Quantitative Economics 2 (2011), 173–210 1759-7331/20110173

Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models

Kenneth L. Judd
Hoover Institution, Stanford University and NBER

Lilia Maliar
University of Alicante and Hoover Institution, Stanford University

Serguei Maliar
University of Alicante and Hoover Institution, Stanford University

We develop numerically stable and accurate stochastic simulation approaches for solving dynamic economic models. First, instead of standard least-squares approximation methods, we examine a variety of alternatives, including least-squares methods using singular value decomposition and Tikhonov regularization, least-absolute deviations methods, and principal component regression method, all of which are numerically stable and can handle ill-conditioned problems. Second, instead of conventional Monte Carlo integration, we use accurate quadrature and monomial integration. We test our generalized stochastic simulation algorithm (GSSA) in three applications: the standard representative–agent neoclassical growth model, a model with rare disasters, and a multicountry model with hundreds of state variables. GSSA is simple to program, and MATLAB codes are provided.

Keywords. Stochastic simulation, generalized stochastic simulation algorithm, parameterized expectations algorithm, least absolute deviations, linear programming, regularization.

JEL classification. C63, C68.

1. Introduction

Dynamic stochastic economic models do not generally admit closed-form solutions and must be studied with numerical methods.[1] Most methods for solving such models fall

Kenneth L. Judd: kennethjudd@mac.com
Lilia Maliar: maliarl@stanford.edu
Serguei Maliar: maliars@stanford.edu
We thank a co-editor and two anonymous referees for very useful comments that led to a substantial improvement of the paper. Lilia Maliar and Serguei Maliar acknowledge support from the Hoover Institution at Stanford University, the Ivie, the Generalitat Valenciana under Grants BEST/2010/142 and BEST/2010/141, respectively, the Ministerio de Ciencia e Innovación de España, and FEDER funds under project SEJ-2007-62656 and under the programs José Castillejo JC2008-224 and Salvador Madariaga PR2008-190, respectively.

[1]For reviews of such methods, see Taylor and Uhlig (1990), Rust (1996), Gaspar and Judd (1997), Judd (1998), Marimon and Scott (1999), Santos (1999), Christiano and Fisher (2000), Miranda and Fackler (2002),

DOI: 10.3982/QE14

into three broad classes: projection methods, which approximate solutions on a prespecified domain using deterministic integration; perturbation methods, which find solutions locally using Taylor expansions of optimality conditions; and stochastic simulation methods, which compute solutions on a set of simulated points using Monte Carlo integration. All three classes of methods have their relative advantages and drawbacks, and the optimal choice of a method depends on the application. Projection methods are accurate and fast when applied to models with few state variables; however, their cost increases rapidly as the number of state variables increases. Perturbation methods are practical to use in high-dimensional applications, but the range of their accuracy is uncertain.[2] Stochastic simulation algorithms are simple to program although they are generally less accurate than projection methods and often numerically unstable.[3] In the present paper, we focus on the stochastic simulation class.[4] We specifically develop a generalized stochastic simulation algorithm (GSSA) that combines advantages of all three classes, namely, it is accurate, numerically stable, tractable in high-dimensional applications, and simple to program.

The key message of the present paper is that a stochastic simulation approach is attractive for solving economic models because it computes solutions only in the part of the state space which is visited in equilibrium—the ergodic set. In Figure 1, we plot the ergodic set of capital and productivity level for a representative–agent growth model with a closed-form solution (for a detailed description of this model, see Section 2.1).

The ergodic set 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 an even smaller area. Thus, the ergodic set is at least 21% smaller than the square. In general, the ratio of the volume of a d-dimensional hypersphere of diameter 1 to the volume of a d-dimensional hypercube of width 1 is

$$\mathcal{V}^d = \begin{cases} \dfrac{(\pi/2)^{(d-1)/2}}{1\cdot 3\cdots d} & \text{for } d = 1, 3, 5, \dots, \\ \dfrac{(\pi/2)^{d/2}}{2\cdot 4\cdots d} & \text{for } d = 2, 4, 6, \dots. \end{cases} \tag{1}$$

The ratio $\mathcal{V}^d$ declines very rapidly with the dimensionality of the state space. For example, for dimensions 3, 4, 5, 10, and 30, this ratio is 0.52, 0.31, 0.16, $3\cdot 10^{-3}$, and $2\cdot 10^{-14}$, respectively.

Aruoba, Fernandez-Villaverde, and Rubio-Ramírez (2006), Heer and Maussner (2008), Den Haan (2010), and Kollmann, Maliar, Malin, and Pichler (2011).

[2]See Judd and Guu (1993), Gaspar and Judd (1997), and Kollmann et al. (2011) for accuracy assessments of perturbation methods.

[3]See Judd (1992) and Christiano and Fisher (2000) for a discussion.

[4]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 (Marcet (1988)), 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).

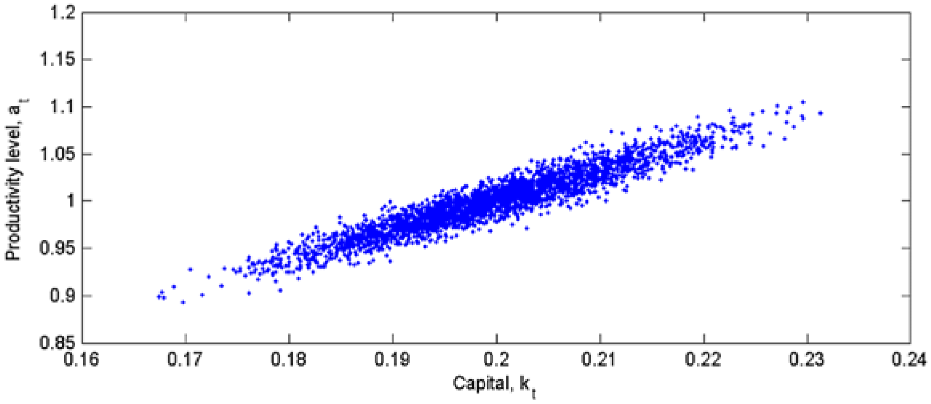

FIGURE 1. The ergodic set in the model with a closed-form solution.

The advantage of focusing on the ergodic set is twofold. First, when computing a solution on an ergodic set that has the shape of a hypersphere, we face just a fraction of the cost we would have faced on a hypercube grid, which is used in conventional projection methods. The higher is the dimensionality of a problem, the larger is the reduction in cost. Second, when fitting a polynomial on the ergodic set, we focus on the relevant domain and can get a better fit inside the relevant domain than conventional projection methods, which face a trade-off between the fit inside and outside the relevant domain.[5]

However, to fully benefit from the advantages of a stochastic simulation approach, we must first stabilize the stochastic simulation procedure. The main reason for the numerical instability of this procedure is that polynomial terms constructed on simulated series are highly correlated with one another even under low-degree polynomial approximations. Under the usual least-squares methods, the multicollinearity problem leads to a failure of the approximation (regression) step.

To achieve numerical stability, we build GSSA on approximation methods that are designed to handle ill-conditioned problems. In the context of a linear regression model, we examine a variety of such methods including least-squares (LS) methods using singular value decomposition (SVD) and Tikhonov regularization, the principal component regression method, and least-absolute deviations (LAD) linear-programming methods (in particular, we present primal and dual LAD regularization methods). In addition, we explore how the numerical stability is affected by other factors such as a normalization of variables, the choice of policy function to parameterize (capital versus marginal-utility policy functions), and the choice of basis functions (ordinary versus Hermite polynomials). Our stabilization strategies are remarkably successful: our approximation methods deliver polynomial approximations up to degree 5 (at least), while the ordinary

[5]The importance of this effect can be seen from the results of the January 2010 special *Journal of Economic Dynamics and Control* issue on numerical methods for solving Krusell and Smith's (1998) model. An Euler equation method based on the Krusell–Smith type of simulation by Maliar, Maliar, and Valli (2010) delivered a more accurate aggregate law of motion than does any other method participating in the comparison analysis, including projection methods; see Table 15 in Den Haan (2010).

least-squares method fails to go beyond the second-degree polynomial in the studied examples.

We next focus on accuracy. We show that if Monte Carlo integration is used to approximate conditional expectations, the accuracy of solutions is dominated by sampling errors from a finite simulation. The sampling errors decrease with the simulation length but the rate of convergence is low, and high-accuracy levels are impractical. For example, in a representative–agent model, Monte Carlo integration leads to accuracy levels (measured by the size of unit-free Euler equation errors on a stochastic simulation) of order 10^{-4}–10^{-5} under the simulation length of 10,000. The highest accuracy is attained under second- or third-degree polynomials. Thus, even though our stabilization strategies enable us to compute a high-degree polynomial approximation, there is no point in doing so with Monte Carlo integration.

To increase the accuracy of solutions, we replace the Monte Carlo integration method with more accurate deterministic integration methods, namely, the Gauss–Hermite quadrature and monomial methods. Such methods are unrelated to the estimated density function and do not suffer from sampling errors. In the representative–agent case, GSSA based on Gauss–Hermite quadrature integration delivers accuracy levels of order 10^{-9}–10^{-10}, which are comparable to those attained by projection methods. Thus, under accurate deterministic integration, high-degree polynomials do help increase the accuracy of solutions.

Given that GSSA allows for a variety of approximation and integration techniques, we can choose a combination of the techniques that takes into account a trade-off between numerical stability, accuracy, and speed for a given application. Some tendencies from our experiments are as follows. LAD methods are generally more expensive than LS methods; however, they deliver smaller mean absolute errors. In small- and moderate-scale problems, the LS method using SVD is more stable than the method using Tikhonov regularization, although the situation reverses in large-scale problems (SVD becomes costly and numerically unstable). Gauss–Hermite quadrature (product) integration rules are very accurate; however, they are practical only with few exogenous random variables (shocks). Monomial (nonproduct) integration rules deliver comparable accuracy and are feasible with many exogenous random variables. Surprisingly, a quadrature integration method with just one integration node is also sufficiently accurate in our examples; in particular, it is more accurate than a Monte Carlo integration method with thousands of integration nodes.

We advocate versions of GSSA that use deterministic integration methods. Such versions of GSSA construct a solution domain using stochastic simulations but compute integrals using methods that are unrelated to simulations; these preferred versions of GSSA, therefore, lie between pure stochastic simulation and pure projection algorithms. Importantly, GSSA keeps the prominent feature of stochastic simulation methods, namely, their tractability in high-dimensional applications. To illustrate this feature, we solve a version of the neoclassical growth model with N heterogeneous countries (the state space is composed of $2N$ variables). For small-scale economies, $N = 6$, 4, and 2, GSSA computes polynomial approximations up to degrees 3, 4, and 5 with maximum absolute errors of 0.001%, 0.0006%, and 0.0002%, respectively. For medium-scale economies, $N = 8$, 10, and 20, GSSA computes second-degree polynomial approximations with the maximum absolute errors of 0.01%, which is comparable to the highest accuracy levels attained in the related literature; see Kollmann et al. (2011). Finally, for large-scale economies, $N = 100$ and 200, GSSA computes first-degree polynomial approximations with the maximum absolute approximation errors of 0.1%. The running time of GSSA depends on the cost of the integration and approximation methods. Our cheapest setup delivers a second-degree polynomial solution to a 20-country model in about 18 minutes using MATLAB and a standard desktop computer.

We present GSSA in the context of examples in which all variables can be expressed analytically in terms of capital policy function, but GSSA can be applied in far more general contexts. In more complicated models (e.g., with valued leisure), intratemporal choices, such as labor supply, are not analytically related to capital policy functions. One way to proceed under GSSA is to approximate intratemporal-choice policy functions as we do with capital; however, this may reduce accuracy and numerical stability. Maliar, Maliar, and Judd (2011) described two intertemporal-choice approaches—precomputation and iteration-on-allocation—that make it possible to find intratemporal choices both accurately and quickly; these approaches are fully compatible with GSSA. Furthermore, GSSA can be applied for solving models with occasionally binding borrowing constraints; see, for example, Marcet and Lorenzoni (1999), Christiano and Fisher (2000), and Maliar, Maliar, and Valli (2010). Finally, the approximation and integration methods described in the paper can be useful in the context of other solution methods, for example, the simulation-based dynamic programming method of Maliar and Maliar (2005).

GSSA is simple to program, and MATLAB codes are provided.[6] Not only can the codes solve the studied examples, but they can be easily adapted to other problems in which the reader may be interested. In particular, the codes include generic routines that implement numerically stable LS and LAD methods, construct multidimensional polynomials, and perform multidimensional Gauss–Hermite quadrature and monomial integration methods. The codes also contain a test suite for evaluating the accuracy of solutions.

The rest of the paper is organized as follows: In Section 2, we describe GSSA using an example of a representative–agent neoclassical growth model. In Section 3, we discuss the reasons for numerical instability of stochastic simulation methods. In Section 4, we elaborate on strategies for enhancing the numerical stability. In Section 5, we compare Monte Carlo and deterministic integration methods. In Section 6, we present the results of numerical experiments. In Section 7, we conclude. The Appendices are available in a supplementary file on the journal website, http://qeconomics.org/supp/14/supplement.pdf.

2. Generalized stochastic simulation algorithm

We describe GSSA using an example of the standard representative–agent neoclassical stochastic growth model. However, the techniques described in the paper are not specific to this model and can be directly applied to other economic models including those

[6]The codes are available at http://www.stanford.edu/~maliars.

with many state and control variables. In Section 7, we show how to apply GSSA for solving models with rare disasters and models with multiple countries.

2.1 *The model*

The agent solves the intertemporal utility-maximization problem

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

$$\text{s.t.} \quad c_t + k_{t+1} = (1-\delta)k_t + a_t f(k_t), \tag{3}$$

$$\ln a_{t+1} = \rho \ln a_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2), \tag{4}$$

where initial condition (k_0, a_0) is given exogenously. Here, E_t is the expectation operator conditional on information at time t; c_t, k_t, and a_t are, respectively, consumption, capital, and productivity level; $\beta \in (0,1)$ is the discount factor; $\delta \in (0,1]$ is the depreciation rate of capital; $\rho \in (-1,1)$ is the autocorrelation coefficient; and $\sigma \geq 0$ is the standard deviation. The utility and production functions, u and f, respectively, are strictly increasing, continuously differentiable, and concave. The solution to (2)–(4) is represented by stochastic processes $\{c_t, k_{t+1}\}_{t=0,\dots,\infty}$ which are measurable with respect to $\{a_t\}_{t=0,\dots,\infty}$. At each time t, the solution to (2)–(4) satisfies the Euler equation

$$u'(c_t) = E_t\{\beta u'(c_{t+1})[1-\delta + a_{t+1} f'(k_{t+1})]\}, \tag{5}$$

where u' and f' are the first derivatives of the utility and production functions, respectively. In a recursive (Markov) equilibrium, decisions of period t are functions of the current state (k_t, a_t). Our objective is to find policy functions for capital, $k_{t+1} = K(k_t, a_t)$, and consumption, $c_t = C(k_t, a_t)$, that satisfy (3)–(5).

