tikhonov, RLS-TSVD) perform slightly better in terms of accuracy than the LAD regularization method (RLAD-DP). However, the accuracy comparisons between the regularization methods should be treated with caution as the accuracy of a regularization method depends on the value of the regularization parameter chosen. For all the methods considered, the highest accuracy is achieved under the second-order polynomial approximations. The introduction of higher order polynomial terms into the regressions does not improve solution accuracy. This is because the true decision rules are close to linear, which means that high-order polynomial terms have little explanatory power and serve effectively to increase the variance of the estimated coefficients. We observe essentially the same tendencies for $\gamma = 0.1$ as for $\gamma = 1$.

High degree of risk aversion: instability of learning The case of $\gamma = 10$, however, is different. No method is able to compute the solutions for all five polynomial approximations: the least successful method is RLAD-DP which can compute only the first-order polynomial approximation ($\gamma = 10$, cases V and VI), while the most successful method is RLS-Tikhonov which is able to calculate the forth-order polynomial approximation with sufficiently small Euler equation errors ($\gamma = 10$, cases III and IV). Our attempts to enhance the convergence by reducing the dampening parameter μ and by varying the regularization parameters did not lead to satisfactory results. Since our methods can handle problems with any degree of ill-conditioning, we conjecture that the convergence failures that occur when $\gamma = 10$ are explained by the instability of the learning process.

To explore this issue in more details, we perform additional experiments for the model in which $\gamma = 10$; namely, we parameterize the marginal-utility decision rule (20) instead of the capital decision rule (6). This modification restored the numerical stability of our methods (see the bottom panel of

[26] Without data normalization, OLS with the ordinary polynomials can only compute the solutions up to the third-order polynomial approximation.

Table 4) and allows the computation of the polynomial approximations up to the fifth degree in all but two instances. In the two exceptional cases, OLS and RLAD-DP with the Hermite polynomials, polynomial approximations can be computed up to degree four ($\gamma = 10$, cases I and VI). The LS methods achieve the highest accuracy under the third-order polynomial approximation with the exception of RLS-Tikhonov with ordinary polynomials; this method achieves the highest accuracy under the fifth-order polynomial approximation ($\gamma = 10$, case III). For RLAD-DP, the second-order polynomial approximation is most accurate. Although the parameterization of the marginal-utility decision rule allows us to compute high-order polynomial approximations, the accuracy of such approximations is actually lower than that of the second-order polynomial approximation under our benchmark parameterization of the capital decision rule.[27]

To gain intuition into why parameterizing the marginal-utility decision rule increases the chance of convergence compared to parameterizing the capital decision rule when $\gamma = 10$, we shall recall the following property of the solution. When the value of γ is high, the agent has strong preferences for consumption smoothing: She effectively chooses an optimal level of consumption and saves the reminder to use as next-period capital. In contrast, when the value of γ is low, the agent cares little about consumption smoothing: She decides on the optimal level of next-period capital and then varies consumption as necessary to absorb all fluctuations. Presumably, the agent has more chances to learn a rational expectation equilibrium (i.e. to achieve the convergence) when we parameterize a "rigid" decision rule that captures the key choice of the agent.

7.3 Comparison of the polynomial and exponentiated polynomial specifications

In Section 3, we argued that the exponentiated polynomial specification (13) is not necessarily preferable to the polynomial specification (11), except in the case of the model with the closed-form solution. In this section, we perform a comparison of the numerical solutions obtained under the two polynomial specifications. In Table 5, we report the results for the model with

[27] We find that for other values of γ, the parameterization of the marginal-utility decision rule also produces less accurate solutions than the parameterization of the capital decision rule.

partial depreciation of capital under the ordinary and Hermite polynomials for three values of the coefficient of relative risk aversion, $\gamma \in \{0.1, 1, 10\}$. To ensure that our comparison is not affected by our choice of a regression method, we estimate the linear and non-linear regression models (14) and (12), respectively, with the same NLLS Levenberg-Marquart method.

Polynomial versus exponentiated polynomial specifications In our experiments, the polynomial specification of Ψ proved to be at least as good as the exponentiated polynomial specification advocated in Den Haan and Marcet (1990). In particular, both when $\gamma = 1$ and $\gamma = 0.1$, the two polynomial specifications are very close in accuracy, with the highest accuracy achieved under the second-order polynomial approximation. Given $\gamma = 10$ and assuming the parameterization of the capital decision rule, the polynomial specification is superior to the exponentiated polynomial specification as it allows us to go beyond the first-order polynomial approximation. In turn, when $\gamma = 10$ and the marginal-utility decision rule is parameterized, the exponentiated polynomial specification ensures faster convergence and more accurate solutions than the polynomial specification.

Linear versus non-linear regression methods So far, we have compared the polynomial and exponentiated polynomial specifications under the same NLLS Levenberg-Marquart regression method. We can also use the results shown in Tables 4 and 5 to compare the performance of the linear and non-linear regression methods under the same polynomial specification in the linear regression model (14). As seen in these tables, the Levenberg-Marquart method is as accurate and almost as fast as our linear LS regularization methods, and it is superior to OLS. The Levenberg-Marquart method is so successful in our experiments because it is designed to regularize the ill-conditioned LS problem in the same way as our linear regularization methods.

7.4 Sensitivity analysis

We perform a few additional sensitivity experiments. To economize on space, we omit the detailed results of these experiments and simply describe the observed regularities.

Modifying the explanatory variables We modify a set of explanatory variables that constitute the matrix X. Specifically, given the polynomial specification of Ψ (13), we construct X in terms of both $\ln k$ and $\ln \theta$ (i.e. $X = [1_T, \ln k, \ln \theta, ...]$) and also, in terms of k and $\ln \theta$ (i.e. $X = [1_T, k, \ln \theta, ...]$). In our numerical experiments, these two modifications did not lead to improvements in accuracy and/or speed relative to our benchmark construction of X in levels, $X = [1_T, k, \theta, ...]$, however, they are still worth considering in other applications.

