Advances in Numerical Dynamic Programming and New Applications

Yongyang Cai, Hoover Institution

(Co-author: Kenneth L. Judd)

June 27, 2013

Outline

- Introduction of Numerical Dynamic Programming
- Advances in Numerical Dynamic Programming
  - Shape-preserving Approximation
  - Hermite Approximation
  - Parallelization
- Applications
  - Dynamic Portfolio Optimization
  - Dynamic and Stochastic Integration of Climate and Economy

Introduction of Dynamic Programming

- Finite horizon and/or non-stationary dynamic programming problems
- Value function:

$$V_t(x_t, \theta_t) = \max_{a_s \in \mathcal{D}(x_s, \theta_s, s)} \sum_{s=t}^{T-1} \beta^{s-t} \mathbb{E}\left\{u_s(x_s, a_s, \theta_s)\right\} + \beta^{T-t} \mathbb{E}\left\{V_T(x_T, \theta_T)\right\}$$

- Bellman equation:

$$\begin{aligned} V_t(x, \theta) = \max_{a \in \mathcal{D}(x, \theta, t)} \quad & u_t(x, a) + \beta \mathbb{E}\left\{V_{t+1}(x^+, \theta^+) \mid x, \theta, a\right\}, \\ \text{s.t.} \quad & x^+ = g_t(x, \theta, a, \omega), \\ & \theta^+ = h_t(\theta, \epsilon), \end{aligned}$$

Three Numerical Parts in DP

- Approximation of Value Functions
  - (Multidimensional) Chebyshev polynomails
- Numerical Integration
  - Gauss-Hermite quadrature
- Optimization
  - NPSOL

Typical Application I

- Optimal growth problem:

$$
\begin{aligned}
V_0(k_0) = \max_{c,l} \quad & \sum_{t=0}^{T-1} \beta^t u(c_t, l_t) + \beta^T V_T(k_T), \\
\text{s.t.} \quad & k_{t+1} = F(k_t, l_t) - c_t, \quad 0 \le t < T, \\
& \underline{k} \le k_t \le \bar{k}, \quad 1 \le t \le T, \\
& c_t,\ l_t \ge \epsilon, \quad 0 \le t < T,
\end{aligned}
$$

- Bellman equation:

$$
\begin{aligned}
V_t(k) \quad = \quad \max_{c,l} \quad & u(c, l) + \beta V_{t+1}(k^+), \\
\text{s.t.} \quad & k^+ = F(k, l) - c, \\
& \underline{k} \le k^+ \le \bar{k}, \quad c, l \ge \epsilon,
\end{aligned}
$$

Typical Application II

- Multi-stage portfolio optimization problem:

$$V_0(W_0) = \max_{S_t, 0 \le t < T} \mathbb{E}\{u(W_T)\},$$

- Wealth transition:

$$W_{t+1} = R_f(W_t - e^\top S_t) + R^\top S_t,$$

- Bellman equation:

$$\begin{aligned} V_t(W) &= \max_{B,S} \mathbb{E}\{V_{t+1}(R_f B + R^\top S)\}, \\ &\quad \text{s.t.} \quad B + e^\top S = W, \end{aligned}$$

Typical Application III

Multi-country optimal growth problem:

$$
\begin{aligned}
V_0(k_0, \theta_0) \quad = \quad & \max_{k_t, I_t, c_t, l_t} \; \mathbb{E}\left\{ \sum_{t=0}^{T-1} \beta^t u(c_t, l_t) + \beta^T V_T(k_T, \theta_T) \right\}, \\
\text{s.t.} \quad & k_{t+1,j} = (1-\delta)k_{t,j} + I_{t,j}, \quad j = 1, \ldots, d, \\
& \Gamma_{t,j} = \frac{\zeta}{2} k_{t,j} \left(\frac{I_{t,j}}{k_{t,j}} - \delta \right)^2, \quad j = 1, \ldots, d, \\
& \sum_{j=1}^{d} (c_{t,j} + I_{t,j} - \delta k_{t,j}) = \sum_{j=1}^{d} \left(f(k_{t,j}, l_{t,j}, \theta_t) - \Gamma_{t,j} \right), \\
& \theta_{t+1} = g(\theta_t, \epsilon_t).
\end{aligned}
$$

Numerical Dynamic Programming

Value function iteration method for solving finite-horizon and/or non-stationary dynamic programming problems.