2.2 *The GSSA algorithm*

To solve the model (2)–(4), we approximate the capital policy function $k_{t+1} = K(k_t, a_t)$. We choose some flexible functional form $\Psi(k_t, a_t; b)$ and search for a vector of coefficients b such that

$$K(k_t, a_t) \approx \Psi(k_t, a_t; b) \tag{6}$$

for some set of points (k_t, a_t) in the state space. We rewrite the Euler equation (5) in the equivalent form

$$k_{t+1} = E_t\left\{\beta \frac{u'(c_{t+1})}{u'(c_t)}[1-\delta + a_{t+1} f'(k_{t+1})]k_{t+1}\right\}. \tag{7}$$

The condition (7) holds because $u'(c_t) \neq 0$ and because k_{t+1} is t-measurable.[7] We now have expressed k_{t+1} in two ways: as a choice implied by the policy function $k_{t+1} =$

[7]In a similar way, one can use the Euler equation (5) to express other t-measurable variables, for example, $\ln(k_{t+1})$, c_t, and $u'(c_t)$.

$K(k_t, a_t)$ and as a conditional expectation of a time $t+1$ random variable on the right side of (7). This construction gives us a way to express the capital policy function as a fixed point: substituting $K(k_t, a_t)$ into the right side of (7) and computing the conditional expectation should give us $k_{t+1} = K(k_t, a_t)$ for all (k_t, a_t) in the relevant area of the state space.

GSSA finds a solution by iterating on the fixed-point construction (7) via stochastic simulation. To be specific, we guess a capital policy function (6), simulate a time-series solution, compute conditional expectation in each simulated point, and use simulated data to update the guess along iterations until a fixed point is found. The formal description of GSSA is as follows:

Stage 1

Initialization:

- Choose an initial guess $b^{(1)}$.
- Choose the initial state (k_0, a_0) for simulations.
- Choose a simulation length T, draw a sequence of productivity shocks $\{\epsilon_t\}_{t=1,\dots,T}$, and compute $\{a_t\}_{t=1,\dots,T}$ as defined in (4).

Step 1. At iteration p, use $b^{(p)}$ to simulate the model T periods forward:

$$k_{t+1} = \Psi(k_t, a_t; b^{(p)}),$$
$$c_t = (1-\delta)k_t + a_t f(k_t) - k_{t+1}.$$

Step 2. For $t = 0, \dots, T-1$, define y_t to be an approximation of the conditional expectation in (7) using J integration nodes and weights, $\{\epsilon_{t+1,j}\}_{j=1,\dots,J}$ and $\{\omega_{t,j}\}_{j=1,\dots,J}$, respectively:

$$y_t = \sum_{j=1}^{J}\left\{\omega_{t,j} \cdot \left(\beta \frac{u'(c_{t+1,j})}{u'(c_t)}[1-\delta + a_{t+1,j} f'(k_{t+1})]k_{t+1}\right)\right\}, \tag{8}$$

where $c_{t+1,j}$, the value of c_{t+1} if the innovation in productivity is $\epsilon_{t+1,j}$, is defined for $j = 1, \dots, J$ by

$$a_{t+1,j} \equiv a_t^{\rho} \exp(\epsilon_{t+1,j}),$$
$$k_{t+2,j} \equiv \Psi\big(\Psi(k_t, a_t; b^{(p)}), a_t^{\rho} \exp(\epsilon_{t+1,j}); b^{(p)}\big),$$
$$c_{t+1,j} \equiv (1-\delta)k_{t+1} + a_{t+1,j} f(k_{t+1}) - k_{t+2,j}.$$

Step 3. Find $\widehat{b}$ that minimizes the errors ε_t in the regression equation

$$y_t = \Psi(k_t, a_t; b) + \varepsilon_t \tag{9}$$

according to some norm $\|\cdot\|$.

Step 4. Check for convergence and end Stage 1 if

$$\frac{1}{T}\sum_{t=1}^{T}\left|\frac{k_{t+1}^{(p)} - k_{t+1}^{(p-1)}}{k_{t+1}^{(p)}}\right| < \varpi, \tag{10}$$

where $\{k_{t+1}\}_{t=1,\dots,T}$ and $\{k_{t+1}^{(p-1)}\}_{t=1,\dots,T}$ are the capital series obtained on iterations p and $p-1$, respectively.

Step 5. Compute $b^{(p+1)}$ for iteration $(p+1)$ using fixed-point iteration

$$b^{(p+1)} = (1-\xi)b^{(p)} + \xi\widehat{b}, \tag{11}$$

where $\xi \in (0, 1]$ is a damping parameter. Go to Step 1.

Stage 2 The purpose of Stage 2 is to subject the candidate solution from Stage 1 to an independent and stringent test. Construct a new set of T^{test} points $\{k_\tau, a_\tau\}_{\tau=0,\dots,T^{\text{test}}}$ for testing the accuracy of the solution obtained in Stage 1 (this can be a set of simulation points constructed with a new random draw or some deterministic set of points). Rewrite the Euler equation (5) at (k_τ, a_τ) in a unit-free form:

$$\mathcal{E}(k_\tau, a_\tau) \equiv E_\tau\left\{\beta\frac{u'(c_{\tau+1})}{u'(c_\tau)}[1-\delta+a_{\tau+1}f'(k_{\tau+1})]\right\} - 1. \tag{12}$$

For each point (k_τ, a_τ), compute $\mathcal{E}(k_\tau, a_\tau)$ by using a high-quality integration method to evaluate the conditional expectation in (12). We measure the quality of a candidate solution by computing various norms, such as the mean, variance, and/or supremum, of the errors (12). If the economic significance of these errors is small, we accept the candidate b. Otherwise, we tighten up Stage 1 by using a more flexible approximating function, and/or increasing the simulation length, and/or improving the method used for computing conditional expectations, and/or choosing a more demanding norm when computing $\widehat{b}$ in Step 3.[8]

2.3 *Discussion*

GSSA relies on generalized notions of integration and approximation. First, in Step 2, the formula (8) represents both Monte Carlo integration methods and deterministic integration methods such as the Gauss–Hermite quadrature and monomial methods. The choice of integration method is critical for the accuracy of GSSA and is analyzed in Section 5. Second, explanatory variables in the regression equation (9) are often highly collinear, which presents challenges to approximation methods. GSSA uses methods that are suitable for dealing with collinear data, namely, the least-squares methods using SVD and Tikhonov regularization, least-absolute deviations methods, and the principal component regression method. The choice of approximation method is critical for numerical stability of GSSA and is studied in Section 4.

GSSA is compatible with any functional form for Ψ that is suitable for approximating policy functions. In this paper, we examine Ψ of the form

$$\Psi(k_t, a_t; b) = \sum_{i=0}^{n} b_i\psi_i(k_t, a_t) \tag{13}$$

[8]For the models considered in the paper, errors in the Euler equation are the only source of approximation errors. In general, we need to check approximation errors in all optimality conditions, the solutions to which are evaluated numerically.

for a set of basis functions $\{\psi_i \mid i = 0, \dots, n\}$, where $b \equiv (b_0, b_1, \dots, b_n)^\top \in \mathbb{R}^{n+1}$. In Appendix A, we examine cases where the coefficients b enter Ψ in a nonlinear manner and we describe nonlinear approximation methods suitable for dealing with collinear data. The specification (13) implies that in Step 3, the regression equation is linear,

$$y = Xb + \varepsilon, \tag{14}$$

where $y \equiv (y_0, y_1, \dots, y_{T-1})^\top \in \mathbb{R}^T$; $X \equiv [1_T, x_1, \dots, x_n] \in \mathbb{R}^{T\times(n+1)}$ with 1_T being a $T \times 1$ vector whose entries are equal to 1 and $x_{ti} = \psi_i(k_t, a_t)$ for $i = 1, \dots, n$; and $\varepsilon \equiv (\varepsilon_0, \varepsilon_1, \dots, \varepsilon_{T-1})^\top \in \mathbb{R}^T$. (Note that 1_T in X means that $\psi_0(k_t, a_t) = 1$ for all t.) The choice of a family of basis functions used to construct X can affect numerical stability of GSSA. In Section 4.5.1, we consider families of ordinary and Hermite polynomials.[9]

The fixed-point iteration method in Step 4 is a simple derivative-free method for finding a fixed point and is commonly used in the related literature. The advantage of this method is that its cost does not considerably increase with the dimensionality of the problem. The shortcoming is that its convergence is not guaranteed. One typically needs to set the damping parameter ξ in (11) at a value much less than 1 to attain convergence (this, however, slows down the speed of convergence). We were always able to find a value for ξ that gave us convergence.[10]

Finally, our convergence criterion (10) looks at the difference between the time series from two iterations. We do not focus on changes in b since we are interested in the function $K(k_t, a_t)$ and not in its representation in some basis. The regression coefficients b have no economic meaning. The criterion (10) focuses on the economic differences implied by different vectors b.

2.4 *Relation to the literature*

GSSA builds on the past literature on solving rational expectation models but uses a different combination of familiar tools. GSSA differs from conventional deterministic-grid methods in the choice of a solution domain: we solve the model on a relatively small ergodic set instead of some, generally much larger, prespecified domains used, for example, in parameterized expectations approaches (PEA) of Wright and Williams (1984) and Miranda and Helmberger (1988) and projection algorithms of Judd (1992),

[9]GSSA can also use nonpolynomial families of functions. Examples of nonpolynomial basis functions are trigonometric functions, step functions, neural networks, and combinations of polynomials with functions from other families.

[10]Other iterative schemes for finding fixed-point coefficients are time iteration and quasi-Newton methods; see Judd (1998, pp. 553–558 and 103–119, respectively). Time iteration can be more stable than fixed-point iteration; however, it requires solving costly nonlinear equations to find future values of variables. Quasi-Newton methods can be faster and can help achieve convergence if fixed-point iteration does not converge. A stable version of a quasi-Newton method for a stochastic simulation approach requires a good initial condition and the use of line-search methods. Since derivatives are evaluated via simulation, an explosive or implosive simulated series can make a Jacobian matrix ill-conditioned and lead to nonconvergence; we had this problem in some of our experiments.

Christiano and Fisher (2000), and Krueger and Kubler (2004).[11] An ergodic-set domain makes GSSA tractable in high-dimensional applications; see condition (1).[12]

To construct the ergodic set realized in equilibrium, GSSA uses stochastic simulation. This approach is taken in Marcet's (1988) simulation-based version of PEA used in, for example, Den Haan and Marcet (1990), Marcet and Lorenzoni (1999), and Maliar and Maliar (2003a). We differ from this literature in the following respects: We incorporate accurate deterministic integration methods, while the above literature uses a Monte Carlo integration method, whose accuracy is limited. Furthermore, we rely on a variety of numerically stable approximation methods, while the simulation-based version of PEA relies on standard least-squares methods, which are numerically unstable in the given context.[13] In addition, GSSA differs from the literature in the use of a linear regression model that can be estimated with simple and reliable approximation methods.[14] Unlike previous simulation-based methods, GSSA delivers high-degree polynomial approximations and attains accuracy comparable to the best accuracy attained in the literature.

3. Ill-conditioned LS problems

In this section, we discuss the stability issues that arise when standard least-squares (LS) methods are used in the regression equation (14). The LS approach to the regression equation (14) solves the problem

$$\min_b \|y - Xb\|_2^2 = \min_b [y - Xb]^\top [y - Xb], \tag{15}$$

where $\|\cdot\|_2$ denotes the L_2 vector norm. The solution to (15) is

$$\widehat{b} = (X^\top X)^{-1} X^\top y. \tag{16}$$

The LS problem (15) is often ill-conditioned when X is generated by stochastic simulation. The degree of ill-conditioning is measured by the condition number of the matrix $X^\top X$, denoted by $\mathcal{K}(X^\top X)$. Let us order the eigenvalues λ_i, $i = 1, \ldots, n$, of $X^\top X$ by their

[11]Krueger and Kubler's (2004) method relies on a nonproduct Smolyak grid constructed in a multidimensional hypercube. This construction reduces the number of grid points inside the hypercube domain but not the size of the domain itself. Other methods using prespecified nonproduct grids are Malin, Krueger, and Kubler (2011) and Pichler (2011).

[12]Judd, Maliar, and Maliar (2010) and Maliar, Maliar, and Judd (2011) developed a projection method that operates on the ergodic set. The grid surrounding the ergodic set is constructed using clustering methods.

[13]Concerning the simulation-based PEA, Den Haan and Marcet (1990) reported that, even for a low (second-degree) polynomial, cross terms are highly correlated with the other terms and must be removed from the regression. See Judd (1992) and Christiano and Fisher (2000) for a discussion.

[14]The simulation-based PEA literature employs exponentiated polynomial specification $\Psi(k_t, a_t; b) = \exp(b_0 + b_1 \ln k_t + b_2 \ln a_t + \cdots)$. The resulting nonlinear regression model is estimated with nonlinear least-squares (NLLS) methods. The use of NLLS methods is an additional source of numerical problems, because such methods typically need a good initial guess, may deliver multiple minima, and on many occasions fail to converge. Moreover, nonlinear optimization is costly because it requires computing Jacobian and Hessian matrices; see Christiano and Fisher (2000) for a discussion.

magnitude $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. The condition number of $X^\top X$ is equal to the ratio of its largest eigenvalue, λ_1, to its smallest eigenvalue, λ_n, that is, $\mathcal{K}(X^\top X) \equiv \lambda_1/\lambda_n$. The eigenvalues of $X^\top X$ are defined by the standard eigenvalue decomposition $X^\top X = V\Lambda V^\top$, where $\Lambda \in \mathbb{R}^{n\times n}$ is a diagonal matrix of eigenvalues of $X^\top X$ and $V \in \mathbb{R}^{n\times n}$ is an orthogonal matrix of eigenvectors of $X^\top X$. A large condition number implies that $X^\top X$ is close to being singular and not invertible, and tells us that any linear operation, such as (16), is very sensitive to perturbation and numerical errors (such as round-off errors).

