Derivative of Value Functions

- Maximization step in the conventional DP algorithm:

$$V_t(x_i) = v_i = \max_{a_i \in \mathcal{D}(x_i,t)} u_t(x_i, a_i) + \beta E\{V_{t+1}(x_i^+) \mid x_i, a_i\},$$

- Conventional fitting step: use the Lagrange data $\{(x_i, v_i) : i = 1, \ldots, m\}$ to construct the approximated value function $\hat{V}_t(x)$.

- Envelope theorem: Let

$$\begin{aligned} V(x) &= \max_y \; f(x,y) \\ & \quad\;\; \text{s.t. } g(x,y) = 0. \end{aligned}$$

Let $y^*(x)$ be the optimizer and $\lambda^*(x)$ be the shadow price.

$$\frac{\partial V}{\partial x} = \frac{\partial f}{\partial x}(x, y^*(x)) + \lambda^*(x)^\top \frac{\partial g}{\partial x}(x, y^*(x)).$$

Optimal Growth Models

- Optimal Growth Problem:

$$V_0(k_0) = \max_{c,l} \sum_{t=0}^{T-1} \beta^t u(c_t, l_t) + \beta^T u_T(k_T), \tag{1}$$

$$\text{s.t.} \quad k_{t+1} = F(k_t, l_t) - c_t, \quad 0 \le t < T,$$

- DP model of optimal growth problem:

$$V_t(k) = \max_{c,l} \; u(c,l) + \beta V_{t+1}(F(k,l) - c),$$

Multi-Stage Portfolio Optimization

- W_t: wealth at stage t; stocks' random return: $R = (R_1, \ldots, R_n)$; bond's riskfree return: R_f;
- $S_t = (S_{t1}, \ldots, S_{tn})^\top$: money in the stocks; $B_t = W_t - e^\top S_t$: money in the bond,
- $W_{t+1} = R_f(W_t - e^\top S_t) + R^\top S_t$
- Multi-Stage Portfolio Optimization Problem:

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

- Bellman Equation:

$$V_t(W) = \max_{S} \; E\{V_{t+1}(R_f(W - e^\top S) + R^\top S)\},$$

W: state variable; S: control variables.

Derivative of Value Functions: Examples

- For the optimal growth DP model,

$$V_t'(k) = \beta V_{t+1}'(F(k,l^*) - c^*)F_k(k,l^*),$$

  where c^* and l^* are the optimal controls for the given k.

- For the multi-stage portfolio optimization problem,

$$V_t'(W) = R_f E\{V_{t+1}'(R_f(W - e^\top S^*) + R^\top S^*)\},$$

  where S^* are the optimal portfolio invested in stocks.

- It seems that we need to compute $V_{t+1}'(F(k,l^*) - c^*)$ or $E\{V_{t+1}'(R_f(W - e^\top X^*) + R^\top X^*)\}$, which are expensive. However, problem (1) has other equivalent forms. We choose a form such that $dV(x)/dx$ can be easily computed.

Derivative of Value Functions in Optimal Growth Models

- For the optimal growth problem,

$$
\begin{aligned}
V_t(k) &= \max_{k^+,c,l} u(c,l) + \beta V_{t+1}(k^+), \\
&\quad \text{s.t. } F(k,l) - c - k^+ = 0,
\end{aligned}
$$

  with k^+, c and l as control variables.

- New formula for computing $V_t'(k)$:

$$
V_t'(k) = \lambda F_k(k,l^*),
$$

  where λ is the shadow price for the constraint $F(k,l) - c - k^+ = 0$, and given directly by optimization packages.

Derivative of Value Functions in Portfolio Optimization

- For the multi-stage portfolio optimization problem,

$$\begin{aligned} V_t(W) &= \max_{B,S} E\{V_{t+1}(R_f B + R^\top S)\}, \\ & \quad \text{s.t. } W - B - e^\top S = 0, \end{aligned} \tag{2}$$

  with the bond allocation B and the stock allocation S.