- Initialization. Choose the approximation grid, $X = \{x_i : 1 \leq i \leq m\}$, and choose functional form for $\hat{V}(x; b)$. Let $\hat{V}(x; b^T) = V_T(x)$. Iterate through steps1 and 2 over $t = T-1, ..., 1, 0$.
- Step 1. Maximization step: Compute

$$v_i = \max_{a_i \in \mathcal{D}(x_i, t)} u_t(x_i, a_i) + \beta \mathbb{E}\{\hat{V}(x_i^+; \mathbf{b}^{t+1})\},$$

  for each $x_i \in X$, $1 \leq i \leq m$.
- Step 2. Fitting step: Using the appropriate approximation method, compute the b^t such that $\hat{V}(x; b^t)$ approximates (x_i, v_i) data.

Computational Challenges

- Smooth function approximation is important for high-dimensional problems:
  - It can avoid the curse of dimensionality
  - Fast Newton-type optimiation solvers can be applied
- Monotonicity and concavity of value functions may be NOT preserved by smooth function approximation
  - Difficult for optimization solvers to find global maximizers
- High-dimensional problems requires many approximation nodes
  - Efficient usage of all possible information (such as slopes of value functions) can improve much
  - Parallelization can also be very efficient

Approximation

- Chebyshev polynomial approximation

$$\hat{V}(x;\mathbf{b}) = \sum_{j=0}^{n} b_j \mathcal{T}_j(Z(x)),$$

  - Chebyshev polynomial basis: $\mathcal{T}_j(z) = \cos(j\cos^{-1}(z))$
  - Normalization: $Z(x) = \frac{2x - x_{\mathbf{min}} - x_{\mathbf{max}}}{x_{\mathbf{max}} - x_{\mathbf{min}}}$
- Multidimensional Chebyshev polynomail approximation
  - Complete polynomial approximation:

$$\hat{V}_n(x;\mathbf{b}) = \sum_{0 \le |\alpha| \le n} b_\alpha \mathcal{T}_\alpha(Z(x)),$$

  - $\mathcal{T}_\alpha(z)$ denote the product $\mathcal{T}_{\alpha_1}(z_1) \cdots \mathcal{T}_{\alpha_d}(z_d)$

Shape-preserving Chebyshev Interpolation

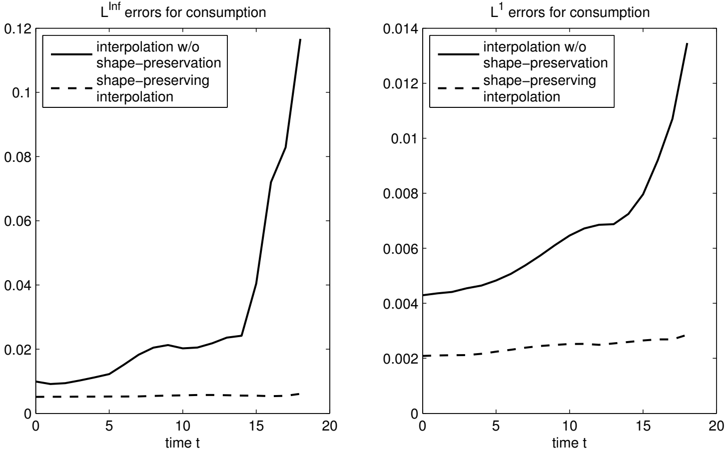
- LP problem to find coefficients

$$
\begin{aligned}
\min_{b_j, b_j^+, b_j^-} \quad & \sum_{j=0}^{m-1} (b_j^+ + b_j^-) + \sum_{j=m}^{n} (j+1-m)^2 (b_j^+ + b_j^-), \\
\text{s.t.} \quad & \sum_{j=0}^{n} b_j \mathcal{T}_j'(y_{i'}) > 0, \qquad i' = 1, \ldots, m', \\
& \sum_{j=0}^{n} b_j \mathcal{T}_j''(y_{i'}) < 0, \qquad i' = 1, \ldots, m', \\
& \sum_{j=0}^{n} b_j \mathcal{T}_j(z_i) = v_i, \quad i = 1, \ldots, m, \\
& b_j - \hat{b}_j = b_j^+ - b_j^-, \quad j = 0, \ldots, m-1, \\
& b_j = b_j^+ - b_j^-, \quad j = m, \ldots, n, \\
& b_j^+, \; b_j^- \geq 0, \quad j = 1, \ldots, n,
\end{aligned}
$$

