Avoiding the Curse of Dimensionality: Numerical Methods for Large-Scale Dynamic Problems

Albert Rodriguez Sala

UAB

September 29, 2020

Outline

Main source

Maliar and Maliar (2014)

Numerical methods for large-scale dynamic economic models. Chapter 7 In Handbook of computational economics (Vol. 3, pp. 325-477). Elsevier.

The Challenge of High-Dimensional Dynamic Economic Models (HD-DEM).

Many of the existing numerical solution methods in the literature are subject to **the curse of dimensionality**: *computational expense grows exponentially with the dimensionality of state space.* HD-DEM represent 3 main challenges to numerical methods:

- **Decision functions** depend on more arguments and are increasingly costly to approximate (we may also have more functions).
- **Cost of integration** increases as the number of exogenous random variables increases. (also, more integrals to approximate).
- Larger models are characterized by larger and more **complex systems of equations**.

Conventional Numerical Solution Methods

Projection Methods

- **Definition**: Projection methods build on tensor-product rules. They are accurate and fast in models with few state variables but suffer curse of dimensionality.
- **Problems in HD-DEM**: They use expensive tensor-product rules both for interpolating decision functions and approximating integrals. They use expensive Newton's methods for solving nonlinear systems of equations.

Conventional Numerical Solution Methods

Simulation Methods

- **Definition**: Simulation methods construct the grid based on simulated data points. They rely on Monte Carlo integration and least-squares learning.
- Marcet (1988) proposes a Parametrized Expectation Algorithm (PEA) for solving economic models with uncertainty.
- Simulation techniques are used in the context of many other solution methods (Aigagary 1994, Krusell and Smith 1998) and in the context of learning models (Marcet and Sargent 1989, etc).
- **Problems in HD-DEM** Monte Carlo integration has a low rate of convergence: an infeasibly long simultion is needed to attain high accuracy levels. Also least-squares learning is often numerically unstable.

Perturbation Methods

- **definition**: Perturbation methods solve models in a steady state using Taylor expansions of Model's equations.
- **Problems in HD-DEM**: the accuracy of local solutions may deteriorate dramatically away from the steady-state point in which such solutions are computed. particularly in the presence of strong non-linearities and kinks in decision functions.

Key ideas and techniques for solving HD-DEM

- **Avoid tensor-product grid**: Efficient non-product techniques for representing, interpolating, and approximating functions.
- Accurate, low-cost **monomial integration** formulas.
- **Derivative-free solvers**.
- Endogenous Grid Method (**EGM**) and Envelope Condition (**ECM**) vs VFI and time-iteration.
- **Precomputation techniques**: precomputation of intratemporal choice manifold (Maliar and Maliar). Precomputation of integrals Judd (2011d) and imperfect aggregation (Maliar and Maliar, 2014).
- **Parallel computation**.

Accuracy in large-scale DP problems

In the context of complex large-scale models, it is typically hard to prove convergence theorems and to derive error bounds analytically. Use two-stage procedure outlined in judd et al.(2011b).

SECTION: Global solution methods for HD-DEM

3 proposed solutions based on conventional Numerical Solutions

- **Smolyak's (1963) Sparse grid method**(global solution): Replace an expensive tensor-product grid with a low nonproduct Smolyak Sparse grid.
- **Generalized Stochastic Simulation Algorithm (GSSA)**(global solution): Introduce a generalized stochastic simulation algorithm in which inaccurate monte Carlo integration is replaced with accurate deterministic integration and in which unstable least-squares learning is replaced with numerically stable regression methods.
- ε-**distinguishable set method** (hybrid of perturbation and global solutions): Merge stochastic simulation and projection: it uses stochastic simulation to identify a high-probability area of the state space and it uses projection-style analysis to accurately solve the model in this area.

Smolyak's (1963) Sparse grid method

- Smolyak (1963) introduces a numerical technique for integrating, representing, and interpolating smooth functions on multidimensional hypercubes. The Smolyak method constructs multidimensional grid points and basis functions using nonproduct rules(grows polynomially with the dimensionality of the hypercube).
- The Smolyak grid was incorporated into numerical methods for solving dynamic economic models by Krueger and Kubler (2004)
- Main reference: Judd, Maliar, Maliar and Valero (2014). *Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain*. Journal of Economic Dynamics and Control, 44, 92-
- Code in Matlab: Serguei Maliar website paper JMMV(2014)
- Code in Python and Julia Github EconForge by C. Coleman and S. Lyon.

