otes on Numerical Techniques for Solving Dynamic

N

Economic Models

Nelson C. Mark[1]

July 17, 2004

[1]I thank Young-Kyu Moh and Raphael Solomon for correcting many typos in an earlier draft.

Contents

List of programs

1. LINEAR-1.PGM. Log-linearized solution and simulation of *stochastic* optimal growth model.

2. LEASTSQ1.PGM. Solution of *deterministic* optimal growth model by least-squares projection method.

3. CHEB_EX1.PGM. Examples of using Chebyshev polynomial approximation to known functions.

4. CHEB_EX2.PGM. Using Chebyshev polynomials to approximate the Cobb-Douglas function.

5. COL_DET1.PGM. Collocation solution of deterministic optimal growth model by parametric value function iteration.

6. COL_DET3.PGM. Collocation method solution of deterministic optimal growth model by policy function approach. Uses the nonlinear equation solver NLSYS.SET.

7. SGM_VAL.PGM. Collocation method solution of stochastic optimal growth model by value function iteration. Function approximation by Chebyshev polynomial.

8. MPEA.PGM. Collocation method solution of Christiano and Fisher's modified parameterized expectations algorithm applied to the stochastic growth model.

These programs come with *no warranty* as to their accuracy. Use at your own risk. The code is inelegant and inefficient. Neither has it undergone serious development or testing.

Chapter 1

Overview

These notes are a brief guide to obtaining numerical solutions to dynamic economic problems. The canonical example used in the notes is the optimal stochastic growth model.

The best way to learn (at least for me) is to start out by applying very simple techniques to concrete examples and to build up from there. So even though there are many methods and applications of interest to economists, these notes are focused on one method that has been found to work well, and on one application that should be pretty familar to all.

Fabrice Collard has a set of lecture notes that is a practical version of Judd. He shows you how to implement the techniques in Judd, but Judd mainly discusses the theoretical aspects (properties) of the techniques. I downloaded his notes when he was teaching this material at UBC and asked him if it was okay to use it in my teaching. He said it was okay, but I feel a little weird distributing it to others. Besides, the notes that I got were still incomplete and had a lot of typos. Try sending him e-mail to see if he'll send his notes to you.

Fackler and Miranda's text is even more basic than Collard's notes. It is (in my opinion) to computational methods in econmics what *Alpha Chiang's* book is to math econ. This is a good source for quickly familiarizing oneself with the material. It comes with a full set of Matlab code and the book has a professional web site. It's almost a little too "canned" and their examples aren't the ones macroeconomists have much interest in.

There also is an excellent book edited by Marimon and Scott that contains contributions from many of the leaders in developing and using these techniques.

References

Collard, Fabrice. Lecture notes on Numerical Methods in Economics. His web page is: http://fabcol.free.fr/.

Judd, Kenneth. *Numerical Methods in Economics*, MIT Press, 1998.

Marimon, Ramon and Andrew Scott, eds. *Computational Methods for the Study of Dynamic Economies*, Oxford University Press, 1999.

Miranda, Mario J. and Paul L. Fackler, *Applied Computational Economics and Finance,* MIT Press, 2002. See also their CompEcon web site: http: http://www4.ncsu.edu/ pfackler/compecon/toolbox.html

1.1 The Stochastic Growth Model

Consider the one-sector stochastic growth model. We have an infinitely-lived representative household that supplies labor *inelastically* to the representative firm which is owned by the household. The household chooses sequences of consumption $\{c_t\}$, and saving (gross investment) $\{i_t\}$ to maximize lifetime utility. Firms are competitive and operate under constant returns to scale with *Cobb-Douglas* production function using labor and capital. Assuming complete markets, the decentralized competitive equilibrium can be supported by a Pareto-Optimum which is obtained as the solution of the social planner's problem,

$$\max_{\{c_t\}} \mathrm{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{1.1}$$

subject to

$$\begin{aligned} y_t &= f(k_t, \ell_t; A_t) = A_t k_t^{\alpha} \ell_t^{1-\alpha} = c_t + i_t && (1.2)\\ k_{t+1} &= i_t + k_t(1-\delta) && (1.3)\\ \ell_t &= 1 && (1.4)\\ u(c_t) &= \frac{c_t^{1-\gamma}}{1-\gamma} && (1.5) \end{aligned}$$

with k_0 given, A_t a stochastic productivity shock, and $0 \leq \delta \leq 1$ the capital depreciation rate.

Prime notation. To ease our demands on notation, we use a 'prime' to denote next

period values and a 'double prime' to denote values two-periods hence. The current state is (k, A). Given the current state, we know y and must choose i and c. But once we've chosen i, we get c as a residual from the *resource constraint* (1.2). Also, choosing i is equivalent to choosing k' since k is given. From (1.2)–(1.4) write consumption as

$$c = h(k, k', A) = f(k, A) - k' + k(1 - \delta) \tag{1.6}$$

where

$$\begin{aligned} h_1 &= f_k + 1 - \delta \\ h_2 &= -1 \\ h_3 &= f_A \\ f_k &= \alpha A k^{\alpha-1} \\ f_{kk} &= \alpha(\alpha - 1) A k^{\alpha-2} \\ f_A &= k^{\alpha} \\ f_{Ak} &= f_{kA} = \alpha k^{\alpha-1} \\ u_c &= c^{-\gamma} \\ u_{cc} &= \frac{-\gamma}{c^{1+\gamma}} \end{aligned}$$

The Value function. Using (1.6), rewrite the problem in recursive form,

$$V(k, A) = \max_{k'} \{u\left[h(k, k', A)\right] + \beta \mathrm{E}\left[V(k', A')\right] \tag{1.7}$$

where $V(k, A)$ is the value function. From the first-order conditions, we get

$$V_1(k, A) = u_c(k, k', A) h_1(k, k', A) \tag{1.8}$$

$$u_c(k, k', A) h_2(k, k', A) = -\beta \mathrm{E}\left[V_1(k', A)\right] \tag{1.9}$$

(1.8) is the envelope condition. This gives the *Euler equation*

$$u_c(c) = \beta \mathrm{E} u_c(c') \left[f_k(A', k') + (1 - \delta)\right] \tag{1.10}$$

or with the developed notation,

$$h(k, k', A)^{-\gamma} = \beta \mathrm{E}\left[h(k', k'', A')^{-\gamma} \left(\alpha A' k'^{(\alpha-1)} + (1 - \delta)\right)\right] \tag{1.11}$$

To solve the model, we can work on either (1.7) by the *value function approach* or (1.11) by the *policy function approach.*

Chapter 2

Log Linearization

Maybe the most intuitive thing to do is to log-linearize and solve the Euler equation (1.11). We might want to log-linearize as opposed to simply linearizing because then the coefficients have interpretations as elasticities, and because we usually want to take logs of the data so this will match up the theoretical variables with the observations.

Remarks

1. Linearization is done by first-order Taylor expansion. *Question:* Around what point do we linearize? *Answer*: The steady state.

2. Linearization of the Euler equation is equivalent to maximizing a quadratic (second-order) expansion of the objective function.