- New formula for computing $V_t'(W)$:

$$V_t'(W) = \lambda,$$

  where λ is the shadow price for the constraint $W - B - e^\top S = 0$.

Derivative of Value Functions in General Models

- For an optimization problem,

$$\begin{aligned} V(x) &= \max_{y} \; f(x,y) \\ &\quad \text{s.t. } g(x,y)=0, h(x,y)\geq 0, \end{aligned}$$

we can modify it as

$$\begin{aligned} V(x) &= \max_{y,z} \; f(z,y) \\ &\quad \text{s.t. } g(z,y)=0, h(z,y)\geq 0, x-z=0, \end{aligned}$$

by adding a trivial control variable z and a trivial constraint $x-z=0$.

- Then by the envelope theorem, we get

$$V'(x)=\lambda,$$

where λ is the shadow price for the trivial constraint $z=x$.

Numerical DP Algorithm with Hermite Interpolation

Initialization. Choose the approximation nodes, $X_t = \{x_{it} : 1 \leq i \leq m_t\}$ for every $t < T$, and choose a functional form for $\hat{V}(x; \mathbf{b})$. Let $\hat{V}(x; \mathbf{b}^T) \equiv u_T(x)$.

Step 1. Maximization step. For each $x_i \in X_t$, $1 \leq i \leq m_t$, compute

$$\begin{aligned} v_i &= \max_{a_i \in \mathcal{D}(y_i, t), y_i} u_t(y_i, a_i) + \beta E\{\hat{V}(x_i^+; \mathbf{b}^{t+1}) \mid y_i, a_i\}, \\ & \qquad \text{s.t.} \quad x_i - y_i = 0, \end{aligned}$$

and $\quad s_i = \lambda_i$, where λ_i is the shadow price of the constraint $x_i - y_i = 0$.

Step 2. Hermite fitting step. Compute the $\mathbf{b}^t$ such that $\hat{V}(x; \mathbf{b}^t)$ approximates (x_i, v_i, s_i) data. ■

Revised Schumaker Shape-Preserving Interpolation

Step 1. Compute $\delta = (v_2 - v_1)/(x_2 - x_1)$. If $|(s_1+s_2)/2 - \delta| < \varepsilon$, then

$$s(x) = v_1 + s_1(x - x_1) + \frac{(s_2 - s_1)(x - x_1)^2}{2(x_2 - x_1)},$$

and STOP.

Step 2. If $(s_1 - \delta)(s_2 - \delta) \geq 0$, set

$$\xi = (x_1 + x_2)/2, \;\; a = b = \xi - x_1, \;\; \bar{s} = 2\delta - \frac{s_1 + s_2}{2}.$$

Else let

$$\lambda = \frac{s_2 - s_1}{x_2 - x_1}, \;\; a = (s_2 - \delta)/\lambda, \;\; b = (\delta - s_1)/\lambda, \;\; \xi = x_1 + a, \;\; \bar{s} = \delta.$$

Then

$$s(x) = \begin{cases} v_1 + s_1(x - x_1) + C_1(x - x_1)^2, & x \in [x_1, \xi], \\ A_2 + \bar{s}(x - \xi) + C_2(x - \xi)^2, & x \in [\xi, x_2], \end{cases}$$

where $C_1 = (\bar{s} - s_1)/(2a)$, $A_2 = v_1 + a(s_1 + \bar{s})/2$, and $C_2 = (s_2 - \bar{s})/(2b)$. ■

Chebyshev Interpolation with Hermite Information

- If we have Hermite data $\{(x_i, v_i, s_i) : i = 1, \ldots, m\}$ on $[a, b]$, then the following system of $2m$ linear equations produces coefficients for degree $2m-1$ Chebyshev polynomial interpolation on the Hermite data:

$$\sum_{j=0}^{2m-1} c_j T_j(z_i) = v_i, \quad i = 1, \ldots, m,$$

$$\frac{2}{b-a} \sum_{j=0}^{2m-1} c_j T_j'(z_i) = s_i, \quad i = 1, \ldots, m,$$

where $z_i = \frac{2x_i - a - b}{b-a}$ ($i = 1, \ldots, m$) are the Chebyshev nodes in $[-1, 1]$, and $T_j(z)$ are Chebyshev basis polynomials.