Two causes of ill-conditioning are multicollinearity and poor scaling of the variables that constitute X. Multicollinearity occurs when the variables that form X are significantly correlated. The following example illustrates the effects of multicollinearity on the LS solution (we analyze the sensitivity to changes in y, but the results are similar for the sensitivity to changes in X).

Example 1. Let $X = \begin{bmatrix} 1+\phi & 1 \\ 1 & 1+\phi \end{bmatrix}$ with $\phi \neq 0$. Then $\mathcal{K}(X^\top X) = (1+\frac{2}{\phi})^2$. Let $y = (0,0)^\top$. Thus, the ordinary least-squares (OLS) solution (16) is $(\widehat{b}_1, \widehat{b}_2) = (0,0)$. Suppose y is perturbed by a small amount, that is, $y = (\varepsilon_1, \varepsilon_2)^\top$. Then the OLS solution is

$$\widehat{b}_1 = \frac{1}{\phi}\left[\frac{\varepsilon_1(1+\phi) - \varepsilon_2}{2+\phi}\right] \quad \text{and} \quad \widehat{b}_2 = \frac{1}{\phi}\left[\frac{\varepsilon_2(1+\phi) - \varepsilon_1}{2+\phi}\right]. \tag{17}$$

Sensitivity of $\widehat{b}_1$ and $\widehat{b}_2$ to perturbation in y is proportional to $1/\phi$ (increases with $\mathcal{K}(X^\top X)$).

The scaling problem arises when the columns (the variables) of X have significantly different means and variances (due to differential scaling among either the state variables, k_t and a_t, or their functions, for example, k_t and k_t^5). A column with only very small entries is treated as if it is a column of zeros. The next example illustrates the effect of the scaling problem.

Example 2. Let $X = \begin{bmatrix} 1 & 0 \\ 0 & \phi \end{bmatrix}$ with $\phi \neq 0$. Then $\mathcal{K}(X^\top X) = 1/\phi$. Let $y = (0,0)^\top$. Thus, the OLS solution (16) is $(\widehat{b}_1, \widehat{b}_2) = (0,0)$. Suppose y is perturbed by a small amount, that is, $y = (\varepsilon_1, \varepsilon_2)^\top$. The OLS solution is

$$\widehat{b}_1 = \varepsilon_1 \quad \text{and} \quad \widehat{b}_2 = \frac{\varepsilon_2}{\phi}. \tag{18}$$

Sensitivity of $\widehat{b}_2$ to perturbation in y is proportional to $1/\phi$ (and $\mathcal{K}(X^\top X)$).

A comparison of Examples 1 and 2 shows that multicollinearity and poor scaling magnify the impact of perturbations on the OLS solution. Each iteration of a stochastic simulation algorithm produces changes in simulated data (perturbations). In the presence of ill-conditioning, these changes together with numerical errors may induce large and erratic jumps in the regression coefficients and failures to converge.

4. Enhancing numerical stability

We need to make choices of approximation methods that ensure numerical stability of GSSA. We face two challenges: first, we must solve the approximation step for any given set of simulation data; second, we must attain the convergence of the iterations over b. The stability of the iterations over b depends on the sensitivity of the regression coefficients to the data (each iteration of GSSA produces different time series and results in large changes in successive values of b and nonconvergence). In this section, we present approximation methods that can handle collinear data, namely, a LS method using a singular value decomposition (SVD) and least-absolute deviations (LAD) method. Furthermore, we describe regularization methods that not only can deal with ill-conditioned data, but can also dampen movements in b by effectively penalizing large values of the regression coefficients. Such methods are a LS method using Tikhonov regularization, LAD regularization methods, and the principal component regression method. We finally analyze other factors that can affect numerical stability of GSSA, namely, data normalization, the choice of a family of basis functions, and the choice of policy functions to parameterize.

4.1 *Normalizing the variables*

Data normalization addresses the scaling issues highlighted in Example 2. Also, our regularization methods require the use of normalized data. We center and scale both the response variable y and the explanatory variables of X to have a zero mean and unit standard deviation. We then estimate a regression model without an intercept to obtain the vector of coefficients $(\widehat{b}_1^+, \dots, \widehat{b}_n^+)$. We finally restore the coefficients $\widehat{b}_1, \dots, \widehat{b}_n$ and the intercept $\widehat{b}_0$ in the original (unnormalized) regression model according to $\widehat{b}_i = (\sigma_y/\sigma_{x_i})\widehat{b}_i^+$, $i = 1, \dots, n$, and $\widehat{b}_0 = \overline{y} - \sum_{i=1}^n \widehat{b}_i^+ \overline{x}_i$, 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.[15]

4.2 *LS approaches*

In this section, we present two LS approaches that are more numerically stable than the standard OLS approach. The first approach, called *LS using SVD* (LS-SVD), uses a singular value decomposition (SVD) of X. The second approach, called *regularized LS using Tikhonov regularization* (RLS-Tikhonov), 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 less ill-conditioned problem.

[15]To maintain a simple system of notation, we do not introduce separate notation for normalized and unnormalized variables. Instead, we remember that when the regression model is estimated with normalized variables, we have $b \in \mathbb{R}^n$, and when it is estimated with unnormalized variables, we have $b \in \mathbb{R}^{n+1}$.

4.2.1 *LS-SVD* We can use the SVD of X to rewrite the OLS solution (16) in a way that does not require an explicit computation of $(X^\top X)^{-1}$. For a matrix $X \in \mathbb{R}^{T\times n}$ with $T > n$, an SVD decomposition is

$$X = USV^\top, \tag{19}$$

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 $s_1 \geq s_2 \geq \cdots \geq s_n \geq 0$, known as *singular values* of X.[16] The condition number of X is its largest singular value divided by its smallest singular value, $\mathcal{K}(X) = s_1/s_n$. The singular values of X are related to the eigenvalues of $X^\top X$ by $s_i = \sqrt{\lambda_i}$; see, for example, Golub and Van Loan (1996, p. 448). This implies that $\mathcal{K}(X) = \mathcal{K}(S) = \sqrt{\mathcal{K}(X^\top X)}$. The OLS estimator $\widehat{b} = (X^\top X)^{-1}X^\top y$ in terms of the SVD (19) is

$$\widehat{b} = VS^{-1}U^\top y. \tag{20}$$

With an infinite-precision computer, the OLS formula (16) and the LS-SVD formula (20) give identical estimates of b. With a finite-precision computer, the standard OLS estimator cannot be computed reliably if $X^\top X$ is ill-conditioned. However, it is still possible that X and S are sufficiently well-conditioned so that the LS-SVD estimator can be computed successfully.[17]

4.2.2 *RLS-Tikhonov* A regularization method replaces an ill-conditioned problem with a well-conditioned problem that gives a similar answer. Tikhonov regularization is commonly used to solve ill-conditioned problems. In statistics, this method is known as ridge regression and it is classified as a shrinkage method because it shrinks the norm of the estimated coefficient vector relative to the nonregularized solution. Formally, Tikhonov regularization imposes an L_2 penalty on the magnitude of the regression-coefficient vector; that is, for a regularization parameter $\eta \geq 0$, the vector $b(\eta)$ solves

$$\min_b \|y - Xb\|_2^2 + \eta\|b\|_2^2 = \min_b (y - Xb)^\top(y - Xb) + \eta b^\top b, \tag{21}$$

where $y \in \mathbb{R}^T$ and $X \in \mathbb{R}^{T\times n}$ are centered and scaled, and $b \in \mathbb{R}^n$. The parameter η controls the amount by which the regression coefficients shrink; larger values of η lead to greater shrinkage.

Note that the scale of an explanatory variable affects the size of the regression coefficient on this variable and hence, it affects how much this coefficient is penalized. Normalizing all explanatory variables x_i to zero mean and unit standard deviation allows us to use the same penalty η for all coefficients. Furthermore, centering the response variable y leads to a no-intercept regression model and, thus, allows us to impose a penalty

[16]For a description of methods for computing the SVD of a matrix, see, for example, Golub and Van Loan (1996, pp. 448–460). Routines that compute the SVD are readily available in modern programming languages.

[17]Another decomposition of X that leads to a numerically stable LS approach is a QR factorization; see, for example, Davidson and MacKinnon (1993, pp. 30–31) and Golub and Van Loan (1996, p. 239).

on the coefficients $b_1, \dots, b_n$ without distorting the intercept b_0 (the latter is recovered after all other coefficients are computed; see Section 4.1).

Finding the first-order condition of (21) with respect to b gives us the estimator

$$\widehat{b}(\eta) = (X^\top X + \eta I_n)^{-1} X^\top y, \tag{22}$$

where I_n is an identity matrix of order n. Note that Tikhonov regularization adds a positive constant multiple of the identity matrix to $X^\top X$ prior to inversion. Thus, if $X^\top X$ is nearly singular, the matrix $X^\top X + \eta I_n$ is less singular, reducing problems in computing $\widehat{b}(\eta)$. Note that $\widehat{b}(\eta)$ is a biased estimator of b. As η increases, the bias of $\widehat{b}(\eta)$ increases and its variance decreases. Hoerl and Kennard (1970) showed that there exists a value of η such that

$$E\big[(\widehat{b}(\eta) - b)^\top (\widehat{b}(\eta) - b)\big] < E[(\widehat{b} - b)^\top (\widehat{b} - b)],$$

that is, the mean squared error (equal to the sum of the variance and the squared bias) of the Tikhonov-regularization estimator, $\widehat{b}(\eta)$, is smaller than that of the OLS estimator, $\widehat{b}$. Two main approaches to finding an appropriate value of the regularization parameter in statistics are ridge trace and cross-validation. The ridge-trace approach relies on a stability criterion: we observe a plot showing how $\widehat{b}(\eta)$ changes with η (ridge trace) and select the smallest value of η for which $\widehat{b}(\eta)$ is stable. The cross-validation approach focuses on a statistical-fit criterion. We split the data into two parts, fix some η, compute an estimate $\widehat{b}(\eta)$ using one part of the data, and evaluate the fit of the regression (i.e., validate the regression model) using the other part of the data. We then iterate on η to maximize the fit. For a detailed discussion of the ridge-trace and cross-validation approaches used in statistics, see, for example, Brown (1993, pp. 62–71).

The problem of finding an appropriate value of η for GSSA differs from that in statistics in two respects: First, in Stage 1, our data are not fixed and not exogenous to the regularization process: on each iteration, simulated series are recomputed using a policy function that was obtained in the previous iteration under some value of the regularization parameter. Second, our criteria of stability and accuracy differ from those in statistics. Namely, our criterion of stability is the convergence of the fixed-point iteration in Stage 1, and our criterion of fit is the accuracy of the converged solution measured by the size of the Euler equation errors in Stage 2. In Section 6.1, we discuss how we chose the regularization parameter for the RLS-Tikhonov method (as well as for other regularization methods presented below) in the context of GSSA.

4.3 *LAD approaches*

LAD, or L_1, regression methods use linear programming to minimize the sum of absolute deviations. LAD methods do not depend on $(X^\top X)^{-1}$ and avoid the ill-conditioning problems of LS methods. Section 4.3.1 develops primal and dual formulations of the LAD problem, and Section 4.3.2 proposes regularized versions of both. Section 4.3.3 discusses the advantages and drawbacks of the LAD approaches.

4.3.1 *LAD* The basic LAD method solves the optimization problem

$$\min_b \|y - Xb\|_1 = \min_b 1_T^\top |y - Xb|, \tag{23}$$

where $\|\cdot\|_1$ denotes the L_1 vector norm and $|\cdot|$ denotes the absolute value.[18] Without a loss of generality, we assume that X and y are centered and scaled.

There is no explicit solution to the LAD problem (23), but this problem is equivalent to the linear-programming problem

$$\min_{g,b} 1_T^\top g \tag{24}$$

$$\text{s.t.} \quad -g \le y - Xb \le g, \tag{25}$$

where $g \in \mathbb{R}^T$. The problem has $n + T$ unknowns. Although this formulation of the LAD problem is intuitive, it is not the most suitable for a numerical analysis.

LAD: Primal problem (LAD-PP). Charnes, Cooper, and Ferguson (1955) showed that a linear LAD problem can be transformed into a canonical linear programming form. They expressed the deviation for each observation as a difference between two nonnegative variables υ_t^+ and υ_t^-, as in

$$y_t - \sum_{i=0}^{n} b_i x_{ti} = \upsilon_t^+ - \upsilon_t^-, \tag{26}$$

where x_{ti} is the tth element of the vector x_i. The variables υ_t^+ and υ_t^- represent the magnitude of the deviations above and below the fitted line $\widehat{y}_t = X_t\widehat{b}$, respectively. The sum $\upsilon_t^+ + \upsilon_t^-$ 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 (26). In vector notation, this problem is

$$\min_{\upsilon^+,\upsilon^-,b} 1_T^\top \upsilon^+ + 1_T^\top \upsilon^- \tag{27}$$

$$\text{s.t.} \quad \upsilon^+ - \upsilon^- + Xb = y, \tag{28}$$

$$\upsilon^+ \ge 0, \qquad \upsilon^- \ge 0, \tag{29}$$

where $\upsilon^+, \upsilon^- \in \mathbb{R}^T$. This is called the *primal problem*. A noteworthy property of its solution is that both υ_t^+ and υ_t^- cannot be strictly positive at a solution; if so, we could reduce both υ_t^+ and υ_t^- by the same quantity and reduce the value of the objective function without affecting the constraint (28). The advantage of (27)–(29) compared to (24) and (25) is that the only inequality constraints in the former problem are the variable bounds (29), a feature that often helps make linear programming algorithms more efficient.

[18]LAD regression is a particular case of quantile regressions introduced by Koenker and Bassett (1978). The central idea behind quantile regressions is the assignation of differing weights to positive versus negative residuals, $y - Xb$. A ςth regression quantile, $\varsigma \in (0, 1)$, is defined as a solution to the problem of minimizing a weighted sum of residuals, where ς is a weight on positive residuals. The LAD estimator is the regression median, that is, the regression quantile for $\varsigma = 1/2$.