2.1 The Steady State

Denote *steady state* values with an overbar. What does it mean to be in a steady state mean when there is a stochastic productivity shock A? We're talking about where the economy will settle if we shut down the productivity shock. Let $\mathrm{E}(A) = 1$, and we'll shut down the shocks by thinking about a steady state value of $\bar{A} = 1$. Then in the steady state with $k' = k$, (1.11) becomes

$$\frac{1}{\beta} = f_k + 1 - \delta \tag{2.1}$$

which gives

$$\bar{k} = \left(\frac{1 - \beta\,(1-\delta)}{\beta\,\alpha} \right)^{1/(\alpha-1)} \tag{2.2}$$

Steady state values $\bar{c}$, $\bar{y}$ and $\bar{i}$ follow directly. In these notes, we will use the following values.

β	0.990
δ	0.025
α	0.40
$\bar{k}$	57.708
$\bar{y}$	5.064
$\bar{i}$	1.443
$\bar{c}$	3.621

We will use the following law of motion for productivity shock:

$$A_t = (1-\rho) + \rho A_{t-1} + \epsilon_t \tag{2.3}$$

where $\rho = 0.93$, $\epsilon_t \overset{NID}{\sim} (0, 0.010224^2)$.

Procedure

1. Drop the expectations operator. Linearize the Euler equation around the steady state.

2. Solve the resulting (system of) stochastic difference equation(s).

3. Re-introduce expectations. We can do this because linear rational expectations models posess the property of certainty equivalence.

Implementation

First-order Taylor expansion around (1.11). Begin with the right hand side. Partial derivatives are indicated with subscripts. h_j means partial differentiation with respect to the $j-$th argument. f_{kk} means the second derivativative of the production function with respect to capital, u_{cc} is the second derivative of the utility function, etc. I'm going to be sloppy with the notation and omit the functional dependence of partial derivatives.

A 'hat' means percentage deviation from the steady state value. That is, $\hat{x} = (x - \bar{x})/\bar{x}$.

Expand the Euler equation. Begin with the *rhs*:

$$\begin{aligned}\beta \mathrm{E} u_c(c')\left[f_k(A', k') + (1-\delta)\right] &= \beta u_c(f_k - \delta\bar{k}) \\ &+\beta u_{cc} h_2 (f_k - \delta\bar{k})\bar{k}\hat{k}'' + \left(\beta\left[(f_k - \delta\bar{k})u_{cc}h_1 - u_c\right]\bar{k}\right)\hat{k}' \\ &+\left(\beta u_c(f_{kk} + 1 - \delta)\bar{k}\right)\hat{k} + \beta(f_k - \delta\bar{k})u_{cc}h_3\hat{A}' + \beta u_c f_{kA}\hat{A}\end{aligned}$$

Expand the *lhs*.

$$u_c(c) = u_c + (u_{cc}h_1\bar{k})\hat{k} + (u_{cc}h_2\bar{k})\hat{k}' + u_{cc}h_3\hat{A}$$

Equate and re-arrange to get,

$$a_0 + a_1\hat{k}'' + a_2\hat{k}' + a_3\hat{k} + a_4\hat{A}' + a_5\hat{A} = 0 \tag{2.4}$$

where

$$\begin{aligned} a_0 &= u_c\beta(f_k + 1 - \delta) - u_c = 0 \\ a_1 &= \beta u_{cc}h_2(f_k - \delta\bar{k})\bar{k} \\ a_2 &= \left(\beta(f_k - \delta\bar{k})u_{cc}h_1 - \beta u_c - u_{cc}h_2\right)\bar{k} \\ a_3 &= \left(\beta u_c(f_{kk} + 1 - \delta) - u_{cc}h_1\right)\bar{k} \\ a_4 &= \beta(f_k - \delta\bar{k})u_{cc}h_3 \\ a_5 &= \beta u_c f_{kA} - u_{cc}h_3 \end{aligned}$$

The model reduces to a second-order linear stochastic difference equation in k. Re-introducing the time subscripts gives

$$a_1\hat{k}_{t+2} + a_2\hat{k}_{t+1} + a_3\hat{k}_t + a_4\hat{A}_{t+1} + a_5\hat{A}_t = 0 \tag{2.5}$$

Solving the stochastic difference equation. Rewrite above equation as

$$(1 - b_1 L - b_2 L^2)\hat{k}_{t+1} = w_t, \tag{2.6}$$

where

$$\begin{aligned} w_t &= b_3\hat{A}_t + b_4\hat{A}_{t-1} \\ b_1 &= -a_2/a_1 \end{aligned}$$

$$
\begin{aligned}
b_2 &= -a_3/a_1 \\
b_3 &= -a_4/a_1 \\
b_4 &= -a_5/a_1
\end{aligned}
$$

The roots of the polynomial $(1 - b_1 z - b_2 z^2) = (1 - \omega_1 z)(1 - \omega_2 z)$ satisfy $b_1 = \omega_1 + \omega_2$ and $b_2 = -\omega_1 \omega_2$. Using the quadratic formula and evaluating at the parameter values that we used to calibrate the model, the roots are,
$z_1 = (1/\omega_2) = [-b_1 + \sqrt{b_1^2 + 4b_2}]/(2b_2) \simeq 1.005$, and
$z_2 = (1/\omega_1) = [-b_1 - \sqrt{b_1^2 + 4b_2}]/(2b_2) \simeq -0.503$. There is a stable root, $|z_1| > 1$ which lies outside the unit circle, and an unstable root, $|z_2| < 1$ which lies inside the unit circle. The presence of an unstable root means that the solution is a saddle path. If you try to simulate (2.6) directly, the capital stock will diverge.

To solve the difference equation, exploit the *certainty equivalence* property of quadratic optimization problems. That is, you first get the *perfect foresight* solution to the problem by solving the stable root backwards and the unstable root forwards. Then, replace future random variables with their expected values conditional upon the time-t information set. Begin by rewriting (2.6) as

$$
\begin{aligned}
w_t &= (1 - \omega_1 L)(1 - \omega_2 L)\hat{k}_{t+1} \\
&= (-\omega_2 L)(-\omega_2^{-1} L^{-1})(1 - \omega_2 L)(1 - \omega_1 L)\hat{k}_{t+1} \\
&= (-\omega_2 L)(1 - \omega_2^{-1} L^{-1})(1 - \omega_1 L)\hat{k}_{t+1},
\end{aligned}
$$

and rearrange to get

$$
\begin{aligned}
(1 - \omega_1 L)\hat{k}_{t+1} &= \frac{-\omega_2^{-1} L^{-1}}{1 - \omega_2^{-1} L^{-1}} w_t \\
&= -\left(\frac{1}{\omega_2} L^{-1}\right) \sum_{j=0}^{\infty} \left(\frac{1}{\omega_2}\right)^j w_{t+j} \\
&= -\sum_{j=1}^{\infty} \left(\frac{1}{\omega_2}\right)^j w_{t+j}. \qquad (2.7)
\end{aligned}
$$

The autoregressive specification for the shocks implies the prediction formulae

$$
\mathrm{E}_t w_{t+j} = b_3 \mathrm{E}_t \hat{A}_{t+j} + b_4 \mathrm{E}_t \hat{A}_{t+j-1} = [b_3 \rho + b_4] \rho^{j-1} \hat{A}_t.
$$

Use this forecasting rule in (2.7) to get

$$\sum_{j=1}^{\infty}\left(\frac{1}{\omega_2}\right)^j \mathrm{E}_t w_{t+j} = [b_3\rho + b_4]\frac{\hat{A}_t}{\rho}\sum_{j=1}^{\infty}\left(\frac{\rho}{\omega_2}\right)^j = \left[\frac{1}{\omega_2 - \rho}\right](b_3\rho + b_4)\hat{A}_t.$$