Modifying the decision rule In addition to the capital decision rule in (6) and the marginal-utility decision rule in (20), we experiment with the parameterization of other decision rules. In particular, we pre-multiply both sides of the Euler equation by $\ln k_{t+1}$ and parameterize the right hand side of the resulting equation to obtain a decision rule for $\ln k_{t+1}$ in terms of $\ln k_t$ and $\ln \theta_t$. (Note that in this way we can parameterize a decision rule for any t-period variable, e.g., $\ln c_t$ or $\exp(c_t)$). This construction parallels that of the non-linear regression model with the exponentiated polynomial (12), and it matches the closed-from solution (10). We find that this parameterization works only under low values of γ, and it leads to cycling under $\gamma \geq 1$. Overall, we have not found any parameterization that dominates the capital and marginal-utility parametizations in terms of accuracy and speed though such a parameterization may in fact exist.

Modifying the polynomial representation Instead of the Hermite polynomials, we use the Chebyshev polynomials. Like the Hermite basis functions, the Chebyshev basis functions look different one from another and hence, decrease the likelihood of the multicollinearity problem compared to ordinary polynomials. We find that the results under the Chebyshev and Hermite polynomials are virtually identical, which suggests that other orthogonal polynomial representations (e.g., Legendre and Laguerre) will also perform well in the context of the stochastic simulation algorithm.

Modifying the non-linear regression method Instead of the Levenberg-Marquart method, we use the standard Gauss-Newton method given in (52), which does not regularize the LS problem. We find that the Gauss-Newton method is considerably slower and less numerically stable than the Levenberg-Marquart method. We do not explore the performance of the non-linear ver-

sions of the LAD method and the non-linear principal component approach described in Section 5.6. We leave this for further research.

8 Conclusion

Standard LS methods, such as OLS and Gauss-Newton methods, fail when confronted with ill-conditioned problems. This failure severely handicaps the performance of existing stochastic simulation algorithms. The present paper contributes to the literature by suggesting a novel way of increasing the numerical stability and high accuracy of the stochastic simulation approach through the use of more powerful and stable approximation methods. The main lessons that can be learned from our analysis are the following: First, normalize the variables, as it never hurts. Second, look for basis polynomial functions that do not automatically give multicollinearity. Third, use approximation methods that can handle ill-conditioned problems. Fourth, apply the unified principal component method if degrees of ill-conditioning are very high. Finally, there is no general rule about which decision function to parameterize in each case; use inferences from the economic theory.

The message of this paper is more broad than selecting one or several particular methods that perform the best. We show that there are many alternative computational strategies that can substantially improve the performance of stochastic simulation methods. We should be aware about the existence of these various alternatives, and should select the alternative that is most suited for the application considered. We provide a code that makes it easy to try them all. For our simple model with two state variables, all the proposed methods work well and any method is sufficient. The ideas developed in the present paper do not depend on the dimension of a problem, however. The real test will come with high-dimensional applications, in which some methods might work better than others. It is too early to make a decision.

References

[1] Aiyagari, R., (1994). Uninsured idiosyncratic risk and aggregate saving. Quarterly Journal of Economics 109, 659-684.

[2] Asmussen, S. and P. W. Glynn, (2007). Stochastic Simulation: Algorithms and Analysis. Springer, New York.

[3] Basett, G. and R. Koenker, (1978). Asymptotic theory of least absolute error regression. Journal of American Statistical Association 73, 618-622.

[4] Bjorke, A., (1996). Numerical Methods for Least Squares Problems. SIAM, Philadelphia.

[5] Brown, P., (1993). Measurement, Regression, and Calibration. Claredon Press, Oxford.

[6] Charnes, A., Cooper W. and R. Ferguson, (1955). Optimal estimation of executive compensation by linear programming. Management Science 1, 138-151.

[7] Christiano, L. and D. Fisher., (2000). Algorithms for solving dynamic models with occasionally binding constraints. Journal of Economic Dynamics and Control 24, 1179-1232.

[8] Creel, M., (2008). Using parallelization to solve a macroeconomic model: a parallel parameterized expectations algorithm. Computational Economics 32, 343-352.

[9] Den Haan, W. and A. Marcet, (1990). Solving the stochastic growth model by parameterized expectations. Journal of Business and Economic Statistics 8, 31-34.

[10] Den Haan, W. and A. Marcet, (1994). Accuracy in simulations. Review of Economic Studies 61, 3-17.

[11] Dielman, T., (2005). Least absolute value: recent contributions. Journal of Statistical Computation and Simulation 75, 263-286.

[12] Eckart, C. and G. Young, (1936). The approximation of one matrix by another of lower rank, Psychometrika 1, 211-218.

[13] Eldén, L., (2007). Matrix Methods in Data Mining and Pattern Recognition. SIAM, Philadelphia.

[14] Fair, R. and J. Taylor, (1983). Solution and maximum likelihood estimation of dynamic nonlinear rational expectation models. Econometrica 51, 1169-1185.

[15] Ferris, M., Mangasarian, O. and S. Wright, (2007). Linear Programming with MATLAB. MPS-SIAM Series on Optimization, Philadelphia.

[16] Hadi, A. and R. Ling, (1998). Some cautionary notes on the use of principal components regression, American Statistician 52, 15-19.

[17] Hastie, T., Tibshirani, R. and J. Friedman, (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.

[18] Hoerl, A., (1959). Optimum solution of many variables, Chemical Engineering Progress 55, 67-78.