LAD: Dual problem (LAD-DP). Linear programming tells us that every primal problem can be converted into a dual problem.[19] The dual problem corresponding to (27)–(29) is

$$\max_q y^\top q \tag{30}$$

$$\text{s.t.} \quad X^\top q = 0, \tag{31}$$

$$-1_T \leq q \leq 1_T, \tag{32}$$

where $q \in \mathbb{R}^T$ is a vector of unknowns. Wagner (1959) argued that if the number of observations T is sizable (i.e., $T \gg n$), the dual problem (30)–(32) is computationally less cumbersome than the primal problem (27)–(29). 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. The elements of the vector b, which is what we want to compute, are equal to the Lagrange multipliers associated with the equality restrictions given in (31).

4.3.2 *Regularized LAD (RLAD)* We next modify the original LAD problem (23) to incorporate an L_1 penalty on the coefficient vector b. We refer to the resulting problem as a *regularized LAD* (RLAD). Like Tikhonov regularization, our RLAD problem shrinks the values of the coefficients toward zero. Introducing an L_1 penalty in place of the L_2 penalty from Tikhonov regularization allows us to have the benefits of biasing coefficients to zero but to do so with linear programming. Formally, for a given regularization parameter $\eta \geq 0$, the RLAD problem attempts to find the vector $b(\eta)$ that solves

$$\min_b \|y - Xb\|_1 + \eta \|b\|_1 = \min_b 1_T^\top |y - Xb| + \eta 1_n^\top |b|, \tag{33}$$

where $y \in \mathbb{R}^T$ and $X \in \mathbb{R}^{T\times n}$ are centered and scaled, and $b \in \mathbb{R}^n$. As in the case of Tikhonov regularization, centering and scaling of X and y in the RLAD problem (33) allows us to use the same penalty parameter for all explanatory variables and to avoid penalizing an intercept. Below, we develop a linear programming formulation of the RLAD problem in which an absolute value term $|b_i|$ is replaced with a difference between two nonnegative variables. Our approach is parallel to the one we used to construct the primal problem (27)–(29) and differs from the approach used in statistics.[20]

RLAD: Primal problem (RLAD-PP). To cast the RLAD problem (33) into a canonical linear programming form, we represent the coefficients of the vector b as $b_i = \varphi_i^+ - \varphi_i^-$, with $\varphi_i^+ \geq 0$, $\varphi_i^- \geq 0$ for $i = 1, \ldots, n$. The regularization is done by adding to the objective a penalty linear in each φ_i^+ and φ_i^-. The resulting regularized version of the primal problem (27)–(29) is

$$\min_{\upsilon^+, \upsilon^-, \varphi^+, \varphi^-} 1_T^\top \upsilon^+ + 1_T^\top \upsilon^- + \eta 1_n^\top \varphi^+ + \eta 1_n^\top \varphi^- \tag{34}$$

$$\text{s.t.} \quad \upsilon^+ - \upsilon^- + X\varphi^+ - X\varphi^- = y, \tag{35}$$

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

[20]Wang, Gordon, and Zhu (2006) constructed a RLAD problem in which $|b_i|$ is represented as $\text{sign}(b_i)b_i$.

$$v^+ \geq 0, \qquad v^- \geq 0, \tag{36}$$

$$\varphi^+ \geq 0, \qquad \varphi^- \geq 0, \tag{37}$$

where $\varphi^+, \varphi^- \in \mathbb{R}^n$ are vectors that define $b(\eta)$. The above problem has $2T + 2n$ unknowns, as well as T equality restrictions (35) and $2T + 2n$ lower bounds (36) and (37).

RLAD: Dual problem (RLAD-DP). The dual problem corresponding to the RLAD-PP (34)–(37) is

$$\max_q y^\top q \tag{38}$$

$$\text{s.t.} \quad X^\top q \leq \eta \cdot 1_n, \tag{39}$$

$$-X^\top q \leq \eta \cdot 1_n, \tag{40}$$

$$-1_T \leq q \leq 1_T, \tag{41}$$

where $q \in \mathbb{R}^T$ is a vector of unknowns. Here, $2n$ linear inequality restrictions are imposed by (39) and (40), and $2T$ lower and upper bounds on T unknown components of q are given in (41). By solving the dual problem, we obtain the coefficients of the vectors φ^+ and φ^- as the Lagrange multipliers associated with (39) and (40), respectively; we can then restore the RLAD estimator using $b(\eta) = \varphi^+ - \varphi^-$.

4.3.3 *Advantages and drawbacks of LAD approaches* LAD approaches are more robust to outliers than LS approaches because they minimize errors without squaring them and, thus, place comparatively less weight on distant observations than LS approaches do. LAD approaches have two advantages compared to LS approaches. First, the statistical literature suggests that LAD estimators are preferable if regression disturbances are nonnormal, asymmetric, or heavy-tailed; see Narula and Wellington (1982) and Dielman (2005) for surveys. Second, LAD methods can easily accommodate additional linear restrictions on the regression coefficients, for example, restrictions that impose monotonicity of policy functions. In contrast, adding such constraints for LS methods changes an unconstrained convex minimization problem into a linearly constrained convex minimization problem and substantially increases the computational difficulty.

LAD approaches have two drawbacks compared to the LS approaches. First, a LAD estimator does not depend smoothly on the data; since it corresponds to the median, the minimal sum of absolute deviations is not differentiable in the data. Moreover, a LAD regression line may not even be continuous in the data: a change in the data could cause a solution switch from one vertex of the feasible set of coefficients to another vertex. This jump will cause a discontinuous change in the regression line, which in turn will produce a discontinuous change in the simulated path. These jumps might create problems in solving for a fixed point. Second, LAD approaches require solving linear-programming problems, whereas LS approaches use only linear algebra operations. Therefore, LAD approaches tend to be more costly than LS approaches.

4.4 *Principal component (truncated SVD) method*

In this section, we describe a principal component method that reduces the multicollinearity in the data to a target level. 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 (19). Let us make a linear transformation of X using $Z \equiv XV$, where $Z \in \mathbb{R}^{T\times n}$ and $V \in \mathbb{R}^{n\times n}$ is the matrix of singular vectors of X defined by (19). The vectors $z_1, \dots, z_n$ are called *principal components* of X. They are orthogonal, $z_{i'}^\top z_i = 0$ for any $i' \neq i$, and their norms are related to the singular values s_i by $z_i^\top z_i = s_i^2$. 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 $x_1, \dots, x_n$, each of which has a zero mean; second, the variance of each principal component is equal to s_i^2/T, because we have $z_i^\top z_i = s_i^2$.

Since the SVD method 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, $z_i = (0, \dots, 0)^\top$, which implies that the variables $x_1, \dots, x_n$ that constitute this particular principal component are linearly dependent. Therefore, we can reduce the degrees of ill-conditioning of X to some target level by excluding low-variance principal components that correspond to small singular values.

To formalize the above idea, let κ represent the largest condition number of X that we are willing to tolerate. Let us compute the ratios of the largest singular value to all other singular values, $\frac{s_1}{s_2}, \dots, \frac{s_1}{s_n}$. (Recall that the last ratio is the actual condition number of the matrix X; $\mathcal{K}(X) = \mathcal{K}(S) = \frac{s_1}{s_n}$.) Let $Z^r \equiv (z_1, \dots, z_r) \in \mathbb{R}^{T\times r}$ be the first r principal components for which $\frac{s_1}{s_i} \leq \kappa$ and let us remove the last $n-r$ principal components for which $\frac{s_1}{s_i} > \kappa$. By construction, the matrix Z^r has a condition number which is smaller than or equal to κ.

Let us consider the regression equation (14) and let us approximate Xb using Z^r such that $Xb = XVV^{-1}b \approx XV^r(V^r)^{-1}b(\kappa) = Z^r\vartheta^r$, where $V^r = (v_1, \dots, v_r) \in \mathbb{R}^{n\times r}$ contains the first r singular vectors of X and $\vartheta^r \equiv (V^r)^{-1}b(\kappa) \in \mathbb{R}^r$. The resulting regression equation is

$$y = Z^r\vartheta^r + \varepsilon, \tag{42}$$

where y is centered and scaled. The coefficients ϑ^r can be estimated by any of the methods described in Sections 4.2 and 4.3. For example, we can compute the OLS estimator (16). Once we compute $\widehat{\vartheta}^r$, we can recover the coefficients $\widehat{b}(\kappa) = V^r\widehat{\vartheta}^r \in \mathbb{R}^n$.

We can remove collinear components of the data using a truncated SVD method instead of the principal component method. Let the matrix $X^r \in \mathbb{R}^{T\times n}$ be defined by a truncated SVD of X, such that $X^r \equiv U^rS^r(V^r)^\top$, where $U^r \in \mathbb{R}^{T\times r}$ 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), X^r is the closest rank r approximation of $X \in \mathbb{R}^{T\times n}$. In terms of X^r, the regression equation is $y = X^rb(r) + \varepsilon$. Using the definition of X^r, we can write $X^rb(r) = X^rV^r(V^r)^{-1}b(r) = X^rV^r\vartheta^r = U^rS^r\vartheta^r$, where $\vartheta^r \equiv (V^r)^{-1}b(r) \in \mathbb{R}^r$. Again, we can estimate the resulting regression model $y = U^r S^r \vartheta^r + \varepsilon$ with any of the methods described in Sections 4.2 and 4.3 and recover $\widehat{b}(r) = V^r \widehat{\vartheta}^r \in \mathbb{R}^n$. In particular, we can find $\widehat{\vartheta}^r$ using the OLS method and arrive at

$$\widehat{b}(r) = V^r (S^r)^{-1} (U^r)^\top y. \tag{43}$$

We call the estimator (43) *regularized LS using truncated SVD (RLS-TSVD)*. If $r = n$, then RLS-TSVD coincides with LS-SVD described in Section 4.2.1.[21] The principal component and truncated SVD methods are related through $Z^r = X^r V^r$.

We make two remarks. First, the principal component regression (42) is well suited to the shrinkage type of regularization methods without additional scaling: the lower is 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. Second, we should be careful removing low-variance principal components, since they may contain important pieces of information.[22] To rule out only the case of extremely collinear variables, a safe strategy is to set κ to a very large number, for example, to 10^{14} on a machine with 16 digits of precision.

4.5 *Other factors that affect numerical stability*

We complement our discussion by analyzing two other factors that can affect numerical stability of GSSA: the choice of a family of basis functions and the choice of policy functions to parameterize.

4.5.1 *Choosing a family of basis functions* We restrict attention to polynomial families of basis functions in (13). Let us first consider an ordinary polynomial family $O_m(x) = x^m$, $m = 0, 1, \ldots$. The basis functions of this family look very similar (namely, $O_2(x) = x^2$ looks similar to $O_4(x) = x^4$, and $O_3(x) = x^3$ looks similar to $O_5(x) = x^5$); see Figure 2(a). As a result, the explanatory variables in the regression equation are likely to be strongly correlated (i.e., the LS problem is ill-conditioned) and estimation methods (e.g., OLS) may fail because they cannot distinguish between similarly shaped basis functions.

In contrast, for families of orthogonal polynomials (e.g., Hermite, Chebyshev, Legendre), basis functions have very different shapes and, hence, the multicollinearity problem is likely to manifest to a smaller degree, if at all.[23] In this paper, we consider the case of Hermite polynomials. Such polynomials can be defined with a simple recursive formula: $H_0(x) = 1$, $H_1(x) = x$, and $H_m(x) = xH_m(x) - mH_{m-1}(x)$. For example, for $m = 1, \ldots, 5$, this formula yields $H_0(x) = 1$, $H_1(x) = x$, $H_2(x) = x^2 - 1$, $H_3(x) = x^3 - 3x$,

[21]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 backslash operator for solving linear systems of equations.

[22]Hadi and Ling (1998) constructed 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.

[23]This useful feature of orthogonal polynomials was emphasized by Judd (1992) in the context of projection methods.

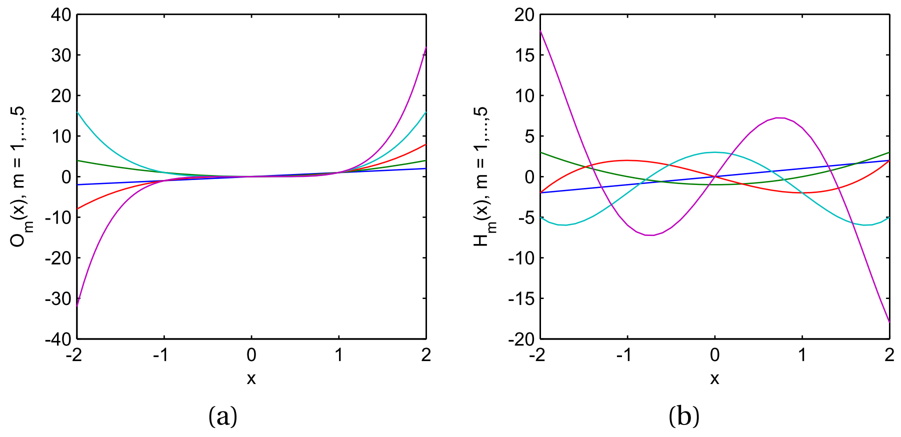

FIGURE 2. (a) Ordinary polynomials. (b) Hermite polynomials.

$H_4(x) = x^4 - 6x^2 + 3$, and $H_5(x) = x^5 - 10x^3 + 15x$. These basis functions look different; see Figure 2(b).

Two points are in order. First, Hermite polynomials are orthogonal under the Gaussian density function, but are not orthogonal under the ergodic measure of our simulations. Still, Hermite polynomials are far less correlated than ordinary polynomials, which may suffice to avoid ill-conditioning. Second, even though using Hermite polynomials helps us avoid ill-conditioning in one variable, it does not help us to deal with multicollinearity across variables. For example, if k_t and a_t happen to be perfectly correlated, certain Hermite polynomial terms for k_t and a_t, like $H_2(k_t) = k_t^2 - 1$ and $H_2(a_t) = a_t^2 - 1$, are also perfectly correlated and, hence, X is singular. Thus, we may still need regression methods that are able to treat ill-conditioned problems.[24]

4.5.2 *Choosing policy functions to approximate* The numerical stability of the approximation step is a necessary but not sufficient condition for the numerical stability of GSSA. It might happen that fixed-point iteration in (11) does not converge along iterations even if the policy function is successfully approximated on each iteration. The fixed-point iteration procedure (even with damping) is sensitive to the nature of nonlinearity of solutions. There exist many logically equivalent ways to parameterize solutions, with some parameterizations working better than others. A slight change in the nonlinearity of solutions due to variations in the model's parameters might shift the balance between different parameterizations; see Judd (1998, p. 557) for an example. Switching to a different policy function to approximate can possibly help stabilize fixed-point iteration. Instead of capital policy function (6), we can approximate the policy function for marginal utility in the left side of the Euler equation (5), $u'(c_t) = \Psi^u(k_t, a_t; b^u)$. This parameterization is common for the literature using Marcet's (1988) simulation-based PEA (although the parameterization of capital policy functions is also used to solve models with multiple Euler equations; see, for example, Den Haan (1990)).