Numerical Examples for Optimal Growth Problems

- $\beta = 0.95$; $f(k,l) = Ak^{\alpha}l^{1-\alpha}$ with $\alpha = 0.25$, $A = (1-\beta)/(\alpha\beta)$;

$$u(c,l) = \frac{c^{1-\gamma}}{1-\gamma} - B\frac{l^{1+\eta}}{1+\eta}$$

  with $B = (1-\alpha)A^{1-\gamma}$. $k \in [0.2, 3]$.

- Errors for optimal consumptions at stage 0:

$$\max_{k \in [0.2,k]} \frac{|c^*_{0,DP}(k) - c^*_0(k)|}{1 + |c^*_0(k)|},$$

  where $c^*_{0,DP}$ is the optimal consumption at stage 0 computed by numerical DP algorithms, and c^*_0 is the optimal consumption directly computed by SNOPT in AMPL code on the model (1).

Errors of optimal solutions computed by numerical DP algorithms with Chebyshev interpolation on m Chebyshev nodes using with Lagrange vs. Hermite data

			Lagrange		Hermite	
γ	η	m	error of c_0^*	error of l_0^*	error of c_0^*	error of l_0^*
0.5	0.1	5	2.3(−2)	1.0(−1)	2.0(−3)	9.0(−3)
		10	1.1(−3)	5.0(−3)	6.1(−6)	2.4(−5)
		20	1.4(−5)	2.7(−5)	4.3(−7)	4.5(−6)
0.5	1	5	2.4(−2)	3.1(−2)	2.2(−3)	2.8(−3)
		10	1.3(−3)	1.6(−3)	6.9(−6)	8.1(−6)
		20	1.6(−5)	9.5(−6)	4.1(−7)	2.4(−6)
2	0.1	5	1.1(−2)	1.6(−1)	1.1(−3)	1.9(−2)
		10	5.9(−4)	1.0(−2)	4.2(−6)	6.1(−5)
		20	4.6(−6)	6.7(−5)	4.3(−7)	2.4(−6)
2	1	5	2.0(−2)	7.0(−2)	2.0(−3)	8.7(−3)
		10	9.7(−4)	4.7(−3)	7.8(−6)	3.3(−5)
		20	7.3(−6)	3.0(−5)	1.3(−6)	2.9(−6)
6	1	5	1.4(−2)	1.2(−1)	1.5(−3)	1.7(−2)
		10	6.2(−4)	9.0(−3)	5.4(−6)	7.2(−5)
		20	4.2(−6)	5.5(−5)	4.8(−7)	2.5(−6)

