BEPP 931 - Solution to Problem Set 8

Projection Method and Finite Difference Method

Shasha Wang and Cung Truong Hoang

April 9, 2020

Question 1 - Optimal Growth

The representative consumer solves

$$\max_{\{c\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u\left(c_t\right)$$

subject to $k_{t+1} = k_t + f\left(k_t\right) - c_t, \quad k_0$ given or, equivalently, with k_0 given.

$$\max_{\substack{\{k_{i+1}\}_{i=0} \\ t=0}} \sum_{t=0}^{\infty} \beta^t u\left(k_t + \hat{f}\left(k_t\right) - k_{t+1}\right)$$

The Euler equations are $-u'\left(k_t + f\left(k_t\right) - k_{t+1}\right) + \beta u'\left(k_{t+1} + f\left(k_{t+1}\right) - k_{t+2}\right)\left(1 + f'\left(k_{t+1}\right)\right) = 0, \quad t = 1, 2, \ldots$ or, equivalently,

$$-u'\left(c_t\right) + \beta u'\left(c_{t+1}\right)\left(1 + f'\left(k_t + f\left(k_t\right) - c_t\right)\right) = 0, \quad t = 0, 1, \ldots$$

note that the Euler equations are necessary but not sufficient.

Projection

The policy function $C(k)$ satisfies

$$-u'(C(k)) + \beta u'(C(k + f(k) - C(k)))\left(1 + f'(k + f(k) - C(k))\right) = 0$$

.

Define the operator $\mathcal{N}$ pointwise by

$$(\mathcal{N}C)(k) = -u'(C(k)) + \beta u'(C(k + f(k) - C(k)))\left(1 + f'(k + f(k) - C(k))\right)$$

Goal: Find C such that $\mathcal{N}\mathcal{C}=0$.

Approximate the policy function using the family of functions

$$\widehat{C}(k;a)$$

and choose the parameters a such that $\hat{C}(k;a)$ "almost" satisfies the operator equation $\mathcal{N}\mathcal{C}=0$

Iterative Method

Define the operator $\mathcal{F}$ pointwise by $\left(\mathcal{F}\left(C_1, C_2, C_3, C_4\right)\right)(k)$

$$=-u'\left(C_1(k)\right)+\beta u'\left(C_2\left(k+f(k)-C_3(k)\right)\right)\left(1+f'\left(k+f(k)-C_4(k)\right)\right)$$

Goal: Find C such that $\mathcal{F}(C,C,C,C)=0 \rightarrow$ wide variety of iterative methods.

Advantage: Avoid solving a (potentially large) system of nonlinear equations to choose the parameters a (time and space requirement of Jacobian.

Disadvantage: Slower convergence.

Time Iteration

Symbolically, the scheme is

$$\mathcal{F}\left(C^{l+1}, C^{l}, C^{l+1}, C^{l+1}\right)=0, \quad l=0,1,\ldots$$

Time iteration algorithm:

- Initialization: Choose a functional form for $\widehat{C}(k;a)$, where $a \in R^m$ and a set of points $K=\{k_i\}_{i=1}^{n}$, where $n \geq m$. Choose initial guess a^0 such that $\widehat{C}\left(k_i;a^0\right)=k_i+f\left(k_i\right), i=1,\ldots,n$, and stopping criterion ϵ

- Step 1: Compute c_i by solving

$$0=-u'\left(c_i\right)+\beta u'\left(\hat{C}\left(k_i+f\left(k_i\right)-c_i;a^l\right)\right)\left(1+f'\left(k_i+f\left(k_i\right)-c_i\right)\right), \quad i=1,\ldots,n$$

- Step 2: Compute a^{l+1} such that $\widehat{C}\left(k;a^{l+1}\right)$ approximates the Lagrange data $\{(k_i, c_i)\}_{i=1}^{n}$

- Step 3: If $\left\|a^{l+1}-a^{l}\right\|<\epsilon$, stop; otherwise, go to step 1 Step 1: Must be solved numerically $\rightarrow$ slow. Stable. Step 2: Function approximation.