[19] Hoerl, A. and R. Kennard, (1970). Ridge regression: biased estimation for nonorthogonal problems, Technometrics 12, 69-82.

[20] Koenker, R. and G. Bassett, (1978). Regression quantiles, Econometrica 46, 33-50.

[21] Koenker, R. and K. Hallock, (2001). Quantile regression, Journal of Economic Perspectives 15, 143-156.

[22] Krueger, D. and F. Kubler, (2004). Computing equilibrium in OLG models with production, Journal of Economic Dynamics and Control 28, 1411-1436.

[23] Krusell, P. and A. Smith, (1998). Income and wealth heterogeneity in the macroeconomy. Journal of Political Economy 106(5), 868-896.

[24] Judd, K., (1992). Projection methods for solving aggregate growth models. Journal of Economic Theory 58, 410-52.

[25] Judd, K., (1998). Numerical Methods in Economics. MIT Press, Cambridge, MA.

[26] Judd, K. and J. Gaspar (1997). Solving large scale rational expectations models, Macroeconomic Dynamics 1, 44-75.

[27] Levenberg, K., (1944). A method for the solution of certain nonlinear problems in least squares, Quarterly Applied Mathematics 4, 164–168.

[28] Maliar, L. and Maliar, S., (2003a). The representative consumer in the neoclassical growth model with idiosyncratic shocks, Review of Economic Dynamics 6, 362-380.

[29] Maliar, L. and Maliar, S., (2003b). Parameterized expectations algorithm and the moving bounds, Journal of Business and Economic Statistics, 21, 88-92.

[30] Maliar L. and S. Maliar, (2005). Solving nonlinear stochastic growth models: iterating on value function by simulations, Economics Letters 87, 135-140.

[31] Marcet, A., (1988). Solution of nonlinear models by parameterizing expectations, Carnegie Mellon University, manuscript.

[32] Marcet, A., and Lorenzoni, G. (1999). The parameterized expectation approach: some practical issues, in: R. Marimon and A. Scott (Eds.), Computational Methods for Study of Dynamic Economies, Oxford University Press, New York, pp. 143-171.

[33] Marimon, R. and A. Scott, (1999). Computational Methods for Study of Dynamic Economies, Oxford University Press, New York.

[34] Marquardt, D., (1963). An algorithm for least-squares estimation of nonlinear parameters, Journal of Society for Industrial Applied Mathematics 11, 431–441.

[35] Michelangeli, V., (2008). Does it pay to get a reverse mortgage? Boston University, manuscript.

[36] Narula, S. and J. Wellington, (1982). The minimum sum of absolute errors regression: a state of the art survey, International Statistical Review 50, 317-326.

[37] Portnoy, S. and R. Koenker, (1997). The Gaussian hare and the Laplacian tortoise: Computability of squared error versus absolute-error estimators. Statistical Science 12, 279-296.

[38] Santos, M., (1999). Numerical solution of dynamic economic models, in: J. Taylor and M. Woodford (Eds.), Handbook of Macroeconomics, Amsterdam: Elsevier Science, pp.312-382.

[39] Smith, A., (1993). Estimating nonlinear time-series models using simulated vector autoregressions. Journal of Applied Econometrics 8, S63-S84.

[40] Taylor, J. and H. Uhlig, (1990). Solving nonlinear stochastic growth models: a comparison of alternative solution methods. Journal of Business and Economic Statistics 8(1), 1-17.

[41] Tibshirani, R., (1996). Regression shrinkage and selection via the lasso. Journal of Royal Statistical Society Series B 58, 267-288.

[42] Tikhonov, A., (1963). Solutions of incorrectly formulated problems and the regularization method, Soviet Math Dokl 4, 1035-1038. English translation of Dokl Akad Nauk SSSR 151, (1963), 501-504.

[43] Wagner, H., (1959). Linear programming techniques for regression analysis, American Statistical Association Journal 54, 206-212.

[44] Wang, L., Gordon, M. and J. Zhu, (2006). Regularized least absolute deviations regression and an efficient algorithm for parameter tuning. Proceedings of the Sixth International Conference on Data Mining, 690–700.

Figure 1. The ergodic distribution in the model with a closed-form solution.

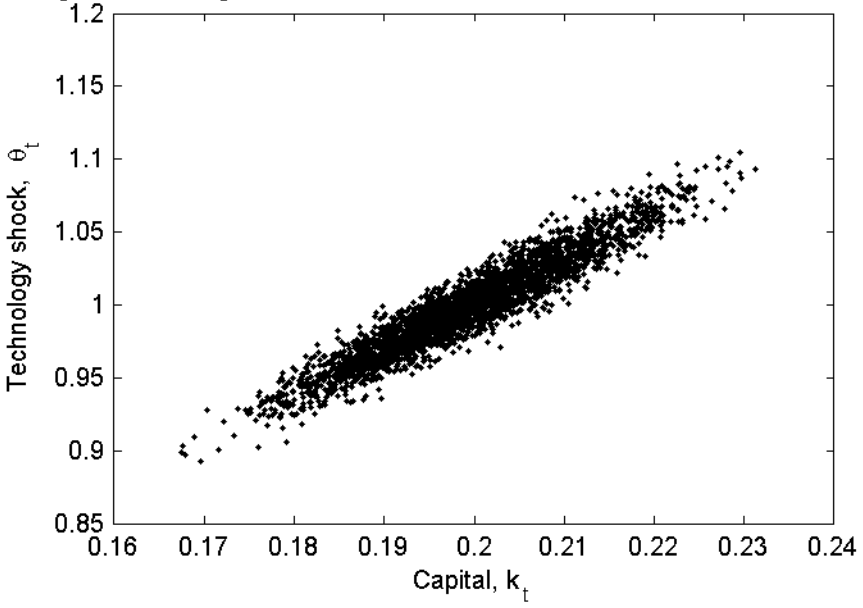

Figure 2. The ratio of hypershere to hypercube volume.