Smolyak's (1963) Sparse grid method

Key idea: *Among all elements produced by a tensor product rule, some elements are more important for the quality of approximation than others. The Smolyak method orders all the tensor-product elements by their potential importance for the quality of approximation and selects the most important ones.*

d	Tensor-product grid with 5^d points	Smolyak grid		
		$\mu = 1$	$\mu = 2$	$\mu = 3$
1	5	3	5	9
2	25	5	13	29
10	9,765,625	21	221	1581
20	95,367,431,640,625	41	841	11561

Smolyak's (1963): Initial steps

- A Smolyak grid is a set of points in the $[-1, 1]^d$ hypercube defined by the dimensions (d) and a level of approximation (μ):

$$H^{d,\mu}$$

- the level of approximation (μ) controls how many tensor-product elements are included into the Smolyak grid.
- Let X be the state space of the economic model, then we need to define a function st: $\Phi : X \to [-1, 1]^d$ and its inverse. JMMV(2014) propose 2 mappings: conventional rectangular domain and a rectangular domain constructed on principal components of simulated series.
- JMMV(2014) Use Chebyshev polynomials and their extrema. We need grid-polynomial families of grid points and basis functions that do not lead to ill-conditioned inverse problems.

Smolyak's (1963): Construct Smolyak Grid

Consider $d = 2$ and Chebyshev extrema grid points. Then, to construct the Smolyak grid:

- **Step 1:** Use the unidemensional grid points to construct a sequence of sets S_1, S_2, ... satisfying two conditions:
  1. A set S_i, has $m(i) = 2^{i-1} + 1$ points for $i \geq 2$ and $m(1) = 1$.
  2. $S_i \subseteq S_{i+1}$ (nested property).

$$i = 1 : S_1 = \{0\}$$

$$i = 2 : S_2 = \{-1, 0, 1\}$$

$$i = 3 : S_3 = \left\{ -1, -\sqrt{2}^{-1}, 0, 1, +\sqrt{2}^{-1} \right\}$$

$$i = 4 : S_4 = \left\{ -1, -\frac{\sqrt{2-\sqrt{2}}}{2}, -\sqrt{2}^{-1}, -\frac{\sqrt{2+\sqrt{2}}}{2}, 0, ... \right.$$
$$\left. + \frac{\sqrt{2-\sqrt{2}}}{2}, +\sqrt{2}^{-1}, + \frac{\sqrt{2+\sqrt{2}}}{2}, 1 \right\}$$

Smolyak's (1963): Construct Smolyak grid

- **Step 2:** : Construct all possible two-dimensional tensor products using elements from the nested sets.

		$i_2 = 1$	$i_2 = 2$	$i_2 = 3$
	$S_{i_1} \setminus S_{i_2}$	0	$-1, 0, 1$	$-1, \frac{-1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 1$
$i_1 = 1$	0	$(0,0)$	$(0,-1)\,,(0,0)\,,(0,1)$	$(0,-1)\,,(0,\frac{-1}{\sqrt{2}}),(0,0)\,,(0,\frac{1}{\sqrt{2}}),(0,1)$
$i_1 = 2$	-1 0 1	$(-1,0)$ $(0,0)$ $(1,0)$	$(-1,-1)\,,(-1,0)\,,(-1,1)$ $(0,-1)\,,(0,0)\,,(0,1)$ $(1,-1)\,,(1,0,)\,,(1,1)$	$(-1,-1)\,,(-1,\frac{-1}{\sqrt{2}}),(-1,0)\,,(-1,\frac{1}{\sqrt{2}}),(-1,1)$ $(0,-1)\,,(0,\frac{-1}{\sqrt{2}}),(0,0)\,,(0,\frac{1}{\sqrt{2}}),(0,1)$ $(1,-1),(1,\frac{-1}{\sqrt{2}}),(1,0)\,,\left(1,\frac{1}{\sqrt{2}}\right)(1,1)$
$i_1 = 3$	-1 $\frac{-1}{\sqrt{2}}$ 0 $\frac{1}{\sqrt{2}}$ 1	$(-1,0)$ $\left(\frac{-1}{\sqrt{2}},0\right)$ $(0,0)$ $\left(\frac{1}{\sqrt{2}},0\right)$ $(1,0)$	$(-1,-1)\,,(-1,0)\,,(-1,1)$ $(\frac{-1}{\sqrt{2}},-1),(\frac{-1}{\sqrt{2}},0),(\frac{-1}{\sqrt{2}},1)$ $(0,-1)\,,(0,0)\,,(0,1)$ $(\frac{1}{\sqrt{2}},-1),(\frac{1}{\sqrt{2}},0),(\frac{1}{\sqrt{2}},1)$ $(1,-1),(1,0)\,,(1,1)$	$(-1,-1)\,,(-1,\frac{-1}{\sqrt{2}}),(-1,0)\,,(-1,\frac{1}{\sqrt{2}}),(-1,1)$ $(\frac{-1}{\sqrt{2}},-1),(\frac{-1}{\sqrt{2}},\frac{-1}{\sqrt{2}}),(\frac{-1}{\sqrt{2}},0),(\frac{-1}{\sqrt{2}},\frac{1}{\sqrt{2}}),(\frac{-1}{\sqrt{2}},1)$ $(0,-1)\,,(0,\frac{-1}{\sqrt{2}}),(0,0)\,,(0,\frac{1}{\sqrt{2}}),(0,1)$ $(\frac{1}{\sqrt{2}},-1),(\frac{1}{\sqrt{2}},\frac{-1}{\sqrt{2}}),(\frac{1}{\sqrt{2}},0),(\frac{1}{\sqrt{2}},\frac{-1}{\sqrt{2}}),(\frac{1}{\sqrt{2}},1)$ $(1,-1),(1,\frac{-1}{\sqrt{2}}),(1,0)\,,\left(1,\frac{1}{\sqrt{2}}\right)(1,1)$