Fixed Point Iteration

Symbolically, the scheme is

$$\mathcal{F}\left(C^{l+1}, C^{l}, C^{l}, C^{l}\right)=0, \quad l=0,1, \ldots$$

Fixed-point iteration algorithm: - Initialization: Choose a functional form for $\hat{C}(k;a)$, where $a \in R^m$ and a set of points $K=\{k_i\}_{i=1}^{n}$, where $n \geq m$. Choose initial guess a^0 such that $\widehat{C}\left(k_i; a^0\right)=f\left(k_i\right), i=1, \ldots, n$, and stopping criterion ϵ - Step 1: Compute c_i by solving

$$0=-u'\left(c_i\right)+\beta u'\left(\widehat{C}\left(k_i+f\left(k_i\right)-\widetilde{C}\left(k_i; a^l\right); a^l\right)\right)\left(1+f'\left(k_i+f\left(k_i\right)-\tilde{C}\left(k_i; a^l\right)\right)\right), \quad i=1, \ldots, n$$

- Step 2: Compute a^{l+1} such that $\widehat{C}\left(k; a^{l+1}\right)$ approximates the Lagrange data $\{(k_i, c_i)\}_{i=1}^{n}$ $-$ Step 3: If $\left\|a^{l+1}-a^l\right\|<\epsilon$, stop; otherwise, go to step 1 Step 1: Can be solved analytically $\rightarrow$ fast. Possibly unstable (stabilization/acceleration). Step 2: Function approximation.

Table 1 reports the time and max residual for all three methods.

	projection		Time Iteration		Fixed Point	
k	residual	time	residual	time	residual	time
1	2.960417e-02	1.457659e-02	2.960417e-02	1.187385e+00	2.960417e-02	1.171530e-01
2	2.012686e-03	1.188427e-02	2.012685e-03	1.236321e+00	2.012685e-03	1.137277e-01
3	1.546169e-04	1.269743e-02	1.546164e-04	1.055900e+00	1.546165e-04	7.526318e-02
4	1.281480e-05	1.109351e-02	1.281522e-05	1.180243e+00	1.281517e-05	7.862263e-02
5	1.109973e-06	1.461205e-02	1.110666e-06	8.019235e-01	1.110583e-06	4.689376e-02
6	9.887897e-08	1.273802e-02	9.845222e-08	1.038780e+00	9.850379e-08	6.024339e-02
7	8.980194e-09	1.280706e-02	9.016794e-09	6.002319e-01	8.974652e-09	2.800553e-02
8	8.270433e-10	1.361182e-02	1.227960e-09	3.523104e-01	1.189164e-09	2.912566e-02
9	7.702994e-11	1.284578e-02	9.474022e-10	2.147791e-02	9.021193e-10	1.648702e-03
10	7.204903e-12	1.683178e-02	7.908056e-10	1.871514e-02	7.104290e-10	9.941661e-04

Table 1. Error - Optimal Growth

We can see that in terms of accuracy, projection method performs the best. Time iteration is the slowest and fixed-point iteration is the fastest.

Figure 1 shows the policy function and residuals.