- y: shape nodes; z: approximation nodes

Application in Example I

Figure: Errors of numerical dynamic programming with Chebyshev interpolation with/without shape-preservation for growth problems



Hermite Value Function Iteration

- Envelope Theorem: If

$$
\begin{aligned}
H(x) &= \max_a \; f(x,a) \\
& \quad \text{s.t.} \;\; g(x,a) = 0, \\
& \qquad\quad\;\; h(x,a) \geq 0,
\end{aligned}
$$

then

$$
\frac{\partial H(x)}{\partial x_j} = \frac{\partial f}{\partial x_j}(x, a^*(x)) + \lambda^*(x)^\top \frac{\partial g}{\partial x_j}(x, a^*(x)) + \mu^*(x)^\top \frac{\partial h}{\partial x_j}(x, a^*(x))
$$

Get Slopes Easily

- Equivalent formulation:

$$\begin{aligned} H(x) \quad = \quad & \max_{a,y} \; f(y,a) \\ & \text{s.t.} \;\; g(y,a) = 0, \\ & \qquad h(y,a) \geq 0, \\ & \qquad x_j - y_j = 0, \quad j = 1, \ldots, d, \end{aligned}$$

- Get slope of H easily:

$$\frac{\partial H(x)}{\partial x_j} = \tau_j^*(x),$$

  - $\tau_j^*(x)$: *the shadow price of the trivial constraint* $x_j - y_j = 0$

Multidimensional Hermite Approximation

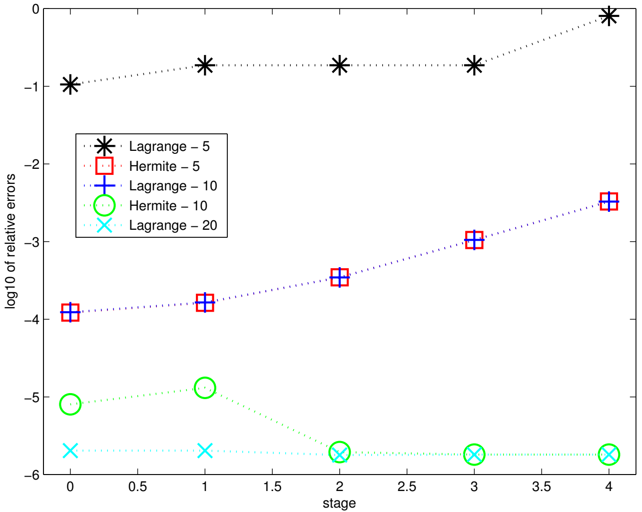
- Least-square problem

$$\min_{\mathbf{b}} \quad \sum_{i=1}^{N} \left(v_i - \sum_{0 \le |\alpha| \le n} b_\alpha \mathcal{T}_\alpha \left(x^i \right) \right)^2 + \sum_{i=1}^{N} \sum_{j=1}^{d} \left(s_j^i - \sum_{0 \le |\alpha| \le n} b_\alpha \frac{\partial}{\partial x_j} \mathcal{T}_\alpha \left(x^i \right) \right)^2$$

- Hermite data $\{(x^i, v_i, s^i) : i = 1, \ldots, N\}$:
  - $v_i = V(x^i)$,
  - $s_j^i = \frac{\partial}{\partial x_j} V(x^i)$