Smolyak's (1963): Construct Smolyak grid

- **Step 3:** select the non-repeated elements in the table that satisfy the Smolyak rule:

$$d \leq i_1 + i_2 \leq d + \mu$$

if $\mu = 1$ then:

$$H^{2,1}\{(0,0),(-1,0),(1,0),(0,-1),(0,1)\}$$

if $\mu = 2$ then:

$$H^{2,2} = \{(-1,-1),(-1,0),(-1,1),(0,-1),(0,0),(0,1),(1,-1),(1,0),$$

$$(1,1),(-\sqrt{2}^{-1},0),(+\sqrt{2}^{-1},0),(0,-\sqrt{2}^{-1}),(0,+\sqrt{2}^{-1})\}$$

Smolyak's (1963): JMMV(2014) approach

The conventional Smolyak method using nested sets is inefficient. First, it creates a list of tensor products with many repeated elements and then, it eliminates the repetitions. As d and μ increase this supposes an increase in the computational expense.
To avoid so, JMMV(2014) propose to construct a sequence of sets such that the sets are disjoint: $A_1 = S_1$ and $A_i = S_i \setminus S_{i-1}$ for $i \geq 2$:

Smolyak's (1963): Smolyak method with ansiotropic grid

- the Smolyak conventional method treats all dimensions equally in the sense that it uses the same number of grid points and basis functions for all variables. However, in economics variables often enter asymetrically in the decision functions. JMMV(2014) propose a method that enables us to separately choose accuracy levels for each dimension. They called the **Smolyak method with ansiotropic grid**.
- Consider that our first dimension (capital) is more important in the decision functions than the second one (discrete shocks).

Smolyak's (1963): Smolyak method with ansiotropic grid

Construct a tensor product where the sequence of sets in one dimension is longer than in the second one.

	$A_{i_1} \setminus A_{i_2}$	$i_2 = 1$	$i_2 = 2$
		0	$-1, 1$
$i_1 = 1$	0	$(0,0)$	$(0,-1), (0,1)$
$i_1 = 2$	-1 1	$(-1,0)$ $(1,0)$	$(-1,-1), (-1,1)$ $(1,-1), (1,1)$
$i_1 = 3$	$\frac{-1}{\sqrt{2}}$ $\frac{1}{\sqrt{2}}$	$\left(\frac{-1}{\sqrt{2}}, 0\right)$ $\left(\frac{1}{\sqrt{2}}, 0\right)$	$\left(\frac{-1}{\sqrt{2}}, -1\right), \left(\frac{-1}{\sqrt{2}}, 1\right)$ $\left(\frac{1}{\sqrt{2}}, -1\right), \left(\frac{1}{\sqrt{2}}, 1\right)$
$i_1 = 4$	$\frac{-\sqrt{2+\sqrt{2}}}{2}$ $\frac{-\sqrt{2-\sqrt{2}}}{2}$ $\frac{\sqrt{2-\sqrt{2}}}{2}$	$\left(\frac{-\sqrt{2+\sqrt{2}}}{2}, 0\right)$ $\left(\frac{-\sqrt{2-\sqrt{2}}}{2}, 0\right)$ $\left(\frac{\sqrt{2-\sqrt{2}}}{2}, 0\right)$	$\left(\frac{-\sqrt{2+\sqrt{2}}}{2}, -1\right), \left(\frac{-\sqrt{2+\sqrt{2}}}{2}, 1\right)$ $\left(\frac{-\sqrt{2-\sqrt{2}}}{2}, -1\right), \left(\frac{-\sqrt{2-\sqrt{2}}}{2}, 1\right)$ $\left(\frac{\sqrt{2-\sqrt{2}}}{2}, -1\right), \left(\frac{\sqrt{2-\sqrt{2}}}{2}, 1\right)$

Smolyak's (1963): Smolyak method with ansiotropic grid

- **Step 3*:** select the elements in the table that satisfy the Smolyak rule under ansiotropic case:

$$d \leq i_1 + i_2 \leq \dot{+}\mu^{max}$$

$$i_1 \leq \mu_1 + 1, i_2 \leq \mu_2 + 1$$

Smolyak's (1963): Lagrange Interpolation

- For the construction of the Smolyak polynomials I follow the efficient implementation of the Smolyak method in JMMV(2014) avoiding nested sets.
- Smolyak polynomials are constructed by following similar steps to the grid formation with the added step of obtaining the coefficients on the basis functions.
- Let $P^{d,\mu}$ denote a Smolyak basis function.