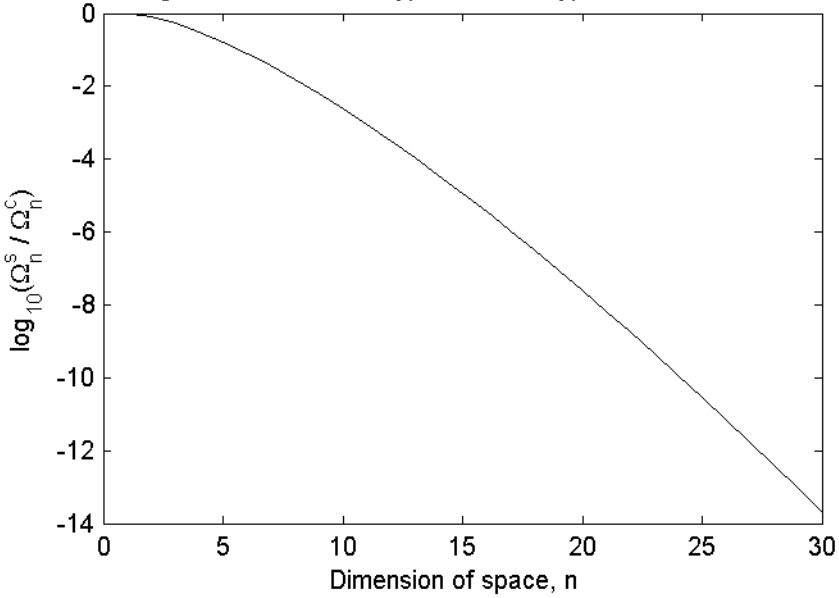

Figure 3. Ordinary polynomials.

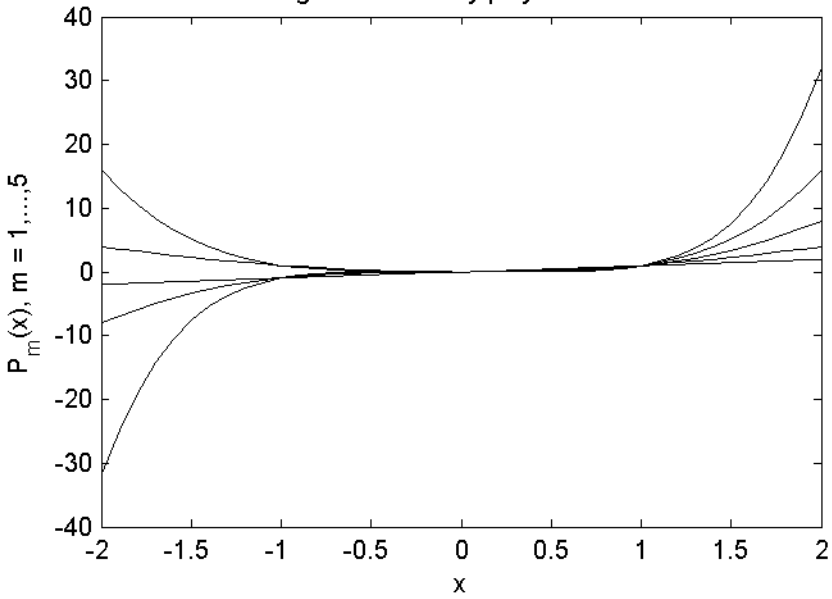

Figure 4. Hermite polynomials.

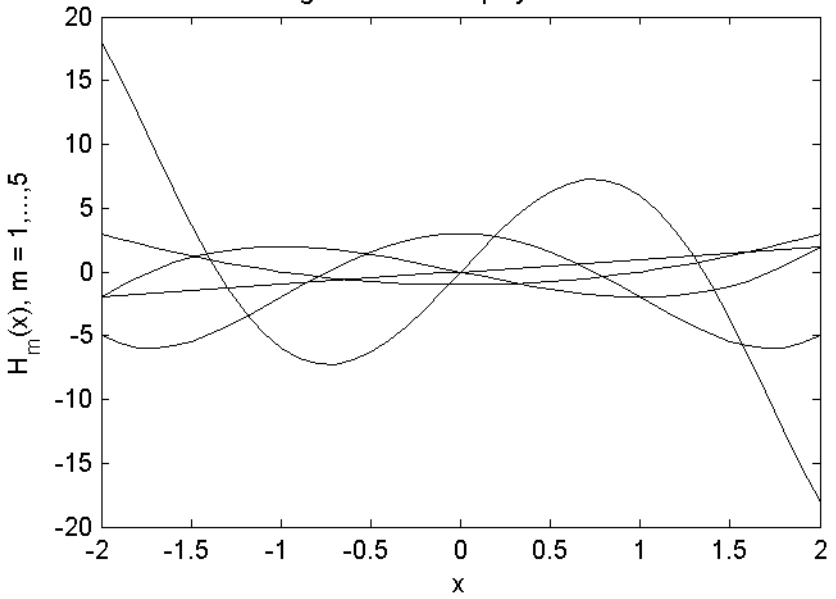

Table 1. The ratio of hypershere to hypercube volume.

Dimension, n	2	3	4	5	6	7	8	9	10	15	20	25	30
Ω_n^s / Ω_n^c	0,79	0,52	0,31	0,16	8(-2)	4(-2)	2(-2)	6(-3)	3(-3)	1(-5)	3(-8)	3(-11)	2(-14)
$log_{10} (\Omega_n^s / \Omega_n^c)$	-0,10	-0,28	-0,51	-0,78	-1,09	-1,43	-1,80	-2,19	-2,60	-4,93	-7,61	-10,54	-13,69

Remark: Ω_n^s and Ω_n^c denote the volumes of hypersphere and hypercube, respectively. The notation $x(m)$ means $x \cdot 10^m$.