It follows that the solution for the capital stock is

$$\hat{k}_{t+1} = \omega_1\hat{k}_t - \left[\frac{1}{\omega_2 - \rho}\right][b_3\rho + b_4]\hat{A}_t. \tag{2.8}$$

Recover the variables of interest. Note that

$$\ln k_t \simeq \hat{k}_t + \ln\bar{k}$$

and plug back into the model.

$$\begin{aligned}
\hat{y}_t &= \hat{A}_t + \alpha\hat{k}_t \\
\hat{i}_t &= (\bar{k}/\bar{i})\hat{k}_{t+1} + (1-\delta)(\bar{k}/\bar{i})\hat{k}_t \\
\hat{c}_t &= (\bar{y}/\bar{c})\hat{y}_t + (\bar{i}/\bar{c})\hat{i}_t
\end{aligned}$$

Then

$$\begin{aligned}
\ln(y_t) &= \hat{y}_t + \ln(\bar{y}) \\
\ln(i_t) &= \hat{i}_t + \ln(\bar{i}) \\
\ln(k_t) &= \hat{k}_t + \ln(\bar{k}) \\
\ln(c_t) &= \hat{c}_t + \ln(\bar{c})
\end{aligned}$$

Remarks

1. Part of the reason that the model does not match volatility correctly because there is no labor decision.

2. The optimal growth model example became a problem in solving a single stochastic difference equation. Blanchard and Kahn[1] show how to handle multivariate models.

[1] "The Solution of Linear Difference Models under Rational Expectations," *Econometrica*, 1980, 48,5: 1305-1311.