Smolyak's (1963): Lagrange Interpolation

Step 1: cosntruct disjoint sets $A_1, ..., A_i ...$ that contain unidimensional basis functions:

$$i = 1 : A_1 = \{1\}$$

$$i = 2 : A_2 = \{\psi_2(x), \psi_3(x)\}$$

$$i = 3 : A_3 = \{\psi_4(x), \psi_5(x)\}$$

$$i = 4 : A_4 = \{\psi_6(x), \psi_7(x), \psi_8(x), \psi_9(x)\}$$

Smolyak's (1963): Lagrange Interpolation

Step 2: Construct table of the tensor products of disjoint sets of Chebyshev polynomial basis:

	$A_{i_1} \setminus A_{i_2}$	$i_2 = 1$	$i_2 = 2$	$i_2 = 3$
		1	$\psi_2(y), \psi_3(y)$	$\psi_4(y), \psi_5(y)$
$i_1 = 1$	1	1	$\psi_2(y), \psi_3(y)$	$\psi_4(y), \psi_5(y)$
$i_1 = 2$	$\psi_2(x)$ $\psi_3(x)$	$\psi_2(x)$ $\psi_3(x)$	$\psi_2(x)\psi_2(y), \psi_2(x)\psi_3(y)$ $\psi_3(x)\psi_2(y), \psi_3(x)\psi_3(y)$	$\psi_2(x)\psi_4(y), \psi_2(x)\psi_5(y)$ $\psi_3(x)\psi_4(y), \psi_3(x)\psi_5(y)$
$i_1 = 3$	$\psi_4(x)$ $\psi_5(x)$	$\psi_4(x)$ $\psi_5(x)$	$\psi_4(x)\psi_2(y), \psi_4(x)\psi_3(y)$ $\psi_5(x)\psi_2(y), \psi_5(x)\psi_3(y)$	$\psi_4(x)\psi_4(y), \psi_4(x)\psi_5(y)$ $\psi_5(x)\psi_4(y), \psi_5(x)\psi_5(y)$

Smolyak's (1963): Lagrange Interpolation

- **Step 3:** Apply the Smolyak rule (i.e. $d \leq i_1 + i_2 \leq d + \mu$) to obtain the set of Smolyak basis functions $P^{d,\mu}$:
- If $\mu = 1$, then the 3 cells that satisfy the Smolyak restriction are $(i_1 = 1, i_2 = 1)$, $(i_1 = 1, i_2 = 2)$, $(i_1 = 2, i_2 = 1)$ so the corresponding 5 Smolyak basis are:

$$P^{2,1} = \{1, \psi_2(x), \psi_3(x), \psi_2(y), \psi_3(y)\}$$

- *If* $\mu = 2$:

$$P^{2,2} = \{1, \psi_2(x), \psi_3(x), \psi_2(y), \psi_3(y), \psi_4(x), \psi_5(x), \psi_4(y), \psi_5(y),$$

$$\psi_2(x)\psi_2(y), \psi_2(x)\psi_3(y), \psi_3(x)\psi_2(y), \psi_3(x)\psi_3(y)\}$$

Smolyak's (1963): Lagrange Interpolation

Step 4: Compute the coefficients of the basis functions, b.
let $f(:b)$ be a polynomial of the form:

$$\hat{f}(x;b) = \sum_{n=1}^{M} b_n \Psi_n(x)$$

We compute b so that the true function f, and its approximation $\hat{f}(;b)$ coincide in all grid points:

$$\begin{bmatrix} f(x_1) \\ \cdots \\ f(x_M) \end{bmatrix} = \begin{bmatrix} \widehat{f}(x_1;b) \\ \cdots \\ \widehat{f}(x_M;b) \end{bmatrix} = \begin{bmatrix} \Psi_1(x_1) & \cdots & \Psi_M(x_1) \\ \cdots & \ddots & \cdots \\ \Psi_1(x_M) & \cdots & \Psi_M(x_M) \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ \cdots \\ b_M \end{bmatrix}.$$

We construct the Smolyak interpolating coefficients solving the previous inverse problem numerically:

$$\begin{bmatrix} b_1 \\ \cdots \\ b_M \end{bmatrix} = \begin{bmatrix} \Psi_1(x_1) & \cdots & \Psi_M(x_1) \\ \cdots & \ddots & \cdots \\ \Psi_1(x_M) & \cdots & \Psi_M(x_M) \end{bmatrix}^{-1} \begin{bmatrix} f(x_1) \\ \cdots \\ f(x_M) \end{bmatrix}.$$