Application in Example II

Figure: Errors of H-VFI or L-VFI for Dynamic Portfolio Optimization



Accuracy and Running Times

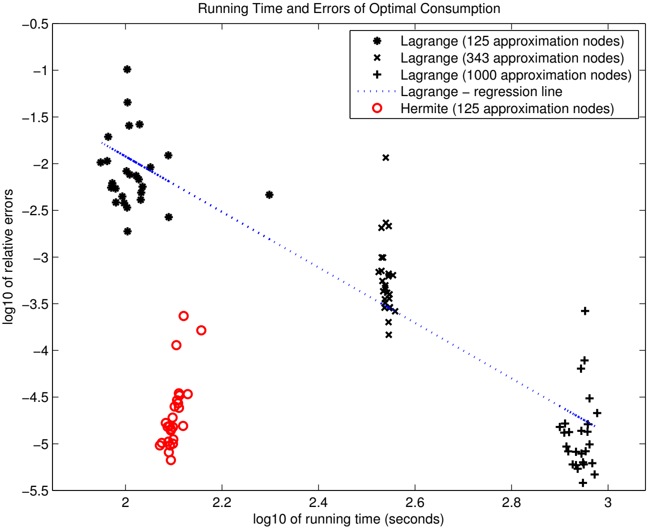
Table: Relative Errors and Running Times of L-VFI or H-VFI for Dynamic Portfolio Optimization

m	L-VFI error	H-VFI error	L-VFI time	H-VFI time
5	0.8	0.00327	9 seconds	10 seconds
10	0.00328	1.3×10^{-5}	12 seconds	17 seconds
20	2.0×10^{-6}		33 seconds	

- To reach the same accuracy of H-VFI, for one-dimensioanl problems, L-VFI needs
  - twice as many nodes
  - twice as much time

Application in Example III (Three Countries)

Figure: L-VFI vs H-VFI for Three-Country Optimal Growth Problems



Application in Example III (Six Countries)

Table: H-VFI vs L-VFI for Six-Dimensional Stochastic Problems

	error of c_0^*		error of l_0^*		time (hour)	
m	L-VFI	H-VFI	L-VFI	H-VFI	L-VFI	H-VFI
3	$3.8(-2)$	$3.6(-3)$	$5.4(-2)$	$5.2(-3)$	0.3	0.67
5	$5.5(-3)$		$8.2(-3)$		8.74	
6	$3.1(-3)$		$4.5(-3)$		36.6	

Note: $a(k)$ means $a \times 10^k$.

- To reach the same accuracy of H-VFI, for six-dimensioanl problems, L-VFI needs
  - 64 times as many nodes (6^6 nodes vs 3^6 nodes)
  - 55 times as much time (36.6 hours vs 0.67 hours)

Parallelization in Dynamic Programming

- Parallelization in Maximization step in NDP: Compute

$$v_i = \max_{a_i \in \mathcal{D}(x_i, t)} \; u_t(x_i, a_i) + \beta \mathbb{E}\{\hat{V}(x_i^+; \mathbf{b}^{t+1})\},$$

for each $x_i \in X_t$, $1 \leq i \leq m_t$.

- Master-Worker system: Master processor, Worker processors.
  - Workers solve the independent maximzation problems
  - Master distributes tasks, collects results, does the fitting step

Parallelization Results for Example III

- Multi-country optimal growth problem:

$$
\begin{aligned}
V_t(k,\theta) \quad = \quad & \max_{c,l,I} \quad u(c,l) + \beta \mathbb{E}\left\{V_{t+1}(k^+,\theta^+) \mid \theta\right\}, \\
& \text{s.t.} \quad k_j^+ = (1-\delta)k_j + I_j + \epsilon_j, \quad j = 1,\ldots,d, \\
& \qquad \Gamma_j = \frac{\zeta}{2} k_j \left(\frac{I_j}{k_j} - \delta\right)^2, \quad j = 1,\ldots,d, \\
& \qquad \sum_{j=1}^{d} (c_j + I_j - \delta k_j) = \sum_{j=1}^{d} \left(f(k_j, l_j, \theta_j) - \Gamma_j\right), \\
& \qquad \theta^+ = g(\theta, \xi_t),
\end{aligned}
$$

- Four-dimensional k (continuous)
- Four-dimensional θ (discrete with 7 values per country)
- Four-dimensional ϵ (discrete with 3 values per country)