[24]Christiano and Fisher (2000) found that multicollinearity can plague the regression step even with orthogonal (Chebyshev) polynomials as basis functions.

5. Increasing accuracy of integration

In Sections 5.1 and 5.2, we describe the Monte Carlo and deterministic integration methods, respectively. We argue that accuracy of integration plays a determinant role in the accuracy of GSSA solutions.

5.1 *Monte Carlo integration*

A one-node Monte Carlo integration method approximates an integral with the next-period's realization of the integrand; we call it MC(1). Setting $\epsilon_{t+1,1} \equiv \epsilon_{t+1}$ and $\omega_{t,1} = 1$ transforms (8) into

$$y_t = \beta \frac{u'(c_{t+1})}{u'(c_t)}[1 - \delta + a_{t+1} f'(k_{t+1})]k_{t+1}. \tag{44}$$

This integration method is used in Marcet's (1988) simulation-based version of PEA.

A J-node Monte Carlo integration method, denoted by MC(J), draws J shocks, $\{\epsilon_{t+1,j}\}_{j=1,\ldots,J}$ (which are unrelated to ϵ_{t+1}, the shock along the simulated path) and computes y_t in (8) by assigning equal weights to all draws, that is, $\omega_{t,j} = 1/J$ for all j and t.

An integration error is given by $\varepsilon_t^I \equiv y_t - E_t[\cdot]$, where $E_t[\cdot]$ denotes the exact value of conditional expectation in (7).[25] The OLS estimator (16) yields $\widehat{b} = b + [(X)^\top X]^{-1} \times (X)^\top \varepsilon^I$, where $\varepsilon^I \equiv (\varepsilon_1^I, \ldots, \varepsilon_T^I)^\top \in \mathbb{R}^T$. Assuming that ε_t^I is independent and identically distributed (i.i.d.) with zero mean and constant variance σ_ε^2, we have the standard version of the central limit theorem. For the conventional one-node Monte Carlo integration method, MC(1), the asymptotic distribution of the OLS estimator is given by $\sqrt{T}(\widehat{b} - b) \sim \mathcal{N}(0, [X^\top X]^{-1}\sigma_\varepsilon^2)$, and the convergence rate of the OLS estimator is $\sqrt{T}$. Similarly, the convergence rate for MC(J) is $\sqrt{TJ}$. To decrease errors by an order of magnitude, we must increase either the simulation length T or the number of draws J by 2 orders of magnitude or do some combination of the two.

Since the convergence of Monte Carlo integration is slow, high accuracy is theoretically possible but impractical. In a typical real business cycle model, variables fluctuate by several percent and so does the variable y_t given by (44). If a unit-free integration error $|\frac{y_t - E_t[\cdot]}{E_t[\cdot]}|$ is on average 10^{-2} (i.e., 1%), then a regression model with $T = 10{,}000$ observations has errors of order $10^{-2}/\sqrt{T} = 10^{-4}$. To reduce errors to order 10^{-5}, we would need to increase the simulation length to $T = 1{,}000{,}000$. Thus, the cost of accuracy improvements is prohibitive.[26]

5.2 *One-dimensional quadrature integration*

Deterministic integration methods are unrelated to simulations. In our model with one normally distributed exogenous random variable, we can approximate a one-di-

[25]Other types of approximation errors are discussed in Judd, Maliar, and Maliar (2011).

[26]In a working-paper version of the present paper, Judd, Maliar, and Maliar (2009) developed a variant of GSSA based on the one-node Monte Carlo integration method. This variant of GSSA is included in the comparison analysis of Kollmann et al. (2011).

mensional integral using Gauss–Hermite quadrature. A J-node Gauss–Hermite quadrature method, denoted by $Q(J)$, computes y_t in (8) using J deterministic integration nodes and weights. For example, a two-node Gauss–Hermite quadrature method, $Q(2)$, uses nodes $\epsilon_{t+1,1} = -\sigma$, $\epsilon_{t+1,2} = \sigma$ and weights $\omega_{t,1} = \omega_{t,2} = \frac{1}{2}$; a three-node Gauss–Hermite quadrature method, $Q(3)$, uses nodes $\epsilon_{t+1,1} = 0$, $\epsilon_{t+1,2} = \sigma\sqrt{\frac{3}{2}}$, $\epsilon_{t+1,3} = -\sigma\sqrt{\frac{3}{2}}$ and weights $\omega_{t,1} = \frac{2\sqrt{\pi}}{3}$, $\omega_{t,2} = \omega_{t,3} = \frac{\sqrt{\pi}}{6}$. A special case of the Gauss–Hermite quadrature method is a one-node rule, $Q(1)$, which uses a zero node $\epsilon_{t+1,1} = 0$ and a unit weight $\omega_{t,1} = 1$. Integration errors under Gauss–Hermite quadrature integration can be assessed using the Gauss–Hermite quadrature formula, see, for example, Judd (1998, p. 261). For a function that is smooth and has little curvature, the integration error decreases rapidly with the number of integration nodes J. In particular, Gauss–Hermite quadrature integration is exact for functions that are linear in the exogenous random variable.

5.3 *Multidimensional quadrature and monomial integration*

We now discuss deterministic integration methods suitable for models with multiple exogenous random variables (in Section 6.6, we extend our baseline model to include multiple countries hit by idiosyncratic shocks). In this section, we just provide illustrative examples; a detailed description of such methods is given in Appendix B.

With a small number of normally distributed exogenous random variables, we can approximate multidimensional integrals with a Gauss–Hermite product rule, which constructs multidimensional nodes as a tensor product of one-dimensional nodes. Below, we illustrate an extension of the two-node quadrature rule to the multidimensional case by way of example.

Example 3. Let $\epsilon^h_{t+1} \sim \mathcal{N}(0, \sigma^2)$, $h = 1, 2, 3$, be uncorrelated random variables. A two-node Gauss–Hermite product rule $Q(2)$ (obtained from the two-node Gauss–Hermite rule) has 2^3 nodes, which are

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$
$\epsilon^1_{t+1,j}$	σ	σ	σ	σ	$-\sigma$	$-\sigma$	$-\sigma$	$-\sigma$
$\epsilon^2_{t+1,j}$	σ	σ	$-\sigma$	$-\sigma$	σ	σ	$-\sigma$	$-\sigma$
$\epsilon^3_{t+1,j}$	σ	$-\sigma$	σ	$-\sigma$	σ	$-\sigma$	σ	$-\sigma$

where weights of all nodes are equal, $\omega_{t,j} = 1/8$ for all j.

Under a J-node Gauss–Hermite product rule, the number of nodes grows exponentially with the number of exogenous random variables N. Even if there are just two nodes for each random variable, the total number of nodes is prohibitively large for large N; for example, if $N = 100$, we have $2^N \approx 10^{30}$ nodes. This makes product rules impractical.

With a large number of exogenous random variables, a feasible alternative to product rules is monomial rules. Monomial rules construct multidimensional integration nodes directly in a multidimensional space. Typically, the number of nodes under monomial rules grows polynomially with the number of exogenous random variables. In Appendix B, we present a description of two monomial rules, denoted by $M1$ and $M2$, which have $2N$ and $2N^2+1$ nodes, respectively. In particular, $M1$ constructs nodes by considering consecutive deviations of each random variable from its expected value, holding the other random variables fixed to their expected values. We illustrate this construction using the setup of Example 3.

Example 4. Let $\epsilon_{t+1}^{h} \sim \mathcal{N}(0, \sigma^2)$, $h = 1, 2, 3$, be uncorrelated random variables. A monomial nonproduct rule $M1$ has $2 \cdot 3$ nodes, which are

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$\epsilon_{t+1,j}^{1}$	$\sigma\sqrt{3}$	$-\sigma\sqrt{3}$	0	0	0	0
$\epsilon_{t+1,j}^{2}$	0	0	$\sigma\sqrt{3}$	$-\sigma\sqrt{3}$	0	0
$\epsilon_{t+1,j}^{3}$	0	0	0	0	$\sigma\sqrt{3}$	$-\sigma\sqrt{3}$

where weights of all nodes are equal, $\omega_{t,j} = 1/6$ for all j.

Since the cost of $M1$ increases with N only linearly, this rule is feasible for approximation of integrals with very large dimensionality. For example, with $N = 100$, the total number of nodes is only $2N = 200$.

The one-node Gauss–Hermite quadrature rule, $Q(1)$, plays a special role in our analysis. This rule is even cheaper than the monomial rules discussed above since there is just one node for any number of exogenous random variables. Typically, there is a trade-off between accuracy and cost of integration methods: having more nodes leads to a more accurate approximation of integrals, but is also more costly. In our numerical experiments, the Gauss–Hermite quadrature rule and monomial rules lead to virtually the same accuracy with an exception of the one-node Gauss–Hermite rule producing slightly less accurate solutions. Overall, the accuracy levels attained by GSSA under deterministic integration methods are orders of magnitude higher than those attained under the Monte Carlo method.[27]

6. Numerical experiments

In this section, we discuss the implementation details of GSSA and describe the results of our numerical experiments. We first solve the representative–agent model of Section 2.1. Then we solve two more challenging applications: a model with rare disasters and a model with multiple countries.

6.1 *Implementation details*

Model's parameters We assume a constant relative risk aversion utility function $u(c_t) = \frac{c_t^{1-\gamma}-1}{1-\gamma}$, with a risk-aversion coefficient $\gamma \in (0, \infty)$, and a Cobb–Douglas production function $f(k_t) = k_t^{\alpha}$, with a capital share $\alpha = 0.36$. The discount factor is $\beta = 0.99$, and the

[27]Quasi-Monte Carlo integration methods based on low-discrepancy sequences of shocks may also give more accurate solutions than Monte Carlo integration methods; see Geweke (1996) for a review.

parameters in (4) are $\rho = 0.95$ and $\sigma = 0.01$. The parameters δ and γ vary across experiments.

Algorithm's parameters The convergence parameter ϖ in the convergence criterion (10) must be chosen by taking into account a trade-off between accuracy and speed in a given application (a too strict criterion wastes computer time, while a too loose criterion reduces accuracy). In our experiments, we find it convenient to adjust ϖ to a degree of the approximating polynomial m and to the damping parameter ξ in (11) by $\varpi = 10^{-4-m}\xi$. The former adjustment allows us to roughly match accuracy levels attainable under different polynomial degrees in our examples. The latter adjustment ensures that different values of ξ imply roughly the same degree of convergence in the time-series solution (note that the smaller is ξ, the smaller is the difference between the series $k_{t+1}^{(p)}$ and $k_{t+1}^{(p-1)}$; in particular, if $\xi = 0$, the series do not change from one iteration to another). In most experiments, we use $\xi = 0.1$, which means that ϖ decreases from 10^{-6} to 10^{-10} when m increases from 1 to 5. To start iterations, we use an arbitrary guess $k_{t+1} = 0.95k_t + 0.05\overline{k}a_t$, where $\overline{k}$ is the steady-state capital. To compute a polynomial solution of degree $m = 1$, we start iterations from a fixed low-accuracy solution; to compute a solution of degree $m \geq 2$, we start from the solution of degree $m - 1$. The initial condition is the steady state $(k_0, a_0) = (\overline{k}, 1)$.

Regularization parameters For RLS-Tikhonov, RLAD-PP, and RLAD-DP, it is convenient to normalize the regularization parameter by the simulation length T and the number of the regression coefficients n. For RLS-Tikhonov, this implies an equivalent representation of the LS problem (21): $\min_b \frac{1}{T}(y - Xb)^\top(y - Xb) + \frac{\eta}{n}b^\top b$, where η reflects a trade-off between the average squared error $\frac{1}{T}(y - Xb)^\top(y - Xb)$ and the average squared coefficient $\frac{1}{n}b^\top b$. Since η is constructed to be invariant to changes in T and n, the same numerical value of η often works well for experiments with different T and n (and thus, different polynomial degrees m). For the RLAD problem (33), we have $\min_b \frac{1}{T}1_T^\top|y - Xb| + \frac{\eta}{n}1_n^\top|b|$.

To select appropriate values of the regularization parameters for our regularization methods, we use the approach that combines the ideas of ridge trace and cross-validation, as described in Section 4.2.2. We specifically search for a value of the regularization parameter that ensures both the numerical stability (convergence) of fixed-point iteration in Stage 1 and the high accuracy of solutions in Stage 2. In our experiments, we typically use the smallest degree of regularization that ensures numerical stability of fixed-point iteration; we find that this choice also leads to accurate solutions.[28]

Results reported, hardware, and software For each experiment, we report the value of a regularization parameter (if applicable) and time necessary for computing a solution, as well as unit-free Euler equation errors (12) on a stochastic simulation of $T^{\text{test}} = 10{,}200$

[28]We tried to automate a search of the regularization parameter by targeting some accuracy criterion in Stage 2. The outcome of the search was sensitive to a realization of shocks and an accuracy criterion (e.g., mean squared error, mean absolute error, maximum error). In the studied models, accuracy improvements were small, while costs increased substantially. We did not pursue this approach.

observations (we discard the first 200 observations to eliminate the effect of initial conditions); see Juillard and Villemot (2011) for a discussion of other accuracy measures. To compute conditional expectations in the test, we use a highly accurate integration method $Q(10)$. We run the experiments on a desktop computer ASUS with Intel® Core™ 2 Quad CPU Q9400 (2.66 GHz). Our programs are written in MATLAB, version 7.6.0.324 (R2008a). To solve the linear-programming problems, we use a routine linprog under the option of an interior-point method.[29] To increase the speed of computations in MATLAB, we use vectorization (e.g., we approximate conditional expectation in all simulated points at once rather than point by point and compute all policy functions at once rather than one by one).

6.2 *Testing numerical stability*

We consider a version of the representative–agent model under $\delta = 1$ and $\gamma = 1$. This model admits a closed-form solution, $k_{t+1} = \alpha\beta a_t k_t^{\alpha}$. To compute conditional expectations, we use the one-node Monte Carlo integration method (44). A peculiar feature of this model is that the integrand of conditional expectation in the Euler equation (7) is equal to k_{t+1} for all possible realizations of a_{t+1}. Since the integrand does not have a forward-looking component, the choice of integration method has little impact on accuracy. We can therefore concentrate on the issue of numerical stability of GSSA.