Example: Python file: smolyaktest.py

- Compare tensor-product grid with Smolyak under different μ.
- Compare symmetric vs ansiotropic case.
- Interpolize smooth and kink functions. Do it for different dimensions and μ:

Smooth function:

$$y = x_1^{1/2} + x_2^{1/2} ...$$

kinky function:

$$\text{if } x_i < 200,\ y = 0$$

$$\text{else },\ y = x_1^{1/2} + x_2^{1/2} ...$$

Generalized Stochastic Simulation Algorithm (GSSA)

- **Stochastic simulation methods** compute solutions on a set of simulated points using Monte Carlo integration. Useful since it computes solutions only in the part of the state space that is visited in equilibrium-the ergodic set.
- **Drawback**: They rely on numerically unstable regression methods and unfeasible (or unacurate) Monte Carlo integration.
- **Solution: GSSA** Judd, Maliar and Maliar (2011).
  - "Instead of standard least-squares approximation methods, we examine a variety of alternatives, including leastsquares methods using **singular value decomposition and Tikhonov regularization, least-absolute deviations methods, and principal component regression method**".
  - Instead of conventional Monte Carlo integration, we use **accurate quadrature and monomial integration**.

ε-Distinguishable Set (EDS)

- Simulated points used by GSSA occupy a much smaller area than the hypercube (like Smolyak method) and allow us to focus on a relevant solution domain. However, a set of simulated points itself is not efficient because it is unevenly spaced, has closely-located/redundant points, and in low-density regions.
- Judd, Maliar, Maliar (2012a) introduce an **ε-distinguishable set (EDS) technique** that selects a subset of points situated at distance at least ε from one another.This construction combines the best features of stochastic simulation and hypercube grids: **combines an adaptive geometry with efficient discretization**.

ε-Distinguishable Set (EDS)

- **Hybrid solution of stochastic simulation and projection methods:**
  1. Use simulation to approximate the ergodic measure of the solution.
  2. construct a fixed grid covering the support of the constructed ergodic measure.
  3. use projection techniques to accurately solve the model on that grid
- **Global solution:** Can hold strong non-linearities and inequality constraints. JMM (2012) Solve a New Keynesian model with ZLB model 8 state variables without local perturbations techniques.
- JMM(2012) solve models with HUNDREDS of state variables.

SECTION: Numerical techniques for HD-DEM

- Approximation of integrals.
- Derivative-free optimization methods.

Approximation of integrals

Class of methods that approximate multidimensional integrals by a weighted sum of integrants:

$$\mathbb{E}(G(\epsilon)) = \int_{\mathbb{R}^n} G(\epsilon) w(\epsilon) d\epsilon \approx \sum_{j=1}^{J} w_j G(\epsilon_j)$$

Where ϵ is a vector of uncorrelated variables, $w(\epsilon)$ is a density function, $\{\epsilon_j\}_{j=1,\ldots,J}$, is a set of nodes and $\{w_j\}_{j=1,\ldots,J}$ a set of weights st $\sum^J w_j = 1$.
We restrict attention to the case of random variables (ϵ) that are normally distributed with zero mean and unit standard deviation.

Approximation of integrals

- **Avoid Montecarlo** integration method. Low degree of convergence.
- **Gauss-Hermite quadrature**: For one exogenous variable or small number of normally distributed exogenous random variables useful. When the the dimension of exogenous variables is large it is too costly. Number nodes grows exponentially on exogenous variables.
- **Monomial Integration** rules are nonproduct: they construct a relatively small set of nodes distributed in some way within a multidimensional hypercube. The computational expense of monomial rules grows only polynomially with d.
- **Tauchen (1986) method):** To convert an AR(1) process of a continuous variable to a discrete Markov chain.

Gauss-Hermite Product Quadrature Rules

GH quadrature method provides a set of nodes $\{\epsilon_j\}$ and weights $\{_j\}$ for approximating unidimensional version of the integral.
We can extend to the multidimensional quadrature by way of a **tensor-product rule**:

$$\mathbb{E}(G(\epsilon)) = \int_{\mathbb{R}^N} G(\epsilon) w(\epsilon) d\epsilon \approx \sum_{j_1=1}^{J_1} \cdots \sum_{j_N=1}^{J_N} w_{j_1} \ldots w_{j_N} G(\epsilon_{j_1}, \ldots, \epsilon_{j_N})$$

To find the set of GH-quadrature nodes and weights we can use the Casio online calculator among other sources.

Monomial Integration

Monomial rules are introduced to economic literature in Judd (1998). Monomial formulas are used for approximating integrals by all global methods focusing on large-scale applications, e.g., Judd et al. (2010, 2011a,b, 2012), Malin et al. (2011), Maliar et al. (2011). Here we focus on Monomial rule "m1" and "m2". Many othe rmonomial rules are available in the mathematical literature: Stroud (1971) and Judd (1998).