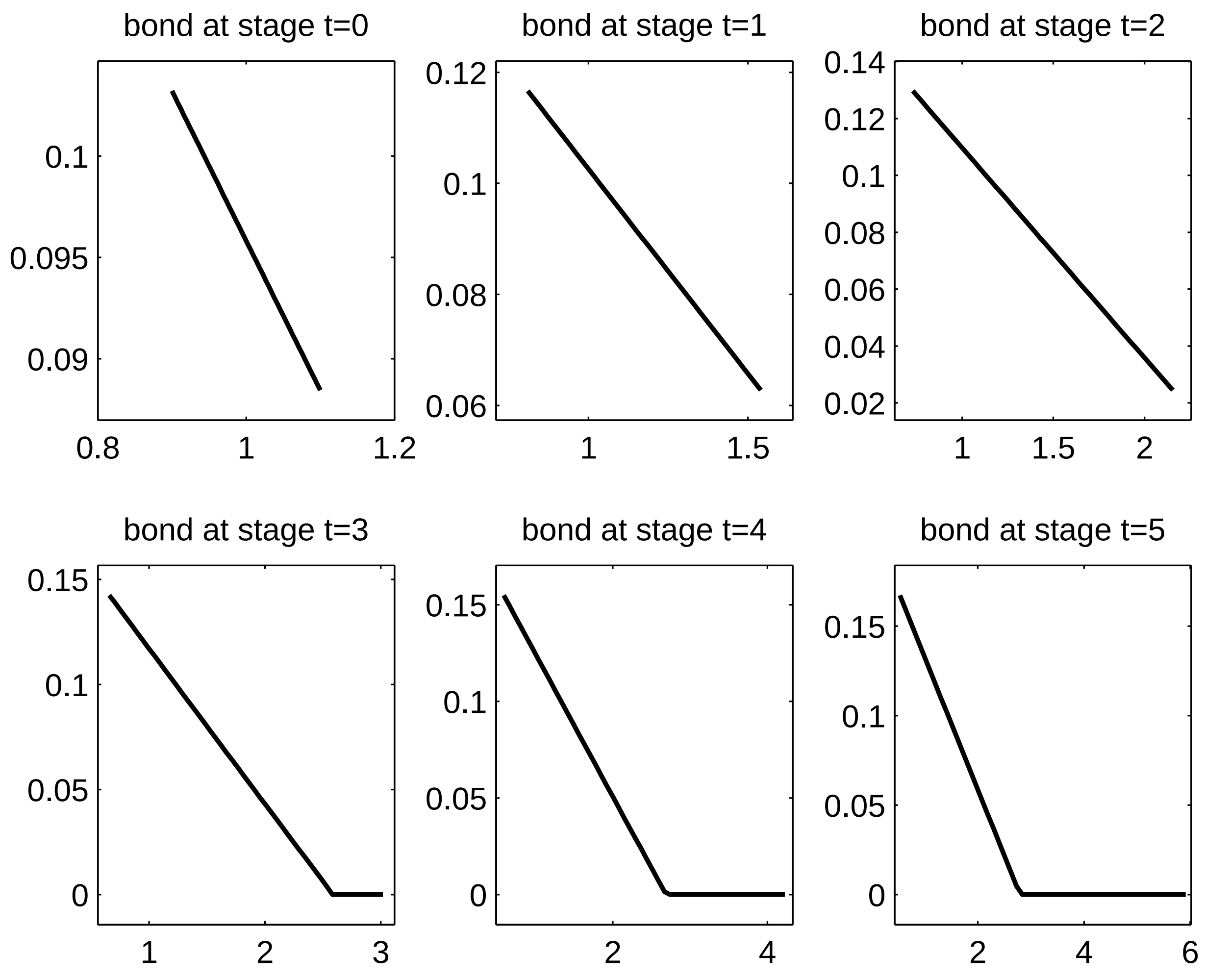
Note: $a(k)$ means $a \times 10^k$.

Errors of optimal solutions computed by numerical DP algorithms with Schumaker interpolation on m nodes using Lagrange vs. Hermite data

			Lagrange		Hermite	
γ	η	m	error of c_0^*	error of l_0^*	error of c_0^*	error of l_0^*
0.5	0.1	10	1.8(−2)	1.6(−1)	2.3(−3)	2.0(−2)
		20	4.8(−3)	5.5(−2)	2.4(−4)	1.8(−3)
		40	5.2(−4)	5.1(−3)	9.3(−5)	9.1(−4)
0.5	1	10	2.1(−2)	5.0(−2)	3.6(−3)	8.9(−3)
		20	6.6(−3)	1.9(−2)	3.2(−4)	9.3(−4)
		40	9.0(−4)	2.6(−3)	1.3(−4)	2.9(−4)
2	0.1	10	7.8(−3)	2.2(−1)	9.7(−4)	2.7(−2)
		20	2.2(−3)	6.7(−2)	1.0(−4)	2.9(−3)
		40	2.8(−4)	7.6(−3)	2.8(−5)	7.0(−4)
2	1	10	1.4(−2)	1.1(−1)	2.2(−3)	1.7(−2)
		20	5.2(−3)	4.3(−2)	3.0(−4)	2.0(−3)
		40	1.3(−3)	1.1(−2)	6.1(−5)	4.9(−4)
6	1	10	7.5(−3)	1.6(−1)	1.2(−3)	2.4(−2)
		20	2.8(−3)	6.1(−2)	1.5(−4)	2.8(−3)
		40	7.7(−4)	1.7(−2)	5.0(−5)	1.0(−3)