2.2 Simulations and Impulse Responses

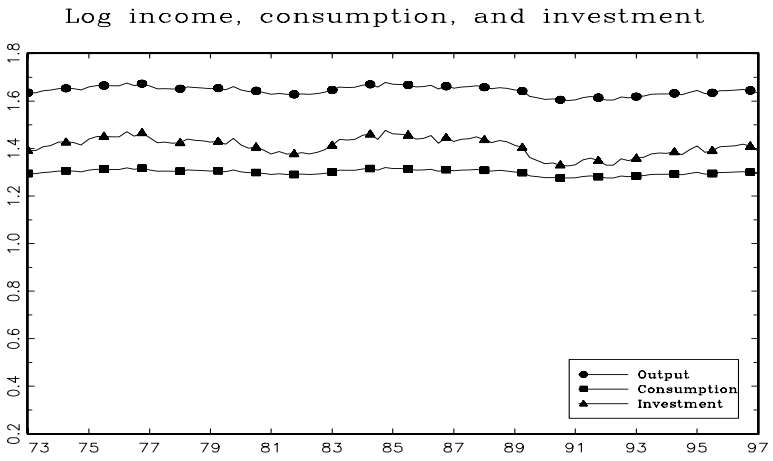

Figure 2.1: Simulated Observations from Log-Linearized RBC model

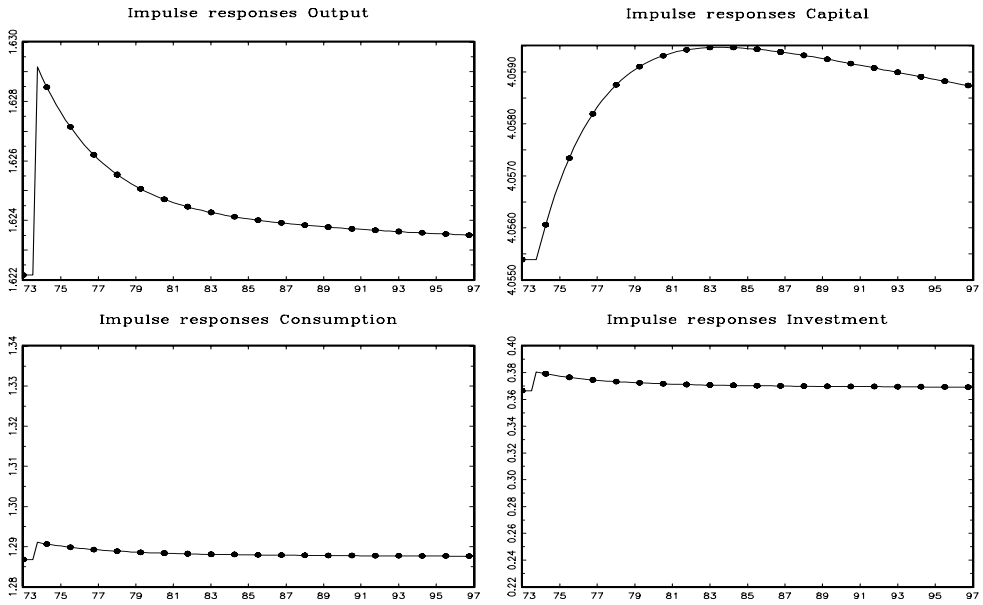

Figure 2.2: Impulse Response from one standard deviation shock

```
new;
#include ols.set;
#include correlgm.set;
@-------------------------------------------------------------------------@
@ LINEAR-1.PGM (adapted from RBC-1.PGM). @
@ Calibrate and simulate a one-country stochastic @
@ growth model.  CRRA utility, no leisure choice @
@-------------------------------------------------------------------------@
output file=linear-1.out reset; output on;
format /rd 8,3;
@=========================================================================@
@::::::::::::::  PARAMETER VALUES ::::::::::::::::::::::::::::::::::::::::@
@------------- CRRA Utility ------------------------------------@
alpha=0.4; @ CAPITAL'S SHARE @
delta=0.025; @ QUARTERLY DEPRECIATION RATE @
beta=0.990; @ QUARTERLY DISCOUNT FACTOR-NORMALIZED BY EFFICIENCY UNITS @
gam=2.0; @ COEFFICIENT OF RELATIVE RISK AVERSION @
rho=0.93; @ AR(1) COEFFICIENT FOR TECHNOLOGY SHOCKS. @
sigma=0.007; @ Innovation Standard Deviation @
sto1=1-beta*(1-delta); @--------------@
sto2=beta*alpha; @ Steady State @
sto3=1/(alpha-1); @--------------@
Kss=(sto1/sto2)^sto3; @ Steady state capital @
Yss=Kss^alpha; @ Steady state output @
Iss=delta*Kss; @ Steady state investment @
Css=Yss-Iss; @ Steady state consumption @
@-----------------------------------------------------------------@
" rho sigma alpha delta beta gamma";
rho sigma alpha delta beta gam;
"
Steady State Values";
" Capital Output Invest Consumption"; Kss Yss Iss Css;
@=========================================================================@
@--- MODEL DYNAMICS ----@
@=========================================================================@
F_k=alpha*Kss^(alpha-1);
F_kk=alpha*(alpha-1)*Kss^(alpha-2);
F_kA=alpha*Kss^(alpha-1);
u_c=Css^(-gam);
u_cc=-gam*(Css^(-1-gam));
H_1=F_k+1-delta;
H_2=-1;
H_3=Kss^(alpha);
a=zeros(5,1);
a[1] = beta*u_cc*h_2*(f_k-delta*kss)*kss;
a[2] = (beta*(f_k-delta*kss)*u_cc*h_1-beta*u_c-u_cc*h_2)*kss;
a[3] = (beta*u_c*(f_kk+1-delta)-u_cc*h_1)*kss;
```

```
a[4] = beta*(f_k-delta*kss)*u_cc*h_3;
a[5] = beta*u_c*f_ka-u_cc*h_3;
b=-a[2:5]./a[1];
"a-coefficients";;a'; "b-coefficients";; b';
z2=-b[1]-sqrt(b[1]*b[1]+4*b[2]); z2=z2/(2*b[2]); @ Roots @
z1=-b[1]+sqrt(b[1]*b[1]+4*b[2]); z1=z1/(2*b[2]);
w1=1/z1; w2=1/z2; "Roots:  ";; "w1=";;w1;;" w2=";; w2;
@-------------------------------------------------------------@
@:::::::::::::::  Simulate ::::::::::::::::::::::::::::::::::::@
@-------------------------------------------------------------@
t1=101;
seed=120455; e=sigma*rndns(t1,1,seed);
Ahat=zeros(t1,1);
t=2; do while t le t1;
Ahat[t]=rho*Ahat[t-1]+e[t]; @ Sequence of A-hats @
t=t+1; endo;
A=1+Ahat; @ Sequence of As @

sto=1/(w2-rho);
khat=zeros(t1,1);yhat=zeros(t1,1); ihat=zeros(t1,1); chat=zeros(t1,1);

t=2; do while t le t1-1; @ APPROXIMATE LOG PERTUBATIONS @
khat[t]=w1*khat[t-1]-sto*(b[3]*rho+b[4])*ahat[t-1]; @ Sequence of k-hats @
t=t+1; endo; @ MUST DEFINE KHATS FIRST @

t=2; do while t le t1-1;
yhat[t]=Ahat[t]+alpha*khat[t];
ihat[t]=(Kss/Iss)*(khat[t+1]-(1-delta)*khat[t]);
chat[t]=(Yss/Css)*yhat[t]-(Iss/Css)*ihat[t];
t=t+1; endo;

khat=trimr(khat,1,1); yhat=trimr(yhat,1,1);
ihat=trimr(ihat,1,1); chat=trimr(chat,1,1); A=trimr(A,1,1);

logy=yhat+ln(Yss); @ Log approximation (trick) @
logk=khat+ln(Kss); @ In levels, Investment is sometimes negative @
logc=chat+ln(Css); @ because nonnegativity constraint is not imposed @
logi=ihat+ln(Iss);
@-------------------------------------------------------------@
@::::::::::::  Compute descriptive statistics ::::::::::::::@
@-------------------------------------------------------------@
"Volatility of log";
"output consumption investment";
stdc(logy logc logi)';
@-------------------------------------------------------------@
@::::::::::::  Impulse response ::::::::::::::::::::::::::::::@
@-------------------------------------------------------------@
t1=101;
```

```
Ahat=zeros(t1,1); t0=5;
t=t0; do while t le t1;
Ahat[t]=(rho^(t-t0))*sigma;
t=t+1; endo;
A=1+Ahat; @ Sequence of As @

sto=1/(w2-rho);
khat=zeros(t1,1);yhat=zeros(t1,1); ihat=zeros(t1,1); chat=zeros(t1,1);

t=2; do while t le t1-1; @ APPROXIMATE LOG PERTUBATIONS @
khat[t]=w1*khat[t-1]-sto*(b[3]*rho+b[4])*ahat[t-1]; @ Sequence of k-hats @
t=t+1; endo;
t=2; do while t le t1-1;
yhat[t]=Ahat[t]+alpha*khat[t];
ihat[t]=(Kss/Iss)*(khat[t+1]-(1-delta)*khat[t]);
chat[t]=(Yss/Css)*yhat[t]-(Iss/Css)*ihat[t];
t=t+1; endo;

khat=trimr(khat,1,1); yhat=trimr(yhat,1,1);
ihat=trimr(ihat,1,1); chat=trimr(chat,1,1); A=trimr(A,1,1);

logy=yhat+ln(Yss); @ Log approximation (trick) @
logk=khat+ln(Kss); @ In levels, Investment is sometimes negative @
logc=chat+ln(Css); @ because nonnegativity constraint is not imposed @
logi=ihat+ln(Iss);
@ Graphing code omitted @
output off;
```

Chapter 3

Drawbacks of Linearization

3.1 Importance of Higher-Ordered Moments

The (log) linearization technique exploits the idea of *certainty equivalence*. The Euler equation that we expanded involves an expectation but we did *not* expand the expectation. We dropped the expectation, linearlized, iterated on the implied second-order stochastic difference equation and then re-introduced the expectations operator at the very end. The form of the solution is equivalent to the solution under perfect forsight of the future $\hat{A}_t$ shocks except we replace future $\hat{A}_{t+j}$ with their conditional expectation, $\mathrm{E}_t(\hat{A}_{t+j})$.

Under certainty equivalence, only first moments matter. Thus if higher-order moments are important in the theory, the log-linearization technique will sweep these effects away. A simple example to illustrate this point draws from Hansen and Singleton's (1983) asset pricing model under log-normality.

They begin with Lucas's (1978) intertemporal asset pricing model. The Euler equation associated with the representative investor's portfolio problem is

$$u_c(c) = \beta \mathrm{E}\left(u_c(c')\left(\frac{p'+d'}{p}\right)\right) \tag{3.1}$$

where we use the prime notation for next period values. p is the price of an asset that pays a stochastic dividend d. Let the utility function be CRRA, denote the asset payoff as r and consumption growth $c'/c \equiv x'$. Then (3.1) can be re-written as

$$1 = \beta \mathrm{E} x'^{-\gamma} r' \tag{3.2}$$

where γ is the coefficient of relative risk aversion.

Now assume that conditional on last period's information, (x, r) are jointly log-

normally distributed,[1]

$$\begin{pmatrix} \ln(x) \\ \ln(r) \end{pmatrix} \sim N\left(\begin{bmatrix} \mu_x \\ \mu_r \end{bmatrix}, \begin{bmatrix} \sigma_x^2 & \sigma_{xr} \\ \sigma_{xr} & \sigma_r^2 \end{bmatrix} \right)$$

Then

$$\begin{aligned} -\gamma \ln(x) + \ln(r) &\sim N(\mu_r - \gamma\mu_x, \gamma^2\sigma_x^2 + \sigma_r^2 - 2\gamma\sigma_{xr}) \\ \Rightarrow \quad \mathrm{E}\left(x^{-\gamma} r\right) &= \exp\left(\mu_r - \gamma\mu_x + \frac{\gamma^2\sigma_x^2 + \sigma_r^2 - 2\gamma\sigma_{xr}}{2}\right) \end{aligned}$$

Plug this back into (3.2) and rearrange to get

$$\mu_{rt} = \gamma\mu_{xt} - \frac{\gamma^2\sigma_{xt}^2 + \sigma_{rt}^2 - 2\gamma\sigma_{xr,t}}{2} - \ln(\beta) \tag{3.3}$$

The t subcripts indicate that these are conditional moments where $\mu_{rt} = \mathrm{E}_t \ln(r_{t+1})$ and $\mu_{xt} = \mathrm{E}_t \ln(x_{t+1})$. In any event, (3.3) tells us that the conditional return most clearly depends on second moments of consumption growth and of returns.