Results for Example III

- 2401 tasks per value function iteration
- 2401 optimization problems per task

Table: Statistics of parallel dynamic programming under HTCondor-MW for the growth problem

Wall clock time for three VFIs	8.28 hours
Total time workers were assigned	16.9 days
Average wall clock time per task	199 seconds
Number of (different) workers	50
Overall Parallel Performance	98.6%

Parallel Efficiency for Example III

Table: Parallel efficiency for various number of worker processors

# Worker processors	Parallel efficiency	Average task wall clock time (seconds)	Total wall clock time (hours)
50	98.6%	199	8.28
100	97%	185	3.89
200	91.8%	186	2.26

PART II:

NEW APPLICATIONS

Dynamic Portfolio Optimization

- n stocks and 1 bond, T periods
- $R = (R_1, \ldots, R_n)^\top$: random return vector of stocks
- R_f: riskless return of bond
- Dynamic Portfolio Problem:

$$V_0(W_0) = \max_{x_t, 0 \le t < T} \mathbb{E}[u(W_T)]$$

- $x_t = (x_{t1}, \ldots, x_{tn})^\top$: fractions of wealth invested in the stocks
- W_t: wealth. When $\tau = 0$:

$$W_{t+1} = W_t(R_f(1 - e^\top x_t) + R^\top x_t),$$

Portfolio with Transaction Costs

- Multi-stage Portfolio Optimization Problem:

$$
\begin{aligned}
V_0(W_0, x_0) = \max_{\delta_t} \quad & \mathbb{E}\{u(W_T)\} \\
\text{s.t.} \quad & W_{t+1} = \mathbf{e}^\top X_{t+1} + R_f(1 - \mathbf{e}^\top x_t - y_t)W_t), \\
& X_{t+1,i} = R_i(x_{t,i} + \delta_{t,i})W_t, \\
& y_t = \mathbf{e}^\top(\delta_t + \tau|\delta_t|), \\
& x_{t+1,i} = X_{t+1,i}/W_{t+1}, \\
& t = 0, \ldots, T-1; \quad i = 1, \ldots, k,
\end{aligned}
$$

- τ: proportional transaction costs
- $\delta_{t,i} > 0$ means buying, and $\delta_{t,i} < 0$ means selling

Bellman Equation

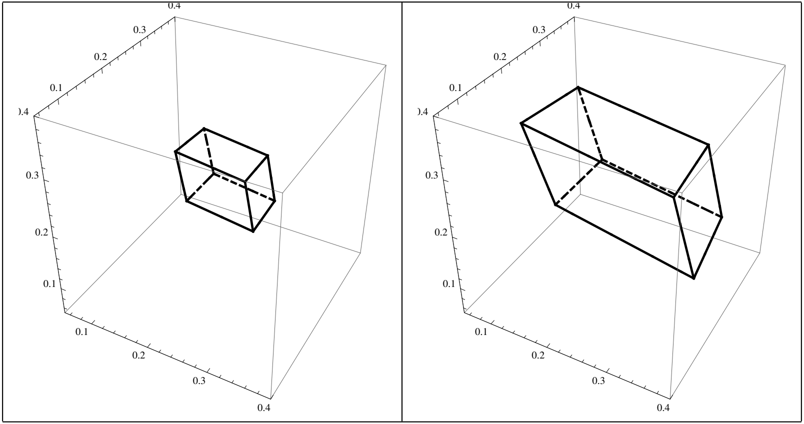
- Bellman equation

$$V_t(W_t, x_t) = \max_{\delta_t} \ \mathbb{E}\left\{V_{t+1}(W_{t+1}, x_{t+1})\right\},$$

where

$$\begin{aligned}
y_t &\equiv \mathbf{e}^\top(\delta_t + \tau|\delta_t|), \\
X_{t+1,i} &\equiv R_i(x_{t,i} + \delta_{t,i})W_t, \\
W_{t+1} &\equiv \mathbf{e}^\top X_{t+1} + R_f(1 - \mathbf{e}^\top x_t - y_t)W_t, \\
x_{t+1,i} &\equiv X_{t+1,i}/W_{t+1},
\end{aligned}$$