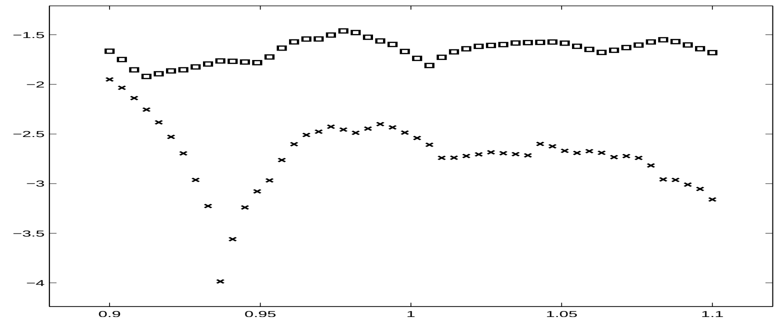
Note: $a(k)$ means $a \times 10^k$.

Exact optimal allocation and value functions

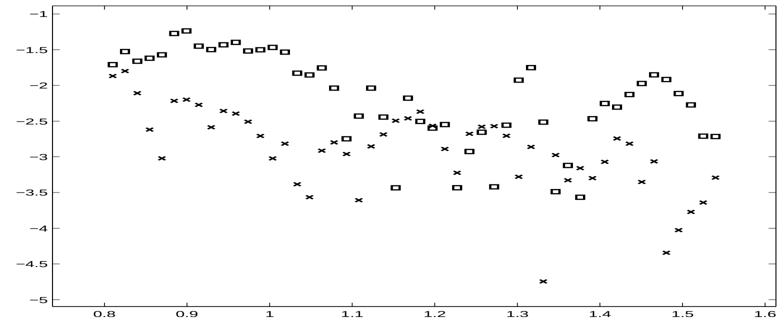


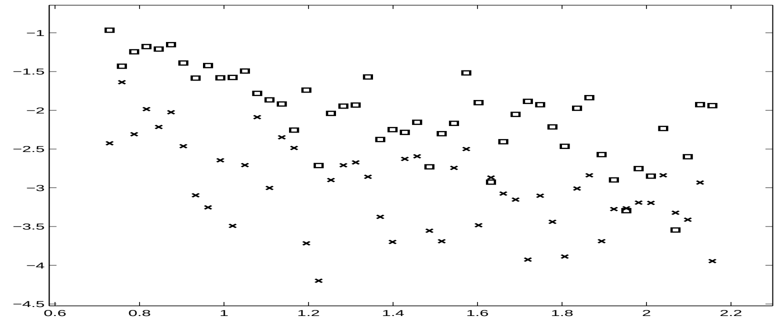
Relative Errors of Optimal Stock Allocations from Numerical DP with Schumaker interpolation using Lagrange vs Hermite data

$\log_{10}$(errors) of S_0

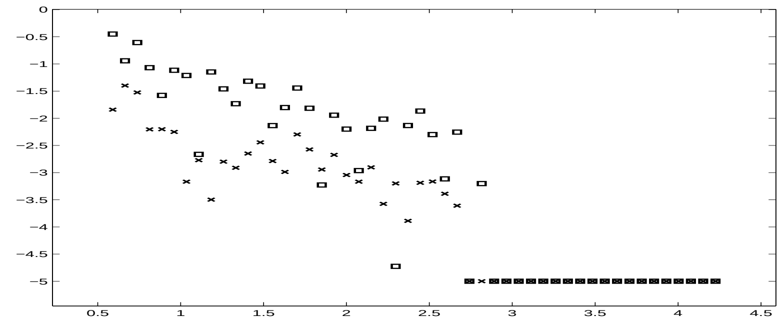


$\log_{10}$(errors) of S_1





$\log_{10}$(errors) of S_2



$\log_{10}$(errors) of S_4