We consider four nonregularization 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. We use both unnormalized and normalized data, and we consider both ordinary and Hermite polynomials. We use a relatively short simulation length of $T = 3000$ because the primal-problem formulations LAD-PP and RLAD-PP proved to be costly in terms of time and memory. In particular, when T exceeded 3000, our computer ran out of memory. The results are shown in Table 1.

Our stabilization techniques proved to be remarkably successful in the examples considered. When the OLS method is used with unnormalized data and ordinary polynomials, we cannot go beyond the second-degree polynomial approximation. Normalization of variables alone allows us to compute degree 3 polynomial solutions. LS-SVD and LAD with unnormalized data deliver the fourth-degree polynomial solutions. All regularization methods successfully computed degree 5 polynomial approximations. Hermite polynomials ensure numerical stability under any approximation method (all methods considered lead to nearly identical results). The solutions are very accurate with mean errors of order 10^{-9}.

For the regularization methods, we compare the results under 2 degrees of regularization. When a degree of regularization is low, the regularization methods deliver accuracy levels that are comparable or superior to those of the corresponding nonregularization methods. However, an excessively large degree of regularization reduces accuracy

[29]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.

TABLE 1. Stability of GSSA in the representative–agent model with a closed-form solution: the role of approximation method, data normalization, and polynomial family.[a]

	Ordinary Polynomials: Nonregularization Methods						Ordinary Polynomials: Regularization Methods						Hermite Polynomials: Nonregularization Methods		
	Unnormalized Data			Normalized Data			Smaller Regularization			Larger Regularization			Unnormalized Data		
Polynomial Degree	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\text{max}}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\text{max}}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\text{max}}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\text{max}}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\text{max}}$	CPU
	OLS						RLS-Tikhonov, $\eta = 10^{-10}$			RLS-Tikhonov, $\eta = 10^{-7}$			OLS		
1st	−3.52	−2.45	0.8	−3.52	−2.45	1	−3.52	−2.45	1	−3.52	−2.45	1	−3.52	−2.45	1
2nd	−5.46	−4.17	3.1	−5.46	−4.17	3	−5.46	−4.17	3	−5.46	−4.16	3	−5.46	−4.17	4
3rd	–	–	–	−6.84	−5.36	5	−6.84	−5.36	5	−5.85	−4.51	4	−6.84	−5.36	6
4th	–	–	–	–	–	–	−6.97	−5.63	8	−6.12	−4.74	7	−7.97	−6.35	8
5th	–	–	–	–	–	–	–	–	–	−6.22	−4.75	11	−9.09	−7.29	10
	LS-SVD						RLS-TSVD, $\kappa = 10^{8}$			RLS-TSVD, $\kappa = 10^{6}$			LS-SVD		
1st	−3.52	−2.45	0.9	−3.52	−2.45	1	−3.52	−2.45	1	−3.52	−2.45	1	−3.52	−2.45	1
2nd	−5.46	−4.17	3.1	−5.46	−4.17	3	−5.46	−4.17	3	−5.46	−4.17	3	−5.46	−4.17	4
3rd	−6.84	−5.36	4.6	−6.84	−5.36	5	−6.84	−5.36	5	−6.84	−5.36	5	−6.84	−5.36	6
4th	−7.98	−6.37	6.1	−7.97	−6.35	6	−7.97	−6.35	6	−7.20	−5.46	6	−7.97	−6.35	8
5th	–	–	–	−9.12	−7.43	10	−9.08	−7.25	8	−7.64	−5.97	9	−9.08	−7.25	9
	LAD-PP						RLAD-PP, $\eta = 10^{-6}$			RLAD-PP, $\eta = 10^{-4}$			LAD-PP		
1st	−3.57	−2.43	28.6	−3.52	−2.45	16	−3.52	−2.45	15	−3.52	−2.45	15	−3.57	−2.43	30
2nd	−5.56	−4.11	246.5	−5.55	−4.12	92	−5.55	−4.12	127	−5.55	−4.11	100	−5.56	−4.11	243
3rd	−6.98	−5.26	386.8	−6.97	−5.25	245	−6.98	−5.25	321	−6.93	−5.22	263	−6.98	−5.26	379
4th	−7.62	−5.58	558.8	−8.16	−6.11	383	−8.17	−6.13	530	−6.75	−5.06	349	−8.16	−6.13	512
5th	–	–	–	−9.10	−7.02	560	−8.17	−6.15	706	−6.64	−4.97	936	−9.09	−7.05	670
	LAD-DP						RLAD-DP, $\eta = 10^{-6}$			RLAD-DP, $\eta = 10^{-4}$			LAD-DP		
1st	−3.57	−2.43	3.1	−3.52	−2.45	9	−3.52	−2.45	3	−3.52	−2.45	3	−3.57	−2.43	3
2nd	−5.56	−4.11	9.3	−5.55	−4.12	34	−5.55	−4.12	10	−5.55	−4.12	11	−5.56	−4.11	9
3rd	−6.98	−5.26	13.2	−6.97	−5.25	55	−6.98	−5.25	19	−6.93	−5.22	25	−6.98	−5.25	13
4th	–	–	–	−8.14	−6.12	74	−8.17	−6.13	45	−6.75	−5.06	30	−8.15	−6.18	18
5th	–	–	–	–	–	–	−8.17	−6.15	71	−6.64	−4.97	62	−9.26	−7.04	21

[a] $\mathcal{E}_{\text{mean}}$ and $\mathcal{E}_{\text{max}}$ are, respectively, the average and maximum absolute unit-free Euler equation errors (in log10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds); η is the regularization parameter in RLS-Tikhonov, RLAD-PP, and RLAD-DP; κ is the regularization parameter in RLS-TSVD. In all experiments, we use the one-node Monte Carlo integration method MC(1), simulation length $T = 3000$, and damping parameter $\xi = 0.1$.

because the regression coefficients are excessively biased. Finally, under any degree of regularization, RLS-Tikhonov leads to visibly less accurate solutions than the other LS regularization method, RLS-TSVD. This happens because RLS-Tikhonov and RLS-TSVD work with different objects: the former works with a very ill-conditioned matrix $X^\top X$, while the latter works with a better conditioned matrix S.[30]

6.3 *Testing accuracy*

We study a version of the model with $\gamma = 1$ and $\delta = 0.02$. With partial depreciation of capital, the integrand of conditional expectation in the Euler equation (7) does depend on a_{t+1}, and the choice of integration method plays a critical role in the accuracy of solutions. In all the experiments, we use ordinary polynomials and RLS-TSVD with $\kappa = 10^7$. This choice ensures numerical stability, allowing us to concentrate on the accuracy of integration.

We first assess the performance of GSSA based on the Monte Carlo method, MC(J), with $J = 1$ and $J = 2000$. (Recall that MC(1) uses one random draw, and MC(2000) uses a simple average of 2000 random draws to approximate an integral in each simulated point.) We consider four different simulation lengths, $T \in \{100, 1000, 10{,}000, 100{,}000\}$. The results are provided in Table 2.

The performance of the Monte Carlo method is poor. Under MC(1), GSSA can deliver high-degree polynomial approximations only if T is sufficiently large (if T is small, Monte Carlo integration is so inaccurate that simulated series either explode or implode). A 10 times increase in the simulation length (e.g., from $T = 10{,}000$ to $T = 100{,}000$) decreases errors by about a factor of 3. This is consistent with a $\sqrt{T}$ rate of convergence of MC(1); see Section 5.1. Increasing the number of nodes J from 1 to 2000 augments accuracy by about $\sqrt{J}$ and helps restore numerical stability. The most accurate solution is obtained under the polynomial of degree 3, and corresponds to a combination of T and J with the largest number of random draws (i.e., $T = 10{,}000$ and $J = 2000$). Overall, high-degree polynomials do not necessarily lead to more accurate solutions than low-degree polynomials because accuracy is dominated by large errors produced by Monte Carlo integration. Thus, even though our stabilization techniques enable us to compute polynomial approximations of 5 degrees, there is no gain in going beyond the third-degree polynomial if Monte Carlo integration is used.

We next consider the Gauss–Hermite quadrature method $Q(J)$ with $J = 1, 2, 10$. The results change dramatically: all the studied cases become numerically stable and the accuracy of solutions increases by orders of magnitude. $Q(J)$ is very accurate even with just two nodes: increasing the number of nodes from $J = 2$ to $J = 10$ does not visibly reduce the approximation errors in the table. The highest accuracy is attained with the degree 5 polynomials, $T = 100{,}000$, and the most accurate integration method $Q(10)$.

[30]Alternatively, we can apply a Tikhonov-type of regularization directly to S by adding ηI_n, that is, $\widehat{b}(\eta) = V(S + \eta I_n)^{-1} U' y$. This version of Tikhonov regularization produces solutions that are at least as accurate as those produced by LS-SVD. However, in some applications, such as large-scale economies, computing the SVD can be costly or infeasible, and the standard Tikhnonov regularization based on $X'X$ can still be useful.

TABLE 2. Accuracy of GSSA in the representative–agent model with partial depreciation of capital: the role of integration method (Monte Carlo versus Gauss–Hermite quadrature methods).[a]

Polynomial Degree	Monte Carlo Method						Gauss–Hermite Quadrature Method								
	MC(1)			MC(2000)			$Q(1)$			$Q(2)$			$Q(10)$		
	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU
							$T = 100$								
1st	−3.54	−2.80	0.2	−4.35	−3.40	56	−4.36	−3.37	3	−4.35	−3.36	1	−4.35	−3.36	2
2nd	–	–	–	−4.07	−3.06	112	−6.05	−4.93	4	−6.13	−4.90	3	−6.13	−4.90	5
3rd	–	–	–	−3.81	−2.52	200	−6.32	−5.85	5	−7.47	−5.94	4	−7.47	−5.94	6
4th	–	–	–	–	–	–	−6.24	−5.25	6	−6.84	−5.26	6	−6.84	−5.26	8
5th	–	–	–	–	–	–	−6.04	−4.73	7	−6.22	−4.72	10	−6.22	−4.72	11
							$T = 1000$								
1st	−4.02	−3.21	0.4	−4.40	−3.47	425	−4.34	−3.48	3	−4.36	−3.47	3	−4.36	−3.47	3
2nd	−3.71	−2.73	6	−5.52	−4.65	644	−6.06	−4.95	7	−6.16	−4.95	6	−6.16	−4.95	7
3rd	–	–	–	−5.33	−4.23	873	−6.32	−5.92	9	−7.57	−6.21	9	−7.57	−6.21	10
4th	–	–	–	−5.22	−3.81	1383	−6.31	−6.20	10	−8.92	−7.30	11	−8.92	−7.30	13
5th	–	–	–	−5.22	−3.80	1730	−6.32	−6.20	12	−8.53	−6.68	13	−8.53	−6.68	15
							$T = 10{,}000$								
1st	−4.26	−3.37	1	−4.40	−3.48	1236	−4.35	−3.37	15	−4.36	−3.37	16	−4.36	−3.37	20
2nd	−4.42	−3.69	11	−6.04	−4.93	1711	−5.99	−4.94	32	−6.13	−4.92	27	−6.13	−4.92	34
3rd	−4.32	−3.37	25	−6.15	−5.07	2198	−6.32	−5.90	45	−7.48	−6.01	35	−7.48	−6.01	44
4th	−4.31	−2.98	47	−6.08	−4.71	3337	−6.32	−6.18	53	−8.72	−7.10	44	−8.72	−7.10	54
5th	−4.23	−3.30	80	−6.07	−4.70	4551	−6.32	−6.18	62	−8.91	−7.26	51	−8.91	−7.26	63
							$T = 100{,}000$								
1st	−4.39	−3.40	4	–	–	–	−4.36	−3.40	117	−4.37	−3.39	113	−4.37	−3.39	142
2nd	−4.87	−3.96	79	–	–	–	−6.03	−4.94	281	−6.16	−4.94	188	−6.16	−4.94	238
3rd	−4.86	−3.60	184	Ran out of memory			−6.32	−5.93	387	−7.52	−6.04	260	−7.52	−6.04	328
4th	−4.72	−3.43	341	–	–	–	−6.32	−6.19	470	−8.78	−7.18	335	−8.78	−7.18	421
5th	−4.71	−3.44	623	–	–	–	−6.32	−6.19	548	−8.98	−7.35	406	−8.98	−7.35	508

[a] $\mathcal{E}_{\text{mean}}$ and $\mathcal{E}_{\max}$ are, respectively, the average and maximum absolute unit-free Euler equation errors (in log 10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds); T is the simulation length; MC(J) and $Q(J)$ denote J-node Monte Carlo and Gauss–Hermite quadrature integration methods, respectively. In all experiments, we use RLS-TSVD with $\kappa = 10^7$, the ordinary polynomial family, and damping parameter $\xi = 0.1$.

The mean absolute error is around 10^{-9} and is nearly 3 orders of magnitude lower than that attained under Monte Carlo integration. Thus, high-degree polynomials do help increase the accuracy of solutions if integration is accurate.

Note that even the least accurate solution obtained under the Gauss–Hermite quadrature method with $T = 100$ and $J = 1$ is still more accurate than the most, accurate solution obtained under the Monte Carlo method with $T = 10{,}000$ and $J = 2000$. The simulation length T plays a less important role in accuracy and numerical stability of GSSA under $Q(J)$ than under $\mathrm{MC}(J)$ because $Q(J)$ uses simulated points only to construct the domain, while $\mathrm{MC}(J)$ uses such points both to construct the domain and to evaluate integrals. To decrease errors from 10^{-5} to 10^{-9} under the Monte Carlo method $\mathrm{MC}(1)$, we would need to increase the simulation length from $T = 10^4$ to $T = 10^{12}$.

6.4 *Sensitivity of GSSA to the risk-aversion coefficient*

We test GSSA in the model with very low and very high degrees of risk aversion, $\gamma = 0.1$ and $\gamma = 10$. We restrict attention to three regularization methods RLS-Tikhonov, RLS-TSVD, and RLAD-DP (in the limit, these methods include nonregularization methods OLS, LS-SVD, and LAD-DP, respectively). We omit RLAD-PP because of its high cost. In all experiments, we use $T = 10{,}000$ and an accurate integration method $Q(10)$ (however, we found that $Q(2)$ leads to virtually the same accuracy). The results are presented in Table 3.