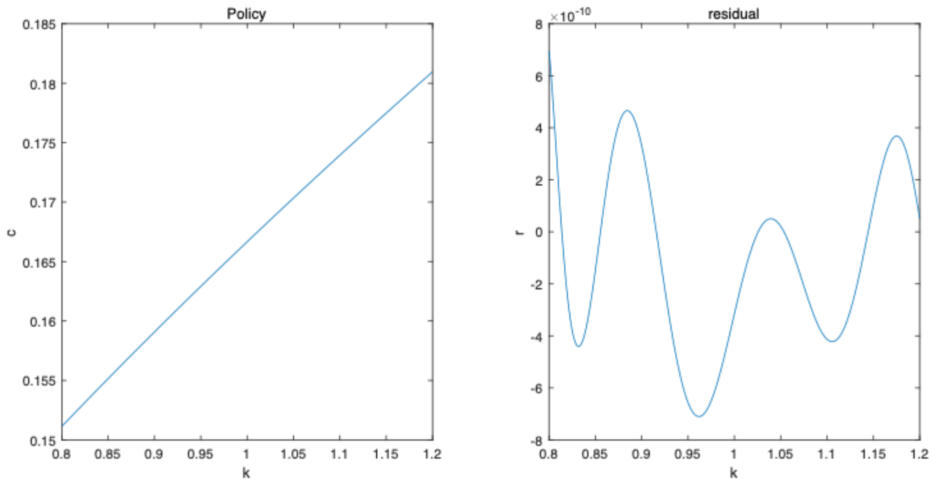

Figure 1. Optimal Growth

Question 2 - Inventory Problem - Without Stockouts

Figure 2 plots the inventory function against x on the left-hand side. For this graph, we choose a Chebyshev polynomial of order 8 ($n = 8$) for the approximation $\hat{I}(x; a)$ and 30 quadrature nodes to approximate the integral ($k = 30$). We use Gauss-Legendre to approximate the integral since $P(c)$ and $\hat{I}(x; a)$ are smooth. We see that the inventory function is monotonically increasing in x and linear. Our initial guess for I is a linearly increasing function with a constant. The right-hand side of the same figure shows the residual function equioscillating around 0.

We obtain the same graph for the inventory function when implementing time- and fixed-point iteration. Our fixed-point approach takes 45 iterations in 0.02 sec to converge. Time-iteration takes 49 iterations in 0.24 sec.

$n = m$	k	Chebyshev
2	30	2.7299e-01
4	30	3.0803e-02
6	30	2.9098e-03
8	30	1.4292e-04
10	30	1.9423e-05
20	30	1.5671e-04

Table 2. Max Error over $[0, 10]$

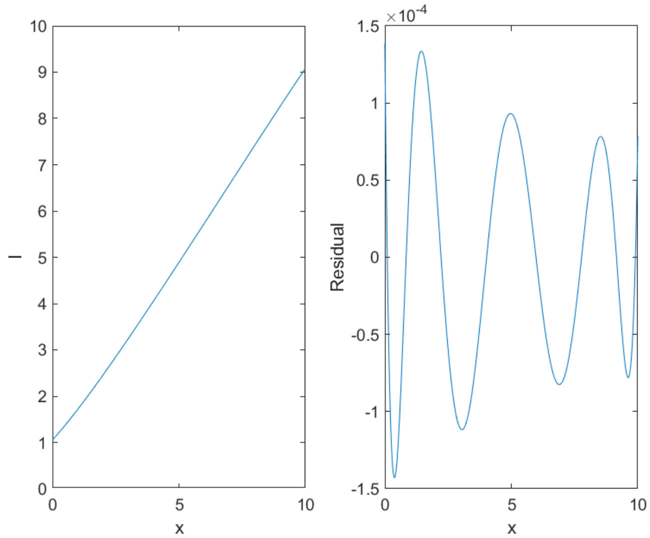

Figure 2. Inventory function and Residuals

Question 3 - Inventory Problem - With Stockouts

Figure 3 shows on the left graph the price function $h(x)$ plotted against x as a solid line. The dotted line depicts the inverse demand function $P(x) = 5 - x$. For this graph, we choose a linear spline with 8 nodes for the polynomial approximation ($n = 8$) and 30 quadrature nodes to approximate the integral ($k = 30$). The right-hand graph shows the oscillating residuals. We obtain the similar graphs for the price function using time- and fixed-point iteration. Our algorithm for the fixed-point iteration takes 25 iterations in 0.03 sec. Our time-iteration takes 33 iterations in 0.3 sec. Our initial guess is $h = max\{1, P(x)\}$. Since $h(x)$ has a kink, we approximate the integral with the trapezoid rule. We repeat this exercise with a cubic spline approximation. Figure 4 shows that the residuals are large for small values of x and residuals are very small for larger values of x. This is also reflected in the graph of h where, instead of a kinked graph, we obtain a smoothed graph. In Table 3 we report the maximum errors over the interval $[0, 10]$ for both linear and cubic splines with varying number of nodes. With a cubic spline we consistently obtain lower maximum errors than with a linear spline except for $n = 8$.

Figure 5 shows our results for Wright-Williams Smoothing. We approximate $\Psi(x)$ with a Chebyshev polynomial of order 8 since it is a smooth function. We plot on the left-hand side the smoothed function $\Psi(x)$ against x and on the right-hand side we show the resulting function $h(x)$. Note that the kink does not coincide with the intersection of the lines in the graph on the left since we need to discount $\Psi(x)$ with β. Takes 23 iterations in 0.05 sec.