Monomial rule M1, 2N

M1 constructs nodes by considering deviations of each random variable holding the other random variables fixed to their expected values. Number of nodes is 2N.

$$\mathbb{E}(G(\epsilon)) \approx \frac{1}{2N} \sum_{h} = 1^{N} G(R\iota^{h}) + G(-R\iota^{h})$$

where $\epsilon \sim N(0_N, I_N)$ and $R := \sqrt{N}, and \iota^h \in \mathbb{R}^N$ is a vector whose hth element is equal to one and the remaining elements are equal to zero, i.e., $\iota_h(0, ..., 1, ..., 0)$.

- Code provided in: monomialintegration.py

Monomial rule M1

Example 8. Let $\epsilon^h \sim \mathcal{N}(0,1)$, $h = 1,2$ be uncorrelated normally distributed variables. A monomial rule $M1$ has four nodes; see Figure 7c

value\node	$j=1$	$j=2$	$j=3$	$j=4$
ϵ_j^1	$\sqrt{2}$	$-\sqrt{2}$	0	0
ϵ_j^2	0	0	$\sqrt{2}$	$-\sqrt{2}$
ω_j	1/4	1/4	1/4	1/4

The expectation of a function $G(\epsilon)$ is approximated as

$$E[G(\epsilon)] = \frac{1}{4}\left[G\left(\sqrt{2},0\right) + G\left(-\sqrt{2},0\right) + G\left(0,\sqrt{2}\right) + G\left(0,-\sqrt{2}\right)\right].$$

Monomial rule M2

A M2 monomial rule has $2N^2 + 1$ nodes and is defined by the following expectation:

$$\mathbb{E}(G(\epsilon)) \approx \frac{2}{2+N} G(0,...,0) + \frac{4-N}{2(2+N^2)} \sum_{h=1}^{N} \{G(R\iota^h) + G(-R\iota^h)\}$$

$$+ \frac{1}{(N+2)^2} \sum_{h=1}^{N-1} \sum_{s=h+1}^{N} G(\pm D\iota^h \pm D\iota^s)$$

where $\epsilon \sim N(0_N, I_N)$ and $R := \sqrt{2+N}$, $D := \sqrt{\frac{2+N}{2}}$.

Monomial rule M2

Example 9. Let $\epsilon^h \sim \mathcal{N}(0,1)$, $h = 1, 2$ be uncorrelated normally distributed variables. A monomial rule $M1$ has nine nodes; see Figure 7d.

value\node	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$
ϵ_j^1	0	2	-2	0	0	$\sqrt{2}$	$\sqrt{2}$	$-\sqrt{2}$	$-\sqrt{2}$
ϵ_j^2	0	0	0	2	-2	$\sqrt{2}$	$-\sqrt{2}$	$\sqrt{2}$	$-\sqrt{2}$
ω_j	$\frac{1}{2}$	1/16	1/16	1/16	1/16	1/16	1/16	1/16	1/16

The expectation of a function $G(\epsilon)$ can be approximated as

$$E[G(\epsilon)] \approx \frac{1}{2}G(0,0) + \frac{1}{16}[G(2,0) + G(-2,0) + G(0,2) + G(0,-2)] + \frac{1}{16}\left[G\left(\sqrt{2},\sqrt{2}\right) + G\left(\sqrt{2},-\sqrt{2}\right) + G\left(-\sqrt{2},\sqrt{2}\right) + G\left(-\sqrt{2},\sqrt{2}\right)\right].$$

Case of correlated exogenous variables

The results can be generalized for the case of correlated variables using a Cholesky decomposition. Let $\epsilon_N \sim \mathcal{N}(\mu_N, \Sigma_{NN})$ The Cholesky decomposition of Σ (i.e: $\Sigma = \Omega\Omega'$) where Ω is a lower triangular matrix with strictly positive diagonal entries. This decomposition allows us to transform correlated variables ϵ into uncorrelated υ with the linear change of variables:

$$\upsilon = \Omega^{-1}(\epsilon - \mu)$$

Case of correlated exogenous variables

Then we can approximate the integral as:

$$\int_{\mathbb{R}^N} G(\epsilon) w(\epsilon) d\epsilon \approx \sum_{j=1}^{J} w_j G(\Omega \upsilon_j + \mu) := \sum_{j=1}^{J} w_j G(\epsilon)$$

Note that we can use the same formula as with uncorrelated normalized shocks with the only modification that instead of a node implied by a given integration rule we use a transformed node $\epsilon_j := \Omega \upsilon_j + \mu$.

Derivative-free Optimization Methods

Here we focus on solving systems of nonlinear equations that arise in the context of the Euler equation methods. The tool proposed is a derivative-free fixed-point iteration (FPI) method.

(FPI): Fixed-point iteration.