Table 2. Solving the model with a closed-form solution using the non-regularization methods.

Polynomial Order	Unnormalized						Normalized					
	Ordinary polynomials			Hermite polynomials			Ordinary polynomials			Hermite polynomials		
	e_{mean}	e_{max}	CPU	e_{mean}	e_{max}	CPU	e_{mean}	e_{max}	CPU	e_{mean}	e_{max}	CPU
	Case I			Case II			Case III			Case IV		
OLS												
1st order	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	1(-1)	3.29(-4)	3.35(-3)	2(-1)
2nd order	3.92(-6)	8.39(-5)	4	3.92(-6)	8.38(-5)	5	3.92(-6)	8.38(-5)	5	3.92(-6)	8.38(-5)	5
3rd order	-	-	-	1.71(-7)	5.99(-6)	3	1.71(-7)	5.93(-6)	4	1.71(-7)	5.99(-6)	3
4th order	-	-	-	1.47(-8)	1.07(-6)	1	-	-	-	1.47(-8)	1.07(-6)	1
5th order	-	-	-	5.28(-9)	4.03(-7)	2(-1)	-	-	-	5.17(-9)	4.15(-7)	2(-1)
LS-SVD												
1st order	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	2(-1)
2nd order	3.92(-6)	8.38(-5)	5	3.92(-6)	8.38(-5)	5	3.92(-6)	8.38(-5)	5	3.92(-6)	8.38(-5)	5
3rd order	1.71(-7)	5.99(-6)	3	1.71(-7)	5.99(-6)	3	1.71(-7)	5.99(-6)	3	1.71(-7)	5.99(-6)	3
4th order	1.34(-8)	9.82(-7)	2	1.47(-8)	1.07(-6)	1	1.47(-8)	1.07(-6)	1	1.47(-8)	1.07(-6)	1
5th order	2.11(-8)	1.49(-7)	4	5.15(-9)	4.05(-7)	2(-1)	5.58(-9)	4.13(-7)	2(-1)	5.15(-9)	4.05(-7)	2(-1)
LAD-PP												
1st order	2.95(-4)	3.58(-3)	1(2)	2.95(-4)	3.58(-3)	2(2)	3.29(-4)	3.36(-3)	8(1)	3.29(-4)	3.36(-3)	8(1)
2nd order	3.40(-6)	9.26(-5)	5(2)	3.40(-6)	9.26(-5)	5(2)	3.39(-6)	9.31(-5)	2(2)	3.39(-6)	9.31(-5)	2(2)
3rd order	1.40(-7)	7.86(-6)	3(2)	1.40(-7)	7.87(-6)	3(2)	1.41(-7)	7.71(-6)	3(2)	1.41(-7)	7.71(-6)	3(2)
4th order	1.26(-8)	1.33(-6)	1(2)	1.15(-8)	1.53(-6)	1(2)	1.19(-8)	1.63(-6)	1(2)	1.19(-8)	1.63(-6)	1(2)
5th order	2.69(-8)	2.17(-6)	2(2)	5.85(-9)	6.21(-7)	2(1)	5.03(-9)	6.77(-7)	2(1)	4.90(-9)	6.85(-7)	2(1)
LAD-DP												
1st order	2.95(-4)	3.58(-3)	1(1)	2.95(-4)	3.58(-3)	1(1)	3.29(-4)	3.36(-3)	5(1)	3.29(-4)	3.36(-3)	5(1)
2nd order	3.40(-6)	9.27(-5)	1(1)	3.40(-6)	9.27(-5)	1(1)	3.39(-6)	9.31(-5)	6(1)	3.39(-6)	9.31(-5)	6(1)
3rd order	1.40(-7)	7.83(-6)	8	1.40(-7)	7.86(-6)	7	1.41(-7)	7.71(-6)	4(1)	1.41(-7)	7.71(-6)	4(1)
4th order	-	-	-	1.13(-8)	1.45(-6)	4	1.33(-8)	1.76(-6)	3(1)	1.19(-8)	1.63(-6)	2(1)
5th order	-	-	-	4.35(-9)	5.64(-7)	4(-1)	-	-	-	4.91(-9)	6.85(-7)	3

Remark: e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively, and CPU is computational time in seconds. The notation $x(m)$ means $x \cdot 10^m$.

Table 3. Solving the model with a closed-form solution using the regularization methods.