Under $\gamma = 0.1$, GSSA is stable even under large values of the damping parameter such as $\xi = 0.5$. In contrast, under $\gamma = 10$, GSSA becomes unstable because fixed-point iteration is fragile. One way to enhance numerical stability is to set the damping parameter ξ to a very small value; for example, $\xi = 0.01$ ensures stability under both ordinary and Hermite polynomials. Another way to do so is to choose a different policy function to approximate; see the discussion in Section 4.5.2. We find that using a marginal-utility policy function (instead of the capital policy function) ensures the stability of GSSA under large values of ξ such as $\xi = 0.5$.

Overall, the accuracy of solutions is higher under $\gamma = 0.1$ than under $\gamma = 10$. However, even in the latter case, our solutions are very accurate: we attain mean errors of order 10^{-8}. The accuracy levels attained under the capital and marginal-utility policy functions are similar. RLAD-DP and RLS-TSVD deliver more accurate solutions than RLS-Tikhonov. As for the cost, RLAD-DP is more expensive than the other methods. Finally, the convergence to a fixed point is considerably faster under the capital policy function than under the marginal-utility policy function.

6.5 *Model with rare disasters*

We investigate how the performance of GSSA depends on specific assumptions about uncertainty. We assume that, in addition to standard normally distributed shocks, the productivity level is subject to large negative low-probability shocks (rare disasters). We modify (4) as $\ln a_{t+1} = \rho \ln a_t + (\epsilon_{t+1} + \zeta_{t+1})$, where $\epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2)$, ζ_{t+1} takes values $-\zeta\sigma$ and 0 with probabilities $\overline{p}$ and $1 - \overline{p}$, respectively, and $\zeta > 0$. We assume that $\zeta = 10$

TABLE 3. Sensitivity of GSSA to the risk-aversion coefficient in the representative–agent model.[a]

Polynomial	RLS-Tikhonov				RLS-TSVD				RLAD-DP			
Degree	η	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	κ	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	η	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU
	$\gamma = 0.1$ (capital policy function, ordinary polynomial, dampening $\xi = 0.5$)											
1st	10^{-7}	−4.95	−3.91	8	10^{7}	−4.95	−3.91	8	10^{-7}	−4.95	−3.90	19
2nd		−6.57	−5.32	14		−6.57	−5.32	15		−6.61	−5.31	45
3rd		−6.47	−5.14	22		−7.93	−6.32	22		−7.99	−6.28	154
4th		−6.94	−5.52	35		−9.06	−7.42	30		−9.08	−7.37	317
5th		−6.98	−5.50	86		−8.92	−7.16	39		−9.57	−7.46	885
	$\gamma = 10$ (capital policy function, ordinary polynomial, dampening $\xi = 0.01$)											
1st	10^{-7}	−2.87	−1.76	92	10^{7}	−2.87	−1.76	89	10^{-7}	−2.87	−1.77	194
2nd		−4.25	−2.95	214		−4.25	−2.95	218		−4.24	−2.91	757
3rd		−5.37	−3.99	374		−5.36	−3.96	332		−5.35	−3.89	1799
4th		−5.60	−3.93	681		−6.36	−4.83	448		−6.34	−4.77	4278
5th		–	–	–		−7.13	−5.63	580		−7.25	−5.49	7107
	$\gamma = 10$ (capital policy function, Hermite polynomial, dampening $\xi = 0.01$)											
1st	10^{-7}	−2.87	−1.76	102	10^{7}	−2.87	−1.76	109	10^{-7}	−2.87	−1.77	217
2nd		−4.25	−2.95	318		−4.25	−2.95	332		−4.24	−2.91	809
3rd		−5.36	−3.96	517		−5.36	−3.96	503		−5.35	−3.89	1859
4th		−6.36	−4.83	693		−6.36	−4.83	710		−6.34	−4.77	4267
5th		−7.31	−5.60	921		−7.30	−5.61	926		–	–	–
	$\gamma = 10$ (marginal-utility policy function, ordinary polynomial, dampening $\xi = 0.5$)											
1st	10^{-10}	−2.84	−2.79	256	10^{7}	−2.84	−2.79	442	10^{-7}	−2.72	−2.68	1206
2nd		−3.67	−3.58	645		−3.67	−3.58	1017		−3.55	−3.50	2596
3rd		−4.06	−4.06	1120		−4.06	−4.06	1674		−5.38	−5.37	4331
4th		−4.63	−4.61	1568		−4.81	−4.75	2274		−5.21	−5.18	9093
5th		–	–	–		−5.41	−5.30	3102		−7.75	−6.57	18,596

[a] $\mathcal{E}_{\text{mean}}$ and $\mathcal{E}_{\max}$ are, respectively, the average and maximum absolute unit-free Euler equation errors (in log10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds); γ is the coefficient of risk aversion; η is the regularization parameter in RLS-Tikhonov and RLAD-DP; κ is the regularization parameter in RLS-TSVD. In all experiments, we use the simulation length $T = 10{,}000$, and the 10 node Gauss–Hermite quadrature integration method, $Q(10)$.

and $\overline{p} = 0.02$, that is, a 10% drop in a productivity level occurs with a probability of 2%. These values are in line with the estimates obtained in recent literature on rare disasters; see Barro (2009).

We solve the model with $\gamma = 1$ using three regularization methods (RLS-Tikhonov, RLS-TSVD, and RLAD-DP). We consider both ordinary and Hermite polynomials. We implement a quadrature integration method with $2J$ nodes and weights. The first J nodes are the usual Gauss–Hermite nodes $\{\epsilon_{t+1,j}\}_{j=1,\dots,J}$ and the remaining J nodes correspond to a rare disaster $\{\epsilon_{t+1,j} - \zeta\sigma\}_{j=1,\dots,J}$; the weights assigned to the former J nodes and latter J nodes are adjusted to the probability of a rare disaster by $\{(1-\overline{p})\omega_{t,j}\}_{j=1,\dots,J}$ and $\{\overline{p}\omega_{t,j}\}_{j=1,\dots,J}$, respectively. We use $J = 10$ and $T = 10{,}000$.

In all cases, GSSA is successful in finding solutions; see Table 4. Overall, the errors are larger than in the case of the standard shocks because the ergodic set is larger, and solutions must be approximated and tested on a larger domain; compare Tables 2 and 4.

TABLE 4. The model with rare disasters (10% negative productivity shocks occur with probability 0.01).[a]

Polynomial Degree	RLS-Tikhonov				RLS-TSVD				RLAD-DP			
	η	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\text{max}}$	CPU	κ	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\text{max}}$	CPU	η	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\text{max}}$	CPU
					Ordinary Polynomials							
1st	10^{-6}	−3.97	−2.87	50	10^8	−3.97	−2.87	40	10^{-6}	−3.98	−2.80	81
2nd		−5.47	−4.09	79		−5.47	−4.09	67		−5.61	−4.08	152
3rd		−6.63	−4.70	110		−6.64	−4.71	97		−6.81	−4.67	257
4th		−7.67	−5.89	134		−7.67	−5.83	118		−7.88	−5.50	642
5th		−8.16	−6.30	158		−8.66	−6.54	143		−8.86	−6.12	1193
					Hermite Polynomials							
1st	10^{-6}	−3.97	−2.87	49	10^8	−3.97	−2.87	49	10^{-6}	−3.98	−2.80	88
2nd		−5.47	−4.09	77		−5.47	−4.09	77		−5.61	−4.08	164
3rd		−6.64	−4.71	108		−6.64	−4.71	108		−6.81	−4.67	266
4th		−7.67	−5.83	131		−7.67	−5.83	131		−7.88	−5.51	516
5th		−8.66	−6.54	156		−8.66	−6.54	156		−8.87	−6.42	1013

[a] $\mathcal{E}_{\text{mean}}$ and $\mathcal{E}_{\text{max}}$ are, respectively, the average and maximum absolute unit-free Euler equation errors (in log 10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds); η is the regularization parameter in RLS-Tikhonov and RLAD-DP; κ is the regularization parameter in RLS-TSVD. In all experiments, we use simulation length $T = 10{,}000$, the 10 node Gauss–Hermite quadrature integration method, $Q(10)$, and damping parameter $\xi = 0.1$.

The accuracy levels are still high: the mean absolute errors are of order 10^{-8}. We perform further sensitivity experiments and find that GSSA is numerically stable and delivers accurate solutions for a wide range of the parameters σ, ρ, ζ, and $\overline{p}$.

6.6 *Multicountry model*

We demonstrate the tractability of GSSA in high-dimensional problems. For this, we extend the representative–agent model (2)–(4) to include multiple countries. Each country $h \in \{1, \dots, N\}$ is characterized by capital k_t^h and productivity level a_t^h (i.e., the state space contains $2N$ state variables). The productivity level of a country is affected by both country-specific and worldwide shocks. The world economy is governed by a planner who maximizes a weighted sum of utility functions of the countries' representative consumers. We represent the planner's solution with N capital policy functions and compute their approximations,

$$k_{t+1}^h = K^h(\{k_t^h, a_t^h\}^{h=1,\dots,N}) \approx \Psi^h(\{k_t^h, a_t^h\}^{h=1,\dots,N}; b^h), \quad h = 1, \dots, N, \tag{45}$$

where Ψ^h and b^h are, respectively, an approximating function and a vector of the approximation parameters of country h. A formal description of the multicountry model and implementation details of GSSA are provided in Appendix C. The results are shown in Table 5.

We first compute solutions using GSSA with the one-node Monte Carlo method MC(1). We use RLS-Tikhonov with $\eta = 10^{-5}$ and $T = 10{,}000$. The performance of Monte Carlo integration is again poor. The highest accuracy is achieved under the first-degree

TABLE 5. Accuracy and cost of GSSA in the multicountry model: the effect of dimensionality.[a]

			RLS-Tikh., MC(1), $T = 10{,}000$, $\eta = 10^{-5}$			RLS-TSVD, $M2/M1$, $T = 1000$, $\kappa = 10^{7}$			RLAD-DP, $M1$, $T = 1000$, $\eta = 10^{-5}$			RLS-Tikh., $Q(1)$, $T = 1000$, $\eta = 10^{-5}$		
Number of Countries	Polynomial Degree	Number of Coefficients	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU	$\mathcal{E}_{\text{mean}}$	$\mathcal{E}_{\max}$	CPU
$N = 2$	1st	5	−4.70	−3.13	251	−4.65	−2.99	37	−4.67	−3.01	127	−4.64	−2.99	28
	2nd	15	−4.82	−3.11	1155	−6.01	−3.99	407	−6.06	−4.02	680	−5.79	−4.04	167
	3rd	35	−4.59	−2.42	3418	−7.09	−4.83	621	−7.10	−4.83	1881	−5.50	−3.55	385
	4th	70	−4.57	−2.53	9418	−7.99	−5.63	978	−8.12	−5.60	14,550	−5.73	−3.81	897
	5th	126	−4.53	−2.38	24,330	−8.00	−5.50	2087	−8.17	−5.65	48,061	−5.76	−3.85	2463
$N = 4$	1st	9	−4.59	−3.06	280	−4.72	−3.17	102	−4.73	−3.17	290	−4.71	−3.18	36
	2nd	45	−4.46	−2.87	1425	−6.05	−4.15	1272	−6.08	−4.16	3912	−5.67	−4.22	189
	3rd	165	−4.29	−2.52	11,566	−7.06	−4.89	5518	−7.00	−4.89	95,385	−5.64	−4.03	1092
	4th	495	−4.20	−2.31	58,102	−7.46	−5.23	37,422	–	–	–	−5.64	−4.39	4858
$N = 6$	1st	13	−4.58	−3.12	301	−4.71	−3.08	225	−4.72	−3.10	562	−4.69	−3.09	41
	2nd	91	−4.30	−2.73	1695	−6.06	−4.21	2988	−5.94	−4.11	13,691	−5.62	−4.26	224
	3rd	455	−4.04	−2.29	30,585	−6.96	−4.88	65,663	–	–	–	−5.55	−3.87	3219
$N = 8$	1st	17	−4.56	−3.14	314	−4.73	−3.08	430	−4.74	−3.07	996	−4.72	−3.09	42
	2nd	153	−4.19	−2.63	1938	−6.06	−4.20	5841	−6.08	−4.20	78,928	−5.59	−4.29	278
$N = 10$	1st	21	−4.54	−3.15	341	−4.73	−3.08	773	−4.74	−3.08	1609	−4.72	−3.09	44
	2nd	231	−4.07	−2.59	2391	−6.05	−4.20	10,494	−5.97	−4.14	183,046	−5.56	−4.32	292
$N = 20$	1st	41	−4.55	−3.12	390	−4.77	−2.93	344	−4.79	−2.94	9727	−4.75	−2.93	56
	2nd	861	−3.88	−2.36	7589	−5.48	−3.99	6585	–	–	–	−5.40	−3.94	1079
$N = 100$	1st	201	−4.17	−2.77	1135	−4.63	−3.04	13,846	–	–	–	−4.64	−3.05	225
$N = 200$	1st	401	−3.97	−2.56	2232	−4.60	−3.10	105,121	–	–	–	−4.59	−3.10	1008

[a] $\mathcal{E}_{\text{mean}}$ and $\mathcal{E}_{\max}$ are, respectively, the average and maximum absolute unit-free Euler equation errors (in log10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds); N is the number of countries; Number of Coefficients is the number of polynomial coefficients in the policy function of one country; T is the simulation length. In all experiments, we use ordinary polynomials and normalized data. For RLS-TSVD, we use $M2$ for $N < 20$ and $M1$ for $N = 20$, 100, and 200. For $N = 200$ in experiments RLS-TSVD, $M2/M1$ and RLS-Tikh., $Q(1)$, we use $T = 2000$.

polynomials. This is because polynomials of higher degrees have too many regression coefficients to identify for a given sample size T. Moreover, when N increases, so does the number of coefficients, and the accuracy decreases even further. For example, going from $N = 2$ to $N = 20$ increases the size of the approximation errors by about a factor of 10 under the second-degree polynomial. Longer simulations increase the accuracy but at a high cost.