Initialization. Fix initial guess $x^{(0)}$, a norm $\|\cdot\|$, and a convergence criterion ϖ.
Step 1. On iteration i, compute $\widehat{x} = F\left(x^{(i)}\right)$.
Step 2. If $\left\|\widehat{x} - x^{(i)}\right\| < \varpi$, then stop.
Otherwise, set $x^{(i+1)} = \widehat{x}$ and go to Step 1.

Ga

and Judd (1997) pointed out that fixed-point iteration is a cheap alternative to time iteration in high-dimensional applications. Judd et al. (2010, 2011, 2012) show a variant of fixed-point iteration, which performs particularly well in the context of the studied models.

Derivative-free Optimization Methods: Fixed-Point Iteration

- FPI 2 advantages over Newton-style methods:
  1. FPI does not require us to compute derivatives (such as Jacobian or Hessian); as a result, its cost does not increase considerably with the dimensionality of the problem.
  2. FPI can iterate on any object x at once (variable, vector of variables, matrix, time series, vector of coefficients, etc.), while Newton-style methods compute a solution point-by-point and are more difficult to vectorize or parallelize.
- FPI has a main drawback:FPI method is that it may fail to converge.
- damping can often help us to achieve the convergence. Instead of a full updating at the end of the iteration $x(i+1) = x$, (Maliar& Maliar, 2014) recommend to use a partial updating:

$$x(i+1) = \xi\hat{x} + (1-\xi)x(i)$$

for some $\xi \in (0, 1]$ and replacing the convergence criterion $\bar{\omega} = \frac{\bar{\omega}}{\xi}$

Example: Fixed-Point Iteration vs Time Iteration

Consider we want to solve the representative agent neoclassical stochastic Model of Growth:

$$\underset{\{c_t, k_{t+1}\}_{t=0}^{\infty}}{\text{Max}} \quad \mathbb{E}_{\not\vdash} \beta^t u(c_t)$$

st :

$$c_t + k_{t+1} = (1 - \delta)k_t + \theta_t f(k_t)$$

$$ln\theta_t = \rho ln\theta_{t-1} + \sigma\varepsilon_t$$

,

$$\varepsilon_t \sim N(0, \sigma^2)$$

Time iteration

Time iteration solves a system of three equilibrium conditions with respect to $\hat{k}'$ in each point of the grid (tensor-product, simulated, smolyak). It parameterizes the capital function by a polynomial $\hat{k}' = K(k, \theta; b)$, where b is the coefficient vector. It iterates on b by solving for current capital $\hat{k}'$ given the interpolant for future capital $K(\hat{k}'; \theta', b)b$

$$
\begin{aligned}
u'(c) &= \beta E\left[u'\left(c'_j\right)\left(1-\delta+\theta'_j\ f'\left(\widehat{k}'\right)\right)\right], \\
c &= (1-\delta)\,k+\theta f\,(k)-\widehat{k}', \\
c'_j &= (1-\delta)\,\widehat{k}'+\theta'_j f\left(\widehat{k}'\right)-K\left(\widehat{k}', \theta'_j; b\right) b.
\end{aligned}
$$

Fixed-point Iteration

1. Parameterize the capital function $\hat{k}' = K(k, \theta; b)$.
2. Rewrite the Euler equation in a way, which is convenient for implementing a fixed-point iteration:

$$k' = \frac{u(c')}{u(c)}(1 + \theta' f'(k') - \delta)k'$$

3. substitute $k' = \hat{k}' = K(k, \theta; b)$ in RHS, we get a different value in LHS. Perform iterations until the two sides coincide.
4. Represent the system of equations as follows:

$$\begin{aligned} k' &= K(k, \theta; b) \text{ and } k_j'' = K(k', \theta'; b); \\ c &= (1-\delta)k + \theta f(k) - k'; \\ c' &= (1-\delta)k' + \theta' f(k') - k''; \\ \widehat{k'} &= \beta E\left[\frac{u'(c')}{u'(c)}(1 - \delta + \theta' f'(k'))k'\right]. \end{aligned}$$

5. In each iteration, given b, compute c, c', k', k'' into last equation. Get a new $\hat{k}'$ and continue iterating till convergence achieved.

SECTION: Dynammic Programming Methods for HD

- Endogenous Grid Method
- Envelope Condition Method

Endogenous Grid Method

Endogenous Grid Method (Carroll, 2005):

- Loop on k', endogenous grid, instead of k. Given Euler equation + BC we can recover k for any k' without needing any solver!
- given k interpolate to obtain k'.
- This second step is problematic under multiple-dimensional interpolation. Note that k won't be ordered and equally spaced implying interpolation problems with standard methods. See Druedhal, Jørgensen (2012) for a discussion of the problem and a proposed solution.

Envelope Condition Method (ECM)

- The ECM (Maliar and Maliar, 2013) simplifies rootfinding using a different mechanism. First, **ECM does not perform conventional backward iteration** on the Bellman equation **but iterates forward**. Second, **to construct policy functions, ECM uses the envelope condition instead of the FOCs**.The systems of equations produced by ECM are typically easier to solve than those produced by conventional VFI.
- As EGM, ECM avoids using a solver per each iteration.
- Provides more accuracy than VFI since the object that is relevant for accuracy is not $V(S)$ but $V_a(S)$ since it is the one that identifies model's variables.