Now suppose that we log-linearize (3.2) instead. Assume that steady-state consumption growth is zero $\bar{x} = 1$, and the steady state payoff is $\bar{r}$. What we get is,

$$1/\beta = \bar{r} - \gamma\bar{r}\hat{x} + \bar{r}\hat{r} \tag{3.4}$$

You don't have to stare at (3.4) very long to see that a really notable feature is the absense of second moments.

3.2 Complementary Slackness, Nonlinear Constraints

Another potential drawback to linearization is that it is difficult to handle binding constraints. Suppose we impose a nonnegativity constraint on investment such that

$$i_t = k_{t+1} - (1-\delta)k_t \geq 0$$

It is difficult to handle such a constraint properly using the log-linearization technique. The decision rule is not differentiable at the point where the constraint binds.

[1]The rule of log-normality is if $\ln(Y) \sim N(\mu, \sigma^2)$, then $\mathrm{E}(Y) = \mathrm{E}\left(e^{\ln(Y)}\right) = e^{\mu + \frac{\sigma^2}{2}}$

Chapter 4

Objectives for Approximation Techniques

Here's what we want to do.

4.1 Value Function Solution

The Bellman equation for the stochastic growth model is

$$V(k,A) = \max\left\{\frac{c^{1-\gamma}}{1-\gamma} + \beta \mathrm{E}V(k',A')\right\}$$

we will solve the model by parametrically approximating the unknown function $V(k,A)$.

4.2 Policy Function Solution

Look at the Euler equation, as we developed it earlier. The model is given by

$$\begin{aligned} U_c(c) &= \beta \mathrm{E}\left\{U_c(c')\left[f_k(A',k') + (1-\delta)\right]\right\} \\ c &= h(k,k',A) = f(k,A) - k' + k(1-\delta) \\ f(k,A) &= Ak^{\alpha} \\ U_c(c) &= c^{-\gamma} \\ f_k(A,k) &= \alpha A k^{\alpha-1} \end{aligned}$$

The problem is to choose k'. The optimal decision rule is contingent on the current state. In fact, it will be an exact function of the current state (no error term) which

is (k, A). We can represent the optimal decision (transition) rule as

$$k' = g(k, A)$$

We need to solve for this unknown function $g(\cdot)$. We can write

$$c = h(k, g(k, A), A) = \tilde{h}(k, A) = f(k, A) - g(k, A) + k(1 - \delta).$$

Now discretize the state space. Let A evolve according to a two-state Markov chain, where

$$p_{ij} = \text{Prob}\left(A' = a_j | A = a_i\right)$$

$A_t = a_1$ is the good (high) state and $A_t = a_2$ is the bad (low) state, and

$$\begin{aligned} p_{11} &= p & p_{12} &= 1 - p \\ p_{21} &= 1 - q & p_{22} &= q \end{aligned}$$

Rewrite the Euler equation for $s, s' = 1, 2$ as

$$\begin{aligned} \left(k^\alpha - g(k, a_s) + k(1 - \delta)\right)^{-\gamma} &= \beta \sum_{s'=1}^{2} p_{s,s'} \left(g(k, a_s)^\alpha - g(k', a_{s'}) + g(k, a_s)(1 - \delta)\right)^{-\gamma} \\ &\quad \times \left[\alpha a_{s'} g(k, a_s)^\alpha + (1 - \delta)\right] \end{aligned}$$

What we want to do is to approximate the unknown $g(\cdot)$ function and then to solve the above function in terms of $g(\cdot)$. Before we do this, we need to learn how to approximate *known* functions.

Chapter 5

Approximation Methods

Notation:

Throughout these notes, vectors are underlined. Matrices are in bold face. $\mathbf{A} = \{a_{i,j}\}$ means $a_{i,j}$ is the $i-j$th element of the matrix $\mathbf{A}$.

Popular Techniques

1. *Desirable criteria.* The approximant should be increasingly accurate and become arbitrarily close as $n \to \infty$.

2. *Spectral methods.* Use basis functions that are nonzero almost everywhere over the entire domain. *Polynomial interpolation* is the most common spectral method and of this strategy, *Chebyshev polynomials* are frequently used. There are good reasons for this.

3. *Finite element methods* use basis functions that are non zero over subintervals of the approximating domain. Polynomial spline approximation is a common finite element method. Of this technique, *cubic splines* are frequently used. These are low degree polynomials over subintervals of the approximating domain that are then glued together. We will not discuss this technique in detail.

5.1 Polynomial approximation (Taylor)

In the one-dimensional case, maybe the first thing that comes to mind is a Taylor series approximation. Say we want to approximate the function $F(x)$ by $\hat{F}(x)$, where

$F(\cdot)$ is known.

$$\begin{aligned} F(x) &\simeq \hat{F}(x) = \sum_{j=0}^{n} b_j \phi_j(x) \\ b_j &= \frac{1}{j!}\frac{\partial^j F}{\partial x^j} \\ \phi_j(x) &= (x - x_0)^j \end{aligned}$$

The Taylor approximation is a polynomial approximation and is an example of the class of *monomials*. That is, the functions $\phi_j(x)$ in the Taylor approximation are powers of x.

5.1.1 Projection (Regression)

Let's use this idea in conjunction with a least-squares criterion and the sequence of monomials, $X_i = \{1, x_i, x_i^2, \cdots x_i^p\}$. We'll apply it to solve the *deterministic* optimal growth model, which consists of an Euler equation and a transition equation,

$$\begin{aligned} c^{-\gamma} &= \beta(c')^{-\gamma}\left(\alpha k'^{\alpha-1} + 1 - \delta\right) \\ k' &= k^\alpha - c + (1-\delta)k \end{aligned}$$

Use the approximation $\ln(\hat{h}(k, \underline{b}) = b_0 + b_1 \ln(k) + b_2 (\ln k)^2$. That is,

$$c \simeq \hat{h}(k, \underline{b}) = \exp\left(b_0 + b_1 \ln(k) + b_2 \left(\ln k\right)^2\right)$$

The least-squares problem is then, to choose $\underline{b}$ to

$$\min \sum_{t=1}^{T} \{R_t\}^2$$

where R_t is the *residual function*,

$$\begin{aligned} R_t &= \hat{h}(k_t, \underline{b})^{-\gamma} - \beta \hat{h}(k_{t+1}, \underline{b})^{-\gamma}\left[\alpha k_{t+1}^{\alpha-1} + 1 - \delta\right] \\ k_{t+1} &= k_t^\alpha - \hat{h}(k_t, \underline{b}) + (1-\delta)k_t. \end{aligned}$$

The procedure:

1. Select a grid $\{k_1, k_2, \ldots, k_n\}$ and initial (guess) values of $\underline{b} = (b_0, b_1, b_2)$.

2. For each $k_i, i = 1, \ldots, n$, construct

$$\begin{aligned} k_i' &= k_i^\alpha - \hat{h}(k_i, \underline{b}) + (1-\delta)k_i \\ \hat{h}(k_i, \underline{b}) &= \exp\left[b_0 + b_1 \ln(k_i) + b_2(\ln(k_i))^2\right] \\ R_i &= R(k_i, \underline{b}) = \hat{h}(k_i, \underline{b})^{-\gamma} - \beta \hat{h}(k_i', \underline{b})^{-\gamma}(\alpha k_i^{'(\alpha-1)} + 1 - \delta) \end{aligned}$$

3. Choose $\underline{b}$ to

$$\min_{\underline{b}} \sum_{i=1}^{n} R_i^2$$

Remarks

1. Instead of generating a "time-series" of R values, we are generating a cross-section of observations. Since the observations are stationary, it doesn't matter. You could have generated a very long time series and do the minimization.