Polynomial order	Ordinary polynomials			Hermite polynomials			Ordinary polynomials			Hermite polynomials		
	e_{mean}	e_{max}	CPU	e_{mean}	e_{max}	CPU	e_{mean}	e_{max}	CPU	e_{mean}	e_{max}	CPU
	Case I			Case II			Case III			Case IV		
RLS-Tikhonov with η = *1(-4)*							RLS-Tikhonov with η = *1(-2)*					
1st order	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	9(-1)	3.29(-4)	3.35(-3)	1
2nd order	3.94(-6)	8.37(-5)	5	3.92(-6)	8.38(-5)	5	2.40(-5)	6.57(-4)	5	3.97(-6)	8.35(-5)	5
3rd order	2.71(-6)	6.10(-5)	2	1.81(-7)	5.95(-6)	3	1.40(-5)	3.33(-4)	3	2.47(-6)	2.40(-4)	3
4th order	1.87(-6)	4.31(-5)	3	2.89(-8)	1.19(-6)	1	1.12(-5)	2.18(-4)	3	2.52(-6)	1.39(-4)	2
5th order	1.58(-6)	4.17(-5)	2	4.54(-7)	2.07(-4)	2	8.85(-6)	1.69(-4)	3	7.44(-6)	1.89(-3)	4
RLAD-PP with η = *1(-4)*							RLAD-PP with η = *1(-2)*					
1st order	3.29(-4)	3.36(-3)	8(1)	3.29(-4)	3.36(-3)	8(1)	3.29(-4)	3.36(-3)	8(1)	3.29(-4)	3.36(-3)	1(2)
2nd order	3.39(-6)	9.31(-5)	2(2)	3.39(-6)	9.31(-5)	2(2)	3.39(-6)	9.29(-5)	3(2)	3.39(-6)	9.31(-5)	4(2)
3rd order	1.41(-7)	7.69(-6)	4(2)	1.41(-7)	7.71(-6)	4(2)	1.45(-7)	7.08(-6)	4(2)	1.41(-7)	7.71(-6)	3(3)
4th order	1.15(-8)	1.59(-6)	2(2)	1.19(-8)	1.63(-6)	2(2)	2.30(-7)	1.16(-5)	3(2)	1.19(-8)	1.63(-6)	1(3)
5th order	1.00(-8)	8.16(-7)	4(1)	4.90(-9)	6.84(-7)	3(1)	1.94(-7)	8.62(-6)	7(2)	4.93(-9)	6.85(-7)	9(1)
RLAD-PP with η = *1(-4)*							RLAD-PP with η = *1(-2)*					
1st order	3.29(-4)	3.36(-3)	1(1)	3.29(-4)	3.36(-3)	1(1)	3.29(-4)	3.36(-3)	1(1)	3.29(-4)	3.36(-3)	1(1)
2nd order	3.39(-6)	9.31(-5)	2(1)	3.39(-6)	9.31(-5)	2(1)	3.39(-6)	9.29(-5)	2(1)	3.39(-6)	9.31(-5)	2(1)
3rd order	1.41(-7)	7.69(-6)	2(1)	1.41(-7)	7.71(-6)	2(1)	1.45(-7)	7.08(-6)	2(1)	1.41(-7)	7.71(-6)	2(1)
4th order	1.15(-8)	1.59(-6)	1(2)	1.19(-8)	1.63(-6)	1(1)	2.30(-7)	1.16(-5)	3	1.19(-8)	1.63(-6)	1(1)
5th order	1.00(-8)	8.18(-7)	6	4.91(-9)	6.85(-7)	6	1.94(-7)	8.62(-6)	5	4.93(-9)	6.85(-7)	6
LS-TSVD with $\bar{\kappa}$ = *1(8)*							LS-TSVD with $\bar{\kappa}$ = *1(4)*					
1st order	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	2(-1)	3.29(-4)	3.35(-3)	2(-1)
2nd order	3.92(-6)	8.38(-5)	5	3.92(-6)	8.38(-5)	5	3.92(-6)	8.38(-5)	5	3.92(-6)	8.38(-5)	5
3rd order	1.71(-7)	5.99(-6)	3	1.71(-7)	5.99(-6)	3	5.91(-6)	1.16(-4)	2	1.71(-7)	5.99(-6)	3
4th order	1.47(-8)	1.07(-6)	1	1.47(-8)	1.07(-6)	2	2.53(-6)	6.02(-5)	3	1.47(-8)	1.07(-6)	1
5th order	5.12(-9)	4.04(-7)	2(-1)	5.15(-9)	4.05(-7)	2(-1)	3.59(-6)	8.62(-5)	2	4.93(-7)	2.31(-4)	2(-1)

Remark: e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively, and CPU is computational time in seconds. The notation *x(m)* means $x \cdot 10^m$.

Table 4. Solving the model with partial depreciation of capital.