No-trade regions



$t = 0$	$t = 5$

Parallelization of Seven-Asset Portfolio Problems

- Number of Value Function Iterations: 6
- Number of optimization problems in one VFI: 15625
- Number of quadrature points for the integration in the objective function for one optimization problem: 15625

	Num of Jobs in one VFI	Wall Clock Time	Total CPU Time	Parallel Efficiency
96 cores	625	1.27 hours	4.7 days	92.3%
480 cores	3125	16 minutes	4.9 days	92%
Condor MW 100 workers	3125	1.3 hours	4.7 days	89%

New Application II: Climate Change Analysis

Question: What can and should be the response to rising CO_2 concentrations?

- Analytical tools in the literature: IAMs (Integrated Assessment Models)
  - Two components: economic model and climate model
  - Interaction is often limited: Economy emits CO_2 which affects world average temperature which affects economic productivity.
- Existing IAMs cannot study dynamic decision-making in an evolving and uncertain world
  - Most are deterministic; economic actors know with certainty the consequences of their actions and the alternatives
  - Most are myopic; standard reason is computational feasibility

Nordhaus' DICE: The Prototypical Model

- DICE2007 was the only dynamic economic model used by the US Interagency Working Group on the Cost of Carbon
- Economic system
  - gross output: $Y_t \equiv f(k_t, t) = A_t k_t^{\alpha} l_t^{1-\alpha}$
  - damage factor: $\Omega_t \equiv 1/\left(1 + \pi_1 T_t^{\mathrm{AT}} + \pi_2 (T_t^{\mathrm{AT}})^2\right)$
  - emission control cost: $\Lambda_t \equiv \psi_t^{1-\theta_2} \theta_{1,t} \mu_t^{\theta_2}$, where μ_t is policy choice
  - output net of damages and emission control: $\Omega_t (1 - \Lambda_t) Y_t$
- Climate system
  - Carbon mass: $\mathbf{M}_t = (M_t^{\mathrm{AT}}, M_t^{\mathrm{UP}}, M_t^{\mathrm{LO}})^{\top}$
  - Temperature: $\mathbf{T}_t = (T_t^{\mathrm{AT}}, T_t^{\mathrm{LO}})^{\top}$
  - Carbon emission: $E_t = \sigma_t (1 - \mu_t) Y_t + E_t^{\mathrm{Land}}$
  - Radiative forcing: $F_t = \eta \log_2 \left(\left(M_t^{\mathrm{AT}} + M_{t+1}^{\mathrm{AT}} \right) / \left(2 M_0^{\mathrm{AT}} \right) \right) + F_t^{\mathrm{EX}}$

Uncertainty and Risk

All agree that uncertainty needs to be a central part of any IAM analysis
Multiple forms of uncertainty

- Risk: productivity shocks, taste shocks, uncertain technological advances, weather shocks
- Parameter uncertainty: policymakers do not know parameters that characterize the economic and/or climate systems
- Model uncertainty: policymakers do not know the proper model or the stochastic processes

Abrupt, Stochastic, and Irreversible Climate Change

Question: What is the optimal carbon tax when faced with abrupt and irreversible climate change?

- Common assumption in IAMs: damages depend only on contemporaneous temperature
- Our criticism: this cannot analyze the permanent and irreversible damages from tipping points
- We show that
  - Abrupt climate change can be modeled stochastically
  - The policy response to the threat of tipping points is very different from the policy response to standard damage representations.

Tipping point

- A tipping point is where temperature causes a big event with permanent damage
- The time of tipping is a Poisson process, and probability of a tipping point occurring at t equals the hazard rate $h_t(T_t^{\mathrm{AT}})$
- Examples:
  - Thermohaline circulation collapse
  - Extreme catastrophy (Weitzman (2009)): small probability (hazard rate is 0.1% at 2100) but big deduction of production (20% damage)

Cai-Judd-Lontzek DSICE Model

DSICE (**D**ynamic **S**tochastic **I**ntegrated Model of **C**limate and **E**conomy)

$$\begin{aligned} DSICE &= DICE2007 \\ &+ \text{stochastic damage factor} \\ &+ \text{stochastic production function} \\ &+ \text{flexible period length} \end{aligned}$$