2. This is a nonlinear least squares problem.

3. The following plots show the implied path of log output, log consumption, and log capital, generated by the GAUSS program, LEASTSQ1.PGM. With $n = 100$, we get $b_0 = -0.5151, b_1 = 0.3970, b_2 = 0.0117$.

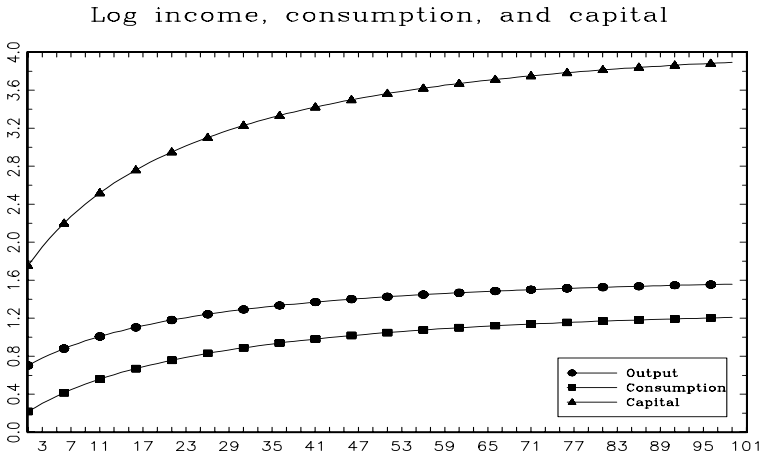

Figure 5.1: Time-paths from deterministic optimal growth model solved by projection on monomials.

```
   new;
#include amoeba.set;
#include nrmin.set;
@-----------------------------------------------------------------------@
@ LEASTSQ1.PGM. @
@ Solve deterministic optimal growth model @
@ by least-squares projection.  CRRA utility, no leisure choice @
@-----------------------------------------------------------------------@
output file=leastsq1.out reset; output on;
format /rd 8,3;
@=======================================================================@
@:::::::::::::  PARAMETER VALUES ::::::::::::::::::::::::::::::::::::::@
@------------- CRRA Utility ------------------------------------------@
alpha=0.4; @ CAPITAL'S SHARE @
delta=0.025; @ QUARTERLY DEPRECIATI ON RATE @
beta=0.99; @ QUARTERLY DISCOUNT FACTOR-NORMALIZED BY EFFICIENCY UNITS @
gam=2.0; @ COEFFICIENT OF RELATIVE RISK AVERSION @
rho=0.95; @ AR(1) COEFFICIENT FOR TECHNOLOGY SHOCKS. @
sigma=0.007; @ Innovation Standard Deviation @

sto1=1-beta*(1-delta); @--------------@
sto2=beta*alpha; @ Steady State @
sto3=1/(alpha-1); @--------------@
Kss=(sto1/sto2)^sto3; @ Steady state capital @
Yss=Kss^alpha; @ Steady state output @
Iss=delta*Kss; @ Steady state investment @
Css=Yss-Iss; @ Steady state consumption @

@-----------------------------------------------------------------@
" rho sigma alpha delta beta gamma";
rho sigma alpha delta beta gam;
"
Steady State Values";
" Capital Output Invest Consumption"; Kss Yss Iss Css;
@=======================================================================@
clear k1,i,R,h0,h1,ktmp0,ktmp1,sto;
n=100;
np=3; @ number of coefficients @
let b= -0.45 .41 .017 ; @ starting values @
t1=100; @ number of "time series" observations to generate @

kl=0.1*kss; ku=1.5*kss;
inc=(ku-kl)/(n-1);
k0=zeros(n,1); k0[1]=kl;
i=2; do while i le n;
k0[i]=k0[i-1]+inc;
i=i+1; endo; @ GRID OF k(i) @
@-----------------------------------------------------------------------@
```

```
@:::::::::::  Options for Minimization Routines ::::::::::::::::::::@
stp1=.1; ftol=1e-4; maxsec=36000; maxit=1000;
varname=zeros(np,1);
let vars=1 2 3;
varname[1]="b_0"; varname[2]="b_1"; varname[3]="b_2";
@-----------------------------------------------------------------------@
"number of nodes:";; n;
pcoef0=strtval1; " STARTING VALUES ";
vn=varname[vars];
omat=(vn (pcoef0[1:np]));
mask=0 1; let fmt[2,3]="-*.*s" 10 8 "*.*lf " 14 4 ;
call printfm(omat,mask,fmt);
output off;
@-------------------------------------------------------@
pcoef1=AMOEBA(&ofn,&strtval1,ftol,stp1,maxit);
output on; format /m1/rd 12,5;
omat=(vn (pcoef1[1:np]));
call printfm(omat,mask,fmt);
"Function value :";;pcoef1[rows(pcoef1)-1];
"Iterations:  ";;pcoef1[rows(pcoef1)];
format /m1/rd 7,3;
output off;
@-------------------------------------------------------@
pcoef1=pcoef1[1:np];
pcoef1,fret,iter,tim = POWELL(pcoef1,ftol,maxsec,maxit,&ofn,&linmib);
output on;format /m1/rd 12,5;
"----------------- POWELL Parameters ----------------------";
omat=(vn (pcoef1[1:np]));
call printfm(omat,mask,fmt);
"Function value :";;fret;
"Iterations:  ";;iter;
format /m1/rd 7,3;
output off;
@-------------------------------------------------------@
pcoef1,fret,iter,tim = DFPMIN(pcoef1,ftol,maxsec,maxit,&ofn,&grdnt,&linmib);
output on; format /m1/rd 12,5;
"----------------- DFP Parameters ----------------------";
omat=(vn (pcoef1[1:np]));
call printfm(omat,mask,fmt);
"Function value :";;fret;
"Iterations:  ";;iter;
format /m1/rd 7,3;
output off;
@------------------- End of optimization part -----------------@
@ Generate time sequence of capital and consumption @
b=pcoef1[1:np];
k=zeros(t1,1);
c=zeros(t1,1);
```

```
k[1]=kl;
t=1; do while t le t1-1;
sto=ln(k[t]); sto=1|sto|sto^2;
h0=exp(b'sto);
c[t]=h0;
k[t+1]=k[t]^(alpha)-c[t]+(1-delta)*k[t];
t=t+1; endo;
c=trimr(c,0,1); k=trimr(k,0,1); y=k^alpha;
@-----------------------------------------------------------@
Graphing code omitted
@===========================================================@
@::::::::::::  Procedures section ::::::::::::::::::::::::::::@
@::::::::::::  Starting Values ::::::::::::::::::::::::::::::@
proc strtval1;local sv; sv=zeros(np,1); sv=b; retp(sv); endp;
@ ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::  @
@ !!  Define the objective function to be minimized:  OFN(b) @
@ To maximize the function, reverse its sign.  @
@:::::::::::::::::::::::  function definition(s) ::::::::::::::::::::  @
proc ofn(x);
local g,vof;
R=zeros(n,1);
i=1; do while i le n;
sto=ln(k0[i]);
ktmp0=1|sto|(sto^2);
h0=exp(x'ktmp0);
k1=k0[i]^(alpha)-h0+(1-delta)*k0[i];
sto=ln(k1);
ktmp1=1|sto|(sto^2);
h1=exp(x'ktmp1);
R[i]=h0^(-gam)-beta*(h1^(-gam))*(alpha*k1^(alpha-1)+1-delta);
i=i+1; endo;
vof=R'R;
retp(vof);
endp;
@===========================================================@
@::  PROCEDURE TO DIFFERENTIATE THE OBJECTIVE FUNCTION ::::::@
@-----------------------------------------------------------@
proc grdnt(&fct,x);
local fct:proc,dt;
dt=gradp(&fct,x);
retp(dt');
endp;
@===========================================================@
```