Polynomial order	OLS						RLS-Tikhonov with η = *1(-3)*					
	Ordinary polynomials			Hermite polynomials			Ordinary polynomials			Hermite polynomials		
	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*
	Case I			Case II			Case III			Case IV		
γ = *1*												
1st order	5.52(-5)	3.03(-4)	7(-1)	5.52(-5)	3.03(-4)	7(-1)	5.52(-5)	3.03(-4)	7(-1)	5.52(-5)	3.03(-4)	7(-1)
2nd order	3.99(-5)	1.97(-4)	9	3.99(-5)	1.97(-4)	1(1)	3.99(-5)	1.97(-4)	1(1)	3.99(-5)	1.97(-4)	1(1)
3rd order	5.11(-5)	3.27(-4)	1(1)	5.11(-5)	3.27(-4)	1(1)	4.95(-5)	3.23(-4)	1(1)	5.11(-5)	3.27(-4)	1(1)
4th order	5.40(-5)	8.99(-4)	1(1)	5.40(-5)	8.99(-4)	1(1)	5.09(-5)	3.26(-4)	9	5.40(-5)	8.99(-4)	1(1)
5th order	-	-	-	7.10(-5)	5.11(-4)	1(1)	5.08(-5)	3.17(-4)	9	7.10(-5)	5.10(-4)	1(1)
γ = *0.1*												
1st order	2.10(-5)	2.54(-4)	5(-1)	2.10(-5)	2.54(-4)	5(-1)	2.10(-5)	2.54(-4)	5(-1)	2.10(-5)	2.54(-4)	5(-1)
2nd order	1.72(-5)	1.70(-4)	2(1)	1.72(-5)	1.70(-4)	2(1)	1.72(-5)	1.70(-4)	2(1)	1.72(-5)	1.70(-4)	2(1)
3rd order	2.21(-5)	1.24(-4)	3(1)	2.21(-5)	1.24(-4)	4(1)	1.83(-5)	1.72(-4)	2(1)	2.21(-5)	1.25(-4)	4(1)
4th order	2.81(-5)	8.03(-4)	5(1)	2.81(-5)	8.04(-4)	4(1)	2.07(-5)	1.66(-4)	3(1)	2.81(-5)	8.05(-4)	4(1)
5th order	-	-	-	3.32(-5)	7.96(-4)	4(1)	2.16(-5)	1.54(-4)	2(1)	3.32(-5)	7.96(-4)	3(1)
γ = *10* (capital decision function with μ=*0.1*)												
1st order	1.19(-3)	8.38(-3)	1	1.19(-3)	8.38(-3)	1	1.19(-3)	8.38(-3)	1	1.19(-3)	8.38(-3)	2
2nd order	4.30(-4)	7.68(-4)	1(1)	4.30(-4)	7.68(-4)	1(1)	4.30(-4)	7.69(-4)	9	4.30(-4)	7.68(-4)	1(1)
3rd order	-	-	-	-	-	-	5.22(-4)	4.58(-3)	8	-	-	-
4th order	-	-	-	-	-	-	5.44(-4)	3.72(-3)	4(1)	-	-	-
5th order	-	-	-	-	-	-	-	-	-	-	-	-
γ = *10* (marginal utility decision function with μ=*0.5*)												
1st order	1.62(-3)	7.73(-3)	4(2)	1.62(-3)	7.73(-3)	4(2)	1.62(-3)	7.73(-3)	4(2)	1.62(-3)	7.73(-3)	5(2)
2nd order	8.05(-4)	2.95(-3)	2(2)	8.05(-4)	2.95(-3)	2(2)	8.02(-4)	2.91(-3)	2(2)	8.05(-4)	2.95(-3)	2(2)
3rd order	5.84(-4)	3.13(-3)	2(2)	5.84(-4)	3.13(-3)	2(2)	8.08(-4)	2.30(-3)	2(2)	5.84(-4)	3.13(-3)	2(2)
4th order	8.07(-4)	8.97(-3)	1(2)	8.07(-4)	8.97(-3)	1(2)	6.44(-4)	2.62(-3)	1(2)	8.07(-4)	8.97(-3)	1(2)
5th order	-	-	-	8.48(-4)	8.29(-3)	8(1)	6.34(-4)	3.16(-3)	8(1)	8.48(-4)	8.29(-3)	8(1)

Remark: e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively, and *CPU* is computational time in seconds. The notation *x(m)* means $x \cdot 10^m$.

Table 4. Solving the model with partial depreciation of capital (continued).

Polynomial order	RLAD-DP with η = 1(-3)						LS-TSVD with $\bar{\kappa}$ = 1(6)					
	Ordinary polynomials			Hermite polynomials			Ordinary polynomials			Hermite polynomials		
	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*
	Case V			Case VI			Case VII			Case VIII		
$\gamma = 1$												
1st order	5.36(-5)	2.76(-4)	2(1)	5.36(-5)	2.76(-4)	2(1)	5.52(-5)	3.03(-4)	7(-1)	5.52(-5)	3.03(-4)	7(-1)
2nd order	4.93(-5)	2.85(-4)	5(1)	4.93(-5)	2.85(-4)	6(1)	3.99(-5)	1.97(-4)	1(1)	3.99(-5)	1.97(-4)	1(1)
3rd order	6.09(-5)	2.85(-4)	7(1)	6.17(-5)	2.97(-4)	6(1)	5.11(-5)	3.27(-4)	1(1)	5.11(-5)	3.27(-4)	1(1)
4th order	6.18(-5)	2.85(-4)	7(1)	7.29(-5)	3.43(-4)	7(1)	5.40(-5)	8.99(-4)	1(1)	5.40(-5)	8.99(-4)	1(1)
5th order	6.82(-5)	2.84(-4)	2(2)	1.17(-4)	3.95(-4)	2(2)	5.38(-5)	8.87(-4)	5	7.10(-5)	5.11(-4)	1(1)
$\gamma = 0.1$												
1st order	2.03(-5)	2.39(-4)	6(1)	2.03(-5)	2.39(-4)	7(1)	2.10(-5)	2.54(-4)	5(-1)	2.10(-5)	2.54(-4)	6(-1)
2nd order	1.81(-5)	1.16(-4)	9(1)	1.81(-5)	1.16(-4)	9(1)	1.72(-5)	1.70(-4)	2(1)	1.72(-5)	1.70(-4)	2(1)
3rd order	2.52(-5)	1.15(-4)	3(2)	2.61(-5)	1.94(-4)	2(2)	2.21(-5)	1.24(-4)	4(1)	2.21(-5)	1.24(-4)	4(1)
4th order	2.56(-5)	2.10(-4)	3(2)	3.59(-5)	2.81(-4)	3(2)	2.34(-5)	1.97(-4)	4(1)	2.81(-5)	8.04(-4)	4(1)
5th order	2.69(-5)	2.25(-4)	3(2)	4.33(-5)	5.52(-4)	3(2)	2.33(-5)	1.90(-4)	2(1)	3.32(-5)	7.96(-4)	3(1)
$\gamma = 10$ (capital decision rule with $\mu=0.1$)												
1st order	1.04(-3)	7.48(-3)	2(1)	1.04(-3)	7.48(-3)	2(1)	1.19(-3)	8.38(-3)	1	1.19(-3)	8.38(-3)	1
2nd order	-	-	-	-	-	-	4.30(-4)	7.68(-4)	1(1)	4.30(-4)	7.68(-4)	1(1)
3rd order	-	-	-	-	-	-	-	-	-	-	-	-
4th order	-	-	-	-	-	-	-	-	-	-	-	-
5th order	-	-	-	-	-	-	-	-	-	-	-	-
$\gamma = 10$ (marginal utility decision rule with $\mu=0.5$)												
1st order	1.59(-3)	8.28(-3)	8(2)	1.59(-3)	8.28(-3)	9(2)	1.62(-3)	7.73(-3)	4(2)	1.62(-3)	7.73(-3)	5(2)
2nd order	7.10(-4)	1.86(-3)	4(2)	7.10(-4)	1.86(-3)	4(2)	8.05(-4)	2.95(-3)	2(2)	8.05(-4)	2.95(-3)	2(2)
3rd order	7.53(-4)	2.02(-3)	7(2)	7.62(-4)	2.13(-3)	5(2)	5.84(-4)	3.13(-3)	2(2)	5.84(-4)	3.13(-3)	2(2)
4th order	8.87(-4)	3.96(-3)	1(3)	9.58(-4)	3.48(-3)	8(2)	8.07(-4)	8.97(-3)	1(2)	8.07(-4)	8.97(-3)	1(2)
5th order	8.81(-4)	2.91(-3)	8(2)	-	-	-	8.73(-4)	1.23(-2)	8(1)	8.48(-4)	8.29(-3)	9(1)