DSICE: new features

- Economic system: $Y_t \equiv f(k_t, \zeta_t, t) = \zeta_t A_t k_t^{\alpha} l_t^{1-\alpha}$ where $\zeta_{t+1} = g^{\zeta}(\zeta_t, \omega_t^{\zeta})$ is an AR(1) process for the productivity state ζ
- Climate system: $\Omega_t \equiv (1 - J_t) / \left(1 + \pi_1 T_t^{AT} + \pi_2 (T_t^{AT})^2\right)$ where $J_{t+1} = g^J(J_t, \omega_t^J)$ is a Markov process for the damage factor state J

DP model of DSICE

- DP model for DSICE

$$
\begin{aligned}
V_t(k, \mathbf{M}, \mathbf{T}, \zeta, J) &= \max_{c,\mu} \quad u_t(c) + \beta \mathbb{E}[V_{t+1}(k^+, \mathbf{M}^+, \mathbf{T}^+, \zeta^+, J^+)] \\
\text{s.t.} \quad k^+ &= (1-\delta)k + \Omega_t(1-\Lambda_t)Y_t - c, \\
\mathbf{M}^+ &= \Phi^M \mathbf{M} + (E_t, 0, 0)^\top, \\
\mathbf{T}^+ &= \Phi^T \mathbf{T} + (\xi_1 F_t, 0)^\top, \\
\zeta^+ &= g^\zeta(\zeta, \omega^\zeta), \\
J^+ &= g^J(J, \omega^J)
\end{aligned}
$$

- One year (or one quarter of a year) time steps over 600 years
- Seven continuous states: $k, \mathbf{M}, \mathbf{T}, \zeta$
- one discrete state: J

Epstein-Zin Preference

- Epstein-Zin preference

$$U_t\left(k,\mathbf{M},\mathbf{T},J\right) = \max_{c,\mu}\left\{(1-\beta)\,u(c_t,l_t) + \beta\left[\mathbb{E}\left\{\left(U_{t+1}\left(k^+,\mathbf{M}^+,\mathbf{T}^+,J^+\right)\right)^{1-\gamma}\right\}\right]^{\frac{1-\psi}{1-\gamma}}\right\}^{\frac{1}{1-\psi}}$$

  - ψ: the inverse of the intertemporal elasticity of substitution
  - γ: the risk aversion parameter

Standardized DP model:

$$V_t(k,\mathbf{M},\mathbf{T},J) = \max_{c,\mu} \quad u(c_t,l_t) + \frac{\beta}{1-\psi}\times \left[\mathbb{E}\left\{\left((1-\psi)\,V_{t+1}\left(k^+,\mathbf{M}^+,\mathbf{T}^+,J^+\right)\right)^{\frac{1-\gamma}{1-\psi}}\right\}\right]^{\frac{1-\psi}{1-\gamma}}$$

Accuracy Test and Running Times

- Relative errors and running times for the deterministic problem for accuracy test

degree	k	M^{AT}	T^{AT}	c	μ	Time
4	6.4(−4)	5.8(−5)	6.1(−5)	1.8(−4)	1.7(−4)	7.8 minutes
6	2.5(−5)	9.4(−7)	1.0(−6)	2.6(−5)	9.5(−6)	2.2 hours

- Running times for various cases of DSICE

	Step Size h	Num of Nodes	Time
One Tipping Point	1 year	31,250	16 minutes
One Economic Shock & Three Tipping Points	1 year	625,000	15.7 hours
Parallel DSICE with One Economic Shock & Three Tipping Points across 112 cores	1 year	625,000	11.75 minutes (total CPU time: 20.9 hours)

Big Increase of Carbon Tax

	ψ	γ	Carbon tax
DICE	2	2	\$37
DSICE, tipping of 2.5% damage	2	10	\$54
DSICE, tipping of 5% damage	2	2	\$69
DSICE, tipping of 5% damage	2	10	\$75
DSICE, tipping of 5% damage	2	20	\$83
DSICE, disaster case	2	10	\$124