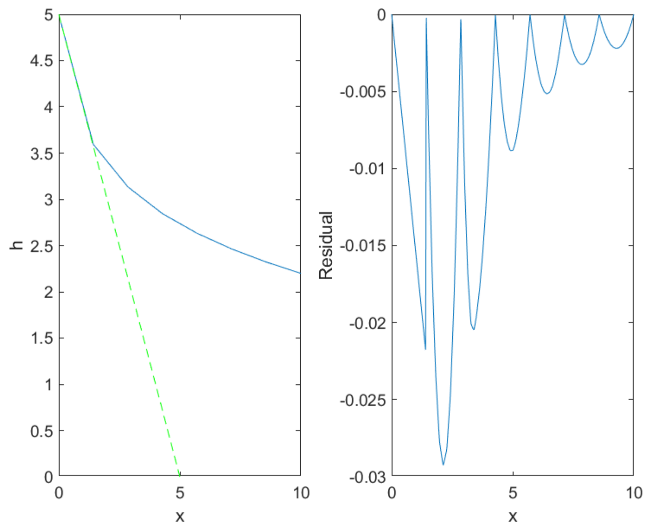

Figure 3. Price function and Residuals with Linear Spline

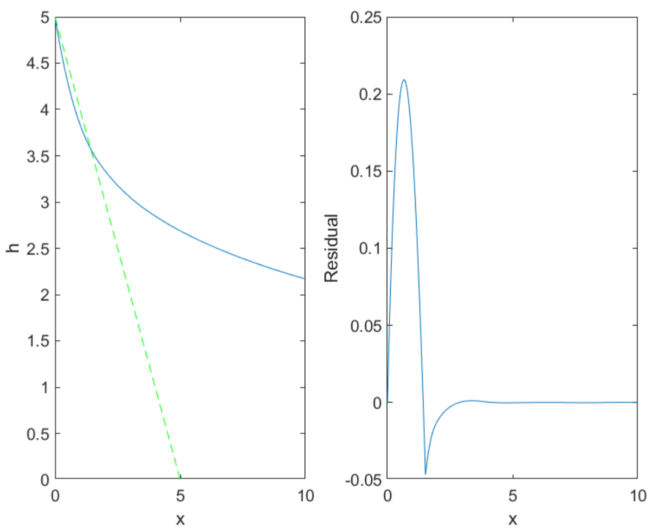

Figure 4. Price function and Residuals with Cubic Spline

$n = m$	k	Linear Spline	Cubic Spline
2	30	5.2312e-01	-
4	30	6.2742e-01	4.1878e-01
6	30	3.5632e-01	1.0899e-01
8	30	2.9249e-02	2.0928e-01
10	30	1.8168e-01	1.2687e-01
20	30	8.4572e-02	2.4569e-01

Table 3. Max Error over $[0, 10]$

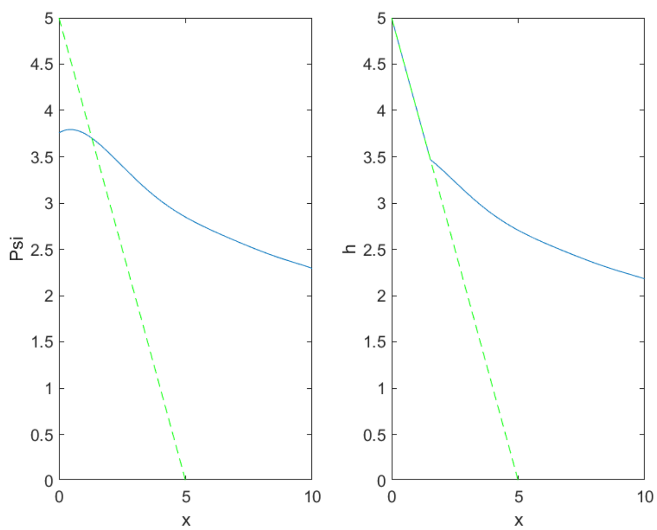

Figure 5. Ψ and Price function with Wright-Williams Smoothing