We next compute solutions using GSSA with the deterministic integration methods. Since such methods do not require long simulations for accurate integration, we use a relatively short simulation length of $T = 1000$ (except for the case of $N = 200$ in which we use $T = 2000$ for enhancing numerical stability). We start with accurate but expensive integration methods (namely, we use the monomial rule $M2$ with $2N^2 + 1$ nodes for $2 \leq N \leq 10$ and we use the monomial rule $M1$ with $2N$ nodes for $N > 10$). The approximation method was RLS-TSVD (with $\kappa = 10^7$). For small-scale economies, $N = 2$, 4, and 6, GSSA computes the polynomial approximations up to degrees 5, 4, and 3, respectively, with maximum absolute errors of $10^{-5.5}$, $10^{-5.2}$, and $10^{-4.9}$, respectively. For medium-scale economies, $N \leq 8$, 10, and 20, GSSA computes the second-degree polynomial approximations with maximum absolute error of 10^{-4}. Finally, for large-scale economies, $N = 100$ and 200, GSSA computes the first-degree polynomial approximations with maximum absolute error of $10^{-2.9}$.

We then compute solutions using RLAD-DP (with $\eta = 10^{-5}$) combined with $M1$. We obtain accuracy levels that are similar to those delivered by our previous combination of RLS-TSVD and $M2$. We observe that RLAD-DP is more costly than the LS methods but is still practical in medium-scale applications. It is possible to increase the efficiency of LAD methods by using techniques developed in the recent literature.[31]

We finally compute solutions using GSSA with a cheap one-node quadrature method, $Q(1)$, and RLS-Tikhonov (with $\eta = 10^{-5}$). For polynomials of degrees larger than 2, the accuracy of solutions is limited. For the first- and second-degree polynomials, the accuracy is similar to that under more expensive integration methods, but the cost is reduced by an order of magnitude or more. In particular, when N increase from 2 to 20, the running time increases only from 3 to 18 minutes. Overall, RLS-Tikhonov is more stable in large-scale problems than RLS-TSVD (because SVD becomes costly and numerically unstable).

The accuracy of GSSA solutions is comparable to the highest accuracy attained in the comparison analysis of Kollmann et al. (2011). GSSA fits a polynomial on a relevant domain (the ergodic set) and as a result, can get a better fit on the relevant domain than methods fitting polynomials on other domains.[32] A choice of domain is especially important for accuracy under relatively rigid low-degree polynomials. In particular, linear solutions produced by GSSA are far more accurate than the first- and second-order perturbation methods of Kollmann, Kim, and Kim (2011) that in a similar model produce

[31]Tits, Absil, and Woessner (2006) proposed a constraint-reduction scheme that can drastically reduce computational cost per iteration of linear-programming methods.

[32]An advantage of focusing on the ergodic set is illustrated by Judd, Maliar, and Maliar (2010) in the context of a cluster grid algorithm. In a model with only two state variables, solutions computed on the ergodic set are up to 10 times more accurate than those computed on the rectangular grid containing the ergodic set.

approximation errors of $10^{-1.7}$ and $10^{-2.66}$, respectively; see Table 2 in a web appendix of the comparison analysis of Kollmann et al. (2011), http://www.sciencedirect.com/science/journal/01651889.[33] The cost of GSSA depends on the integration and approximation methods and the degree of the approximating polynomial, as well as the simulation length. There is a trade-off between accuracy and speed, and cheap versions of GSSA are tractable in problems with very high dimensionality. Finally, GSSA is highly parallelizable.[34]

7. Conclusion

Methods that operate on an ergodic set have two potential advantages compared to methods that operate on domains that are exogenous to the models. The first advantage is in terms of cost: ergodic-set methods compute solutions only in a relevant domain—the ergodic set realized in equilibrium—while exogenous-domain methods compute solutions both inside and outside the relevant domain, and spend time computing solutions in unnecessary points. The second advantage is in terms of accuracy: ergodic-set methods fit a polynomial in a relevant domain, while exogenous-domain methods fit the polynomial in generally larger domains, and face a trade-off between the fit (accuracy) inside and outside the relevant domain.

Stochastic simulation algorithms in previous literature (based on standard LS approximation methods and Monte Carlo integration methods) did not benefit from the above advantages. Their performance was severely handicapped by two problems: numerical instability (because of multicollinearity) and large integration errors (because of low accuracy of Monte Carlo integration). GSSA fixes both of these problems: First, GSSA relies on approximation methods that can handle ill-conditioned problems; this allows us to stabilize stochastic simulation and to compute high-degree polynomial approximations. Second, GSSA uses a generalized notion of integration that includes both Monte Carlo and deterministic (quadrature and monomial) integration methods; this allows us to compute integrals very accurately. GSSA has shown great performance in the examples considered. It extends the speed–accuracy frontier attained in the related literature, it is tractable for problems with high dimensionality, and it is very simple to program. GSSA appears to be a promising method for many economic applications.

References

Aiyagari, R. (1994), "Uninsured idiosyncratic risk and aggregate saving." *Quarterly Journal of Economics*, 109, 659–684. [174]

[33]Maliar, Maliar, and Villemot (2011) implement a perturbation-based method which is comparable in accuracy to global solution methods. This is a hybrid method that computes some policy functions locally (using perturbation) and computes the remaining policy functions globally (using analytical formulas and numerical solvers).

[34]For example, Creel (2008) developed a parallel computing toolbox which reduces the cost of a simulation-based PEA, studied in Maliar and Maliar (2003b), by running simulations on a cluster of computers.

Aruoba, S., J. Fernandez-Villaverde, and J. Rubio-Ramírez (2006), "Comparing solution methods for dynamic equilibrium economies." *Journal of Economic Dynamics and Control*, 30, 2477–2508. [174]

Asmussen, S. and P. Glynn (2007), *Stochastic Simulation: Algorithms and Analysis*. Springer, New York. [174]

Barro, R. (2009), "Rare disasters, asset prices, and welfare costs." *American Economic Review*, 99, 243–264. [202]

Brown, P. (1993), *Measurement, Regression, and Calibration*. Clarendon Press, Oxford. [186]

Charnes, A., W. Cooper, and R. Ferguson (1955), "Optimal estimation of executive compensation by linear programming." *Management Science*, 1, 138–151. [187]

Christiano, L. and D. Fisher (2000), "Algorithms for solving dynamic models with occasionally binding constraints." *Journal of Economic Dynamics and Control*, 24, 1179–1232. [173, 174, 177, 182, 192]

Creel, M. (2008), "Using parallelization to solve a macroeconomic model: A parallel parameterized expectations algorithm." *Computational Economics*, 32, 343–352. [206]

Davidson, R. and J. MacKinnon (1993), *Estimation and Inference in Econometrics*. Oxford University Press, New York. [185]

Den Haan, W. (1990), "The optimal inflation path in a Sidrauski-type model with uncertainty." *Journal of Monetary Economics*, 25, 389–409. [192]

Den Haan, W. (2010), "Comparison of solutions to the incomplete markets model with aggregate uncertainty." *Journal of Economic Dynamics and Control*, 34, 4–27. [174, 175]

Den Haan, W. and A. Marcet (1990), "Solving the stochastic growth model by parameterized expectations." *Journal of Business and Economic Statistics*, 8, 31–34. [182]

Dielman, T. (2005), "Least absolute value: Recent contributions." *Journal of Statistical Computation and Simulation*, 75, 263–286. [189]

Eckart, C. and G. Young (1936), "The approximation of one matrix by another of lower rank." *Psychometrika*, 1, 211–218. [190]

Eldén, L. (2007), *Matrix Methods in Data Mining and Pattern Recognition*. SIAM, Philadelphia. [191]

Fair, R. and J. Taylor (1983), "Solution and maximum likelihood estimation of dynamic nonlinear rational expectation models." *Econometrica*, 51, 1169–1185. [174]

Ferris, M., O. Mangasarian, and S. Wright (2007), *Linear Programming With MATLAB*. MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania. [188]

Gaspar, J. and K. Judd (1997), "Solving large scale rational expectations models." *Macroeconomic Dynamics*, 1, 45–75. [173, 174]

Geweke, J. (1996), "Monte Carlo simulation and numerical integration." In *Handbook of Computational Economics* (H. Amman, D. Kendrick, and J. Rust, eds.), 733–800, Elsevier Science, Amsterdam. [195]

Golub, G. and C. Van Loan (1996), *Matrix Computations*. The John Hopkins University Press, Baltimore, Maryland. [185]

Hadi, A. and R. Ling (1998), "Some cautionary notes on the use of principal components regression." *American Statistician*, 52, 15–19. [191]

Heer, B. and A. Maussner (2008), "Computation of business cycle models: A comparison of numerical methods." *Macroeconomic Dynamics*, 12, 641–663. [174]

Hoerl, A. and R. Kennard (1970), "Ridge regression: Biased estimation for nonorthogonal problems." *Technometrics*, 12, 69–82. [186]

Judd, K. (1992), "Projection methods for solving aggregate growth models." *Journal of Economic Theory*, 58, 410–52. [174, 181, 182, 191]

Judd, K. (1998), *Numerical Methods in Economics*. MIT Press, Cambridge, Massachusetts. [173, 181, 192, 194]

Judd, K. and S. Guu (1993), "Perturbation solution methods for economic growth models." In *Economic and Financial Modeling With Mathematica* (H. Varian, ed.), 80–103, Springer-Verlag, Santa Clara, California. [174]

Judd, K., L. Maliar, and S. Maliar (2009), "Numerically stable stochastic simulation methods for solving dynamic economic models." Working Paper 15296, NBER. [193]

Judd, K., L. Maliar, and S. Maliar (2010), "A cluster-grid projection method: Solving problems with high dimensionality." Working Paper 15965, NBER. [182, 205]

Judd, K., L. Maliar, and S. Maliar (2011), "One-node quadrature beats Monte Carlo: A generalized stochastic simulation algorithm." Working Paper 16708, NBER. [193]

Juillard, M. and S. Villemot (2011), "Multi-country real business cycle models: Accuracy tests and testing bench." *Journal of Economic Dynamics and Control*, 35, 178–185. [197]

Koenker, R. and G. Bassett (1978), "Regression quantiles." *Econometrica*, 46, 33–50. [187]

Kollmann, R., S. Kim, and J. Kim (2011), "Solving the multi-country real business cycle model using a perturbation method." *Journal of Economic Dynamics and Control*, 35, 203–206. [205]

Kollmann, R., S. Maliar, B. Malin, and P. Pichler (2011), "Comparison of solutions to the multi-country real business cycle model." *Journal of Economic Dynamics and Control*, 35, 186–202. [174, 177, 193, 205, 206]

Krueger, D. and F. Kubler (2004), "Computing equilibrium in OLG models with production." *Journal of Economic Dynamics and Control*, 28, 1411–1436. [182]

Krusell, P. and A. Smith (1998), "Income and wealth heterogeneity in the macroeconomy." *Journal of Political Economy*, 106, 868–896. [174, 175]

Maliar, L. and S. Maliar (2003a), "The representative consumer in the neoclassical growth model with idiosyncratic shocks." *Review of Economic Dynamics*, 6, 362–380. [182]

Maliar, L. and S. Maliar (2003b), "Parameterized expectations algorithm and the moving bounds." *Journal of Business and Economic Statistics*, 21, 88–92. [206]

Maliar, L. and S. Maliar (2005), "Solving nonlinear stochastic growth models: Iterating on value function by simulations." *Economics Letters*, 87, 135–140. [174, 177]

Maliar, L., S. Maliar, and F. Valli (2010), "Solving the incomplete markets model with aggregate uncertainty using the Krusell–Smith algorithm." *Journal of Economic Dynamics and Control*, 34, 42–49. [175, 177]

Maliar, L., S. Maliar, and S. Villemot (2011), "Taking perturbation to the accuracy frontier: A hybrid of local and global solutions." Dynare, Working Paper 6. [206]

Maliar, S., L. Maliar, and K. Judd (2011), "Solving the multi-country real business cycle model using ergodic set methods." *Journal of Economic Dynamic and Control*, 35, 207–228. [177, 182]

Malin, B., D. Krueger, and F. Kubler (2011), "Solving the multi-country real business cycle model using a Smolyak-collocation method." *Journal of Economic Dynamics and Control*, 35, 229–239. [182]

Marcet, A. (1988), "Solving non-linear models by parameterizing expectations." Unpublished manuscript, Carnegie Mellon University, Graduate School of Industrial Administration. [174, 182, 192, 193]

Marcet, A. and G. Lorenzoni (1999), "The parameterized expectation approach: Some practical issues." In *Computational Methods for Study of Dynamic Economies* (R. Marimon and A. Scott, eds.), 143–171, Oxford University Press, New York. [177, 182]

Marimon, R. and A. Scott (1999), *Computational Methods for Study of Dynamic Economies*. Oxford University Press, New York. [173]

Miranda, M. and P. Fackler (2002), *Applied Computational Economics and Finance*. MIT Press, Cambridge, Massachusetts. [173]

Miranda, M. and P. Helmberger (1988), "The effects of commodity price stabilization programs." *American Economic Review*, 78, 46–58. [181]

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. [189]

Pichler, P. (2011), "Solving the multi-country real business cycle model using a monomial rule Galerkin method." *Journal of Economic Dynamics and Control*, 35, 240–251. [174, 182]

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. [197]

Rust, J. (1996), "Numerical dynamic programming in economics." In *Handbook of Computational Economics* (H. Amman, D. Kendrick, and J. Rust, eds.), 619–722, Elsevier Science, Amsterdam. [173]

Santos, M. (1999), "Numerical solution of dynamic economic models." In *Handbook of Macroeconomics* (J. Taylor and M. Woodford, eds.), 312–382, Elsevier Science, Amsterdam. [173]

Smith, A. (1993), "Estimating nonlinear time-series models using simulated vector autoregressions." *Journal of Applied Econometrics*, 8, S63–S84. [174]

Taylor, J. and H. Uhlig (1990), "Solving nonlinear stochastic growth models: A comparison of alternative solution methods." *Journal of Business and Economic Statistics*, 8, 1–17. [173]

Tits, A., P. Absil, and W. Woessner (2006), "Constraint reduction for linear programs with many inequality constraints." *SIAM Journal on Optimization*, 17, 119–146. [205]

Wagner, H. (1959), "Linear programming techniques for regression analysis." *American Statistical Association Journal*, 54, 206–212. [188]

Wang, L., M. Gordon, and J. Zhu (2006), "Regularized least absolute deviations regression and an efficient algorithm for parameter tuning." In *Proceedings of the Sixth International Conference on Data Mining*, 690–700, IEEE Computer Society, Los Alamitos, California. [188]

Wright, B. and J. Williams (1984), "The welfare effects of the introduction of storage." *Quarterly Journal of Economics*, 99, 169–192. [181]

Submitted August, 2009. Final version accepted January, 2011.