SECTION: Other useful techniques

- Precomputational techniques.
- Parallel computation

Precomputational techniques

- Precomputation of integrals: Precomputation of expectations for polynomial functions. Precomputation of the expectations in the Euler equation. Precomputation of the expectation in the Bellman equation.
- Precomputation of intratemporal choice manifold.
- Precomputation of aggregate decision rules.

Parallel Computation

- Using more than one computer at the same time. Solve different routines at the same time. Your code need to be able to work separately. Separate inner and outer loops properly.
- Free-programs (Python, Julia, etc) have more access to PC than non-free ones (Matlab).
- Amazon offer supercomputers to solve your code and it isn't that expensive.

Solving the neoclasical model of growth

Consider we want to solve the representative agent neoclassical stochastic Model of Growth:

$$\underset{\{c_t, k_{t+1}\}_{t=0}^{\infty}}{\text{Max}} \; \mathbb{E}_{\not\vdash} \beta^t u(c_t)$$

st :

$$c_t + k_{t+1} = (1-\delta)k_t + \theta_t f(k_t)$$

$$ln\theta_t = \rho ln\theta_{t-1} + \sigma\varepsilon_t$$

,

$$\varepsilon_t \sim N(0, \sigma^2)$$

Smolyak's Algorithm as in JMMV(2014)

Smolyak-based projection method

Initialization.

a. Choose the approximation level, μ.

b. Construct the Smolyak grid $\mathcal{H}^{2,\mu} = \{(x_n, y_n)\}_{n=1,...,M}$ on $[-1, 1]^2$.

c. Compute the Smolyak basis functions $\mathcal{P}^{2,\mu}$ in each grid point n. The resulting $M \times M$ matrix is $\mathcal{B}$.

d. Fix $\Phi : (k, \theta) \to (x, y)$, where $(k, \theta) \in \mathbb{R}^2_+$ and $(x, y) \in [-1, 1]^2$. Use Φ^{-1} to compute (k_n, θ_n) that corresponds to (x_n, y_n) in $\mathcal{H}^{2,\mu}$.

e. Choose integration nodes, ϵ_j, and weights, ω_j, $j = 1, ..., J$.

f. Construct future productivities, $\theta'_{n,j} = \theta_n^\rho \exp(\epsilon_j)$ for all j;

g. Choose an initial guess $b^{(1)}$.

Smolyak's Algorithm as in JMMV(2014)

Step 1. *Computation of a solution for* K.

a. At iteration i, for $n = 1, ..., M$, compute

– $k'_n = \mathcal{B}_n b^{(i)}$, where $\mathcal{B}_n$ is the nth row of $\mathcal{B}$.

– $\left(x'_n, y'_{n,j}\right)$ that corresponds to $\left(k'_n, \theta'_{n,j}\right)$ using Φ.

– Compute the Smolyak basis functions in each point $\left(x'_n, y'_{n,j}\right)$

– The resulting $M \times M \times J$ matrix is $\mathcal{B}'$.

– $k''_{n,j} = \mathcal{B}'_{n,j} b^{(i)}$, where $\mathcal{B}'_{n,j}$ is the nth row of $\mathcal{B}'$ in state j.

– $c_n = (1-\delta)\, k_n + \theta_n f\left(k_n\right) - k'_n$;

– $c'_{n,j} = (1-\delta)\, k'_n + \theta_n^{\rho} \exp\left(\epsilon_j\right) f\left(k'_n\right) - k''_{n,j}$ for all j;

– $\widehat{k}'_n \equiv \beta \sum_{j=1}^{J} \omega_j \cdot \left[\frac{u'\left(c'_{n,j}\right)}{u'\left(c_n\right)} \left[1 - \delta + \theta_n^{\rho} \exp\left(\epsilon_j\right) f'\left(k'_n\right)\right] k'_n\right]$.

b. Find b that solves the system in Step 1a.

– Compute $\widehat{b}$ that solves $\widehat{k}' = \mathcal{B}\widehat{b}$, i.e., $\widehat{b} = \mathcal{B}^{-1}\widehat{k}'_n$.

– Use damping to compute $b^{(i+1)} = (1-\xi)\, b^{(i)} + \xi \widehat{b}$, where $\xi \in (0, 1]$.

– Check for convergence: end Step 1 if $\frac{1}{M\xi} \sum_{n=1}^{M} \left| \frac{(k'_n)^{(i+1)} - (k'_n)^{(i)}}{(k'_n)^{(i)}} \right| < 10^{-\vartheta}$, $\vartheta > 0$.

Iterate on Step 1 until convergence.

Summary