Remark: e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively, and *CPU* is computational time in seconds. The notation *x(m)* means $x \cdot 10^m$.

Table 5. Solving the model with partial depreciation of capital under polynomial and exponentiated polynomial specifications with Levenberg-Marquart method.

Polynomial order	Exponentiated polynomial specification						Polynomial specification					
	Ordinary polynomials			Hermite polynomials			Ordinary polynomials			Hermite polynomials		
	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*
	Case I			Case II			Case III			Case IV		
$\gamma = 1$												
1st order	7.62(-5)	9.23(-4)	8	7.62(-5)	9.23(-4)	1(1)	5.52(-5)	3.03(-4)	7(-1)	5.52(-5)	3.03(-4)	8(-1)
2nd order	3.99(-5)	2.16(-4)	1(1)	4.02(-5)	2.26(-4)	1(1)	3.99(-5)	1.97(-4)	9	3.99(-5)	1.97(-4)	1(1)
3rd order	4.95(-5)	2.81(-4)	2(1)	5.07(-5)	3.22(-4)	1(1)	5.11(-5)	3.27(-4)	1(1)	5.11(-5)	3.27(-4)	1(1)
4th order	5.29(-5)	6.07(-4)	1(2)	5.47(-5)	9.13(-4)	1(1)	4.71(-5)	2.10(-4)	2(1)	5.40(-5)	8.99(-4)	1(1)
5th order	6.50(-5)	6.39(-4)	9(1)	7.16(-5)	5.30(-4)	2(1)	5.21(-5)	8.52(-4)	2(1)	7.10(-5)	5.11(-4)	2(1)
$\gamma = 0.1$												
1st order	1.96(-5)	9.33(-5)	8	1.98(-5)	9.45(-5)	7	2.10(-5)	2.54(-4)	5(-1)	2.10(-5)	2.54(-4)	5(-1)
2nd order	1.68(-5)	1.54(-4)	2(1)	1.71(-5)	1.69(-4)	2(1)	1.72(-5)	1.70(-4)	2(1)	1.72(-5)	1.70(-4)	2(1)
3rd order	2.20(-5)	1.29(-4)	4(1)	2.20(-5)	1.20(-4)	4(1)	2.21(-5)	1.24(-4)	3(1)	2.21(-5)	1.24(-4)	4(1)
4th order	2.44(-5)	2.35(-4)	2(2)	2.88(-5)	8.28(-4)	5(1)	2.23(-5)	1.14(-4)	2(1)	2.81(-5)	8.04(-4)	4(1)
5th order	2.97(-5)	3.28(-4)	3(2)	3.30(-5)	8.19(-4)	5(1)	2.48(-5)	1.40(-4)	6(1)	3.32(-5)	7.96(-4)	4(1)
$\gamma = 10$ (capital decision rule with $\mu=0.1$)												
1st order	1.03(-3)	1.06(-2)	3(1)	1.03(-3)	1.06(-2)	9	1.19(-3)	8.38(-3)	9(-1)	1.19(-3)	8.38(-3)	1
2nd order	-	-	-	-	-	-	4.30(-4)	7.68(-4)	9	4.30(-4)	7.68(-4)	1(1)
3rd order	-	-	-	-	-	-	-	-	-	-	-	-
4th order	-	-	-	-	-	-	-	-	-	-	-	-
5th order	-	-	-	-	-	-	-	-	-	-	-	-
$\gamma = 10$ (marginal utility decision rule with $\mu=0.5$)												
1st order	5.47(-4)	1.43(-3)	2(2)	5.47(-4)	1.43(-3)	2(2)	1.62(-3)	7.73(-3)	4(2)	1.62(-3)	7.73(-3)	4(2)
2nd order	4.43(-4)	7.89(-4)	1(2)	4.41(-4)	7.92(-4)	1(2)	8.05(-4)	2.95(-3)	2(2)	8.05(-4)	2.95(-3)	2(2)
3rd order	5.69(-4)	2.71(-3)	4(2)	6.34(-4)	2.86(-3)	1(2)	5.84(-4)	3.13(-3)	2(2)	5.84(-4)	3.13(-3)	2(2)
4th order	7.23(-4)	8.04(-3)	9(2)	7.33(-4)	9.04(-3)	1(2)	8.07(-4)	8.97(-3)	1(2)	8.07(-4)	8.97(-3)	1(2)
5th order	7.72(-4)	1.46(-2)	9(1)	8.97(-4)	1.30(-2)	2(2)	8.48(-4)	8.29(-3)	9(1)	8.48(-4)	8.29(-3)	9(1)

Remark: e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively, and *CPU* is computational time in seconds. The notation $x(m)$ means $x \cdot 10^m$.