5.1.2 Drawbacks of Projection on Monomials

We solved the model by projecting consumption onto powers of $\ln(k_t)$. If this works, then fine. But a potential problem in projecting onto a set of monomials $\{1, x, x^2, x^3, \ldots, x^n\}$ is *multicolinearity*. Here is the correlation matrix for $(\ln(k_t), (\ln(k_t))^2, (\ln(k_t))^3, (\ln(k_t))^4)$ generated from the program LEASTSQ1.PGM.

$$\text{Cross Sec} : \begin{pmatrix} 1.000 & 0.994 & 0.980 & 0.960 \\ 0.994 & 1.000 & 0.996 & 0.984 \\ 0.980 & 0.996 & 1.000 & 0.997 \\ 0.960 & 0.984 & 0.997 & 1.000 \end{pmatrix}$$

$$\text{Time series} : \begin{pmatrix} 1.000 & 0.996 & 0.985 & 0.970 \\ 0.996 & 1.000 & 0.997 & 0.988 \\ 0.985 & 0.997 & 1.000 & 0.997 \\ 0.970 & 0.988 & 0.997 & 1.000 \end{pmatrix}$$

This can be especially problematic in more complicated problems because we are solving a *nonlinear* least squares problem to find the $b-$coefficients. The way that applied mathematicians have attacked this problem is to replace the collinear monomials with *orthogonal* polynomials.

5.2 Orthogonal Polynomials

Definition 1 *(Weighting Function). A weighting function $\omega(x)$ on $[a, b]$ is positive almost everywhere on $[a, b]$ and has a finite integral on $[a, b]$.*

Example.

$$\begin{aligned} \omega(x) &= \frac{1}{\sqrt{1-x^2}} \quad \text{on} \quad [-1, 1] \\ \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} dx &= \arcsin|_{-1}^{1} = \pi \end{aligned}$$

Definition 2 *(Inner Product) Consider two functions $f_1(x), f_2(x)$ defined on $[a, b]$. The inner product of f_1 and f_2 with respect to the weighting function $\omega(x)$ is*

$$< f_1, f_2 >= \int_a^b f_1(x) f_2(x) \omega(x) dx$$

Definition 3 *(Orthogonal Polynomials.) The family of polynomials* $\{P_i(x)\}_{i=1}^n$ *is mutually orthogonal with respect to* $\omega(x)$ *iff* $< P_i, P_j >= 0$ *for* $i \neq j$*. If in addition,* $< P_i, P_i >= 1$*, the polynomials are orthonormal.*

Our *objective* here is to get a good approximation where good is defined in the sense of minimizing the sum of square errors. To be explicit, we will obtain the least squares polynomial approximation.

Definition 4 *Let* $F : [a, b] \to \mathcal{R}$ *be the function that we want to approximate. The least squares polynomial approximation to* F *with respect to* $\omega(x)$ *is the* $n-th$ *order polynomial that solves* $\min \int_a^b \left(F(x) - \hat{F}(x)\right)^2 \omega(x)dx$*.*

Remarks

If we are working with a sequence of orthogonal polynomials $\{\phi_i(x)\}_{i=0}^n$, then the problem is to

$$\min_{\{b_i\}_{i=1}^n} \int_a^b \left(F(x) - \sum_{i=0}^n b_i\phi_i(x)\right)^2 \omega(x)dx$$

The first-order conditions are then the least-squares normal equations:

$$\int_a^b \left(F(x) - \sum_{i=0}^n b_i\phi_i(x)\right) \phi_j(x)\omega(x)dx = 0$$

which solves for

$$b_j = \frac{< F, \phi_j >}{< \phi_j\phi_j >}$$

Procedure is as follows.

1. Approximate the function $F(x)$ by $\hat{F}(x)$.

$$\hat{F}(x) = \sum_{j=1}^n b_j\phi_j(x),$$

$b_1, b_2, \ldots, b_n$, are the basis coefficients,
$\phi_1(x), \phi_2(x), \ldots, \phi_n(x)$, are the basis functions,[1]

[1]That is, $\{\phi_j(x)\}_{j=1}^n$ is a basis of a function space. $\hat{F}$ is a member of that space that is described with respect to the basis coefficients, $\{b_j\}_{j=1}^n$.

$x_1, x_2, \ldots, x_n$, are the interpolation nodes,

n is the degree of interpolation.

2. The nodes allow us to set up the system of equations, known as the *interpolation conditions*. For $i = 1, \ldots, n$,

$$F(x_i) \simeq \hat{F}(x_i) = \sum_{j=1}^{n} b_j \phi_j(x_i),$$

That is, set

$$\begin{array}{lcl} F(x_1) & = & b_1\phi_1(x_1) + b_2\phi_2(x_1) + \cdots + b_n\phi_n(x_1) \\ F(x_2) & = & b_1\phi_1(x_2) + b_2\phi_2(x_2) + \cdots + b_n\phi_n(x_2) \\ \vdots & \vdots & \vdots \\ F(x_n) & = & b_1\phi_1(x_n) + b_2\phi_2(x_n) + \cdots + b_n\phi_n(x_n) \end{array}$$

Solve the interpolation conditions for the basis coefficients $\underline{b}$. When the number of basis functions is the same as the degree of interpolation (n), then the solution has a least-squares interpretation.

3. $F(\cdot)$ is known. Do the following function evaluations and assignments.

$y_i = F(x_i)$, $i = 1, \ldots, n$.

$\phi_{ij} = \phi_j(x_i)$.

$e_i = y_i - \sum_{j=1}^{n} b_j \phi_{ij}$, (approximation error.)

In matrix notation, we have

$$\underline{e} = \underline{y} - \mathbf{\Phi}\underline{b} \tag{5.1}$$

where $\underline{e}' = (e_1, e_2, \ldots, e_n)$, $\underline{y}' = (y_1, y_2, \ldots, y_n)$, $\underline{b}' = (b_1, b_2, \ldots, b_n)$, and

$$\mathbf{\Phi} = \begin{pmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n1} & \phi_{n2} & \cdots & \phi_{nn} \end{pmatrix} = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_n(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_n(x_n) \end{pmatrix}$$

Then we can solve for $\underline{b}$ by minimizing the sum of square errors.

$$b = \left(\mathbf{\Phi}'\mathbf{\Phi}\right)^{-1} \mathbf{\Phi}'\underline{y}$$

This is an example of *least squares polynomial approximation.*

5.2.1 Chebyshev Polynomials

The orthogonal polynomials that we will use are *Chebyshev polynomials* and defined for $u \in [-1, 1]$ by

$$\phi_{n+1}(u) = T_n(u) = \cos\left(n \cos^{-1}(u)\right)$$

and they are orthogonal with respect to the weighting function

$$\omega(u) = \frac{1}{\sqrt{1-u^2}}.$$

The Chebyshev polynomials can also be obtained recursively according to

$$\begin{aligned} \phi_1(u) &= T_0(u) = 1 \\ \phi_2(u) &= T_1(u) = u \\ \phi_{n+1}(u) &= T_n(u) = 2uT_{n-1}(u) - T_{n-2}(u), \quad n \geq 2 \end{aligned}$$

The roots of $T_n(u) = 0$ are called the *Chebyshev interpolation nodes*. Call them u_i. For $i = 1, \ldots, n$, they are

$$u_i = -\cos\left(\frac{\pi}{2n}(2i-1)\right) = \cos\left(\frac{\pi}{n}(n-i+0.5)\right)$$

Example. For $n = 10$, we get the following u_i and Δu_i.

u_i	Δu_i
−0.987	
−0.891	0.096
−0.707	0.183
−0.453	0.253
−0.156	0.297
0.156	0.312
0.453	0.297
0.707	0.253
0.891	0.183
0.987	0.096

Relative to a set of uniform nodes, Chebyshev nodes are scrunched together more at the endpoints than in the center.

They have the property that

$$\sum_{i=1}^{n} T_i(u_k)T_j(u_k) = \begin{cases} 0 & i \neq j \\ n & i = j = 0 \\ \frac{n}{2} & i = j > 0 \end{cases}$$

Jackson's theorem gives the theoretical justification for using Chebyshev polynomials. (Fackler-Miranda, p.120).

Theorem 1 *Let F be a continuous function whose first k derivatives over $[-1, 1]$ exist and $\hat{F}_n$ be the $n-$th degree Chebyshev-node polynomial interpolant. Then there exists an $\epsilon < \infty$ such that for all $n \geq 2$,*

$$||F(x) - \hat{F}(x)||_\infty \leq \epsilon \frac{\ln(n)}{n^k}$$

The theorem says that $\hat{F}_n(x)$ becomes arbitrarily close to $F(x)$ as $n \to \infty$ and the approximation error is bounded from above. Chebyshev approximation is therefore shown to be accurate for smooth functions.

5.2.2 Chebyshev Polynomial Interpolation over $[a, b]$

Consider a bounded interval $[a, b]$ over which the function $F(x)$ is to be approximated.

1. For $i = 1, \ldots, n$, construct a set of Chebyshev nodes for $u \in [-1, 1]$.

$$u_i = \cos\left(\frac{\pi}{n}(n - i + 0.5)\right)$$

2. Adjust the nodes to accomodate $x \in [a, b]$.

$$x_i = \frac{a + b}{2} + \frac{b - a}{2} u_i.$$

3. Use the Chebyshev polynomials as the orthogonal polynomials by setting

$$\phi_j(x_i) = T_{j-1}\left(\frac{2(x_i - a)}{b - a} - 1\right) = T_{j-1}(u_i)$$

where for $u \in [-1, 1]$,

$$
\begin{aligned}
T_0(u) &= 1 \\
T_1(u) &= u \\
T_2(u) &= 2u^2 - 1 \\
T_3(u) &= 4u^3 - 3u \\
&\vdots \quad \vdots \quad \vdots \\
T_j(u) &= 2uT_{j-1}(u) - T_{j-2}(u)
\end{aligned}
$$

Chebyshev polynomials with Chebyshev nodes have the following properties.

$$
\begin{aligned}
\phi_{ij} &= \phi_j(x_i) = \cos\left(\frac{\pi(n-i+0.5)(j-1)}{n}\right) \\
\mathbf{\Phi'\Phi} &= n\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1/2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/2 \end{bmatrix}
\end{aligned}
$$

In matrix notation:

$$
X = \begin{bmatrix} T(u_1)' \\ T(u_2)' \\ \vdots \\ T(u_n)' \end{bmatrix} = \begin{bmatrix} T_0(u_1) & \cdots & T_{n-1}(u_1) \\ T_0(u_2) & \cdots & T_{n-1}(u_2) \\ \vdots & \ddots & \vdots \\ T_0(u_n) & \cdots & T_{n-1}(u_n) \end{bmatrix} = \begin{bmatrix} \phi_{11} & \cdots & \phi_{1n} \\ \phi_{21} & \cdots & \phi_{2n} \\ \vdots & \ddots & \vdots \\ \phi_{n1} & \cdots & \phi_{nn} \end{bmatrix} = \mathbf{\Phi}
$$

5.2.3 Examples

1. $F(x) = \exp(-x)$ for $x \in [-2, 2]$. We approximate the function using $n = 2, 3, 4, 5$ nodes, then evaluate the function $y_i = \exp(-x_i)$ and the approximant $\hat{y}_i = \sum_{j=1}^{n} b_j \phi_j(x_i) = \sum_{j=1}^{n} b_j T_{j-1}(2(x_i+2)/4-1)$ over $x_i \in [-2, 2]$ at 100 equally spaced points. The function and approximant are plotted in figure 5.4.[2]

2. Runge's function $F(x) = 1/(1 + 25x^2)$. We approximate this function using $n = 5, 10, 15, 20$ nodes, then evaluate the function $y_i = 1/(1 + 25x_i^2)$ and the approximant $\hat{y}_i = \sum_{j=1}^{n} b_j \phi_j(2(x_i + 2)/4 - 1)$ over $x_i \in [-2, 2]$ at 100 equally spaced points. The function and approximant are plotted in figure 5.5.

[2] cheb_ex1.pgm.

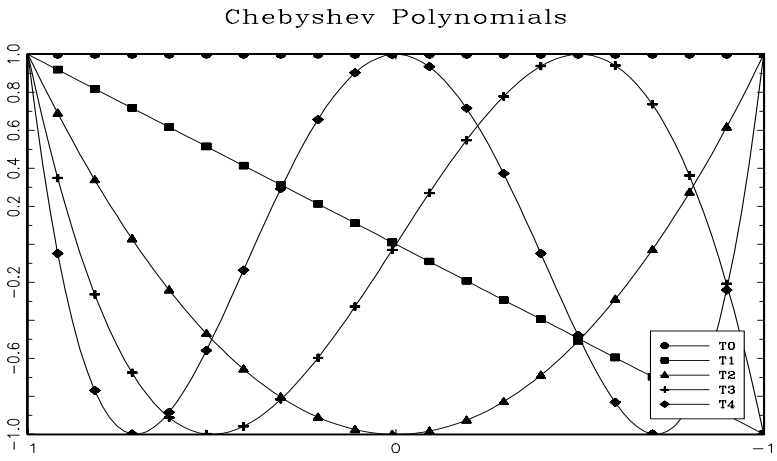

Figure 5.2: Chebyshev Polynomials

3. $F(x) = |x|^{1/2}$. We approximate this function using $n =$ nodes, then evaluate the function $y_i = |x_i|^{1/2}$ and the approximant $\hat{y}_i = \sum_{j=1}^{n} b_j \phi_j(2(x_i + 2)/4 - 1)$ over $x_i \in [-2, 2]$ at 100 equally spaced points. The function and approximant are plotted in figure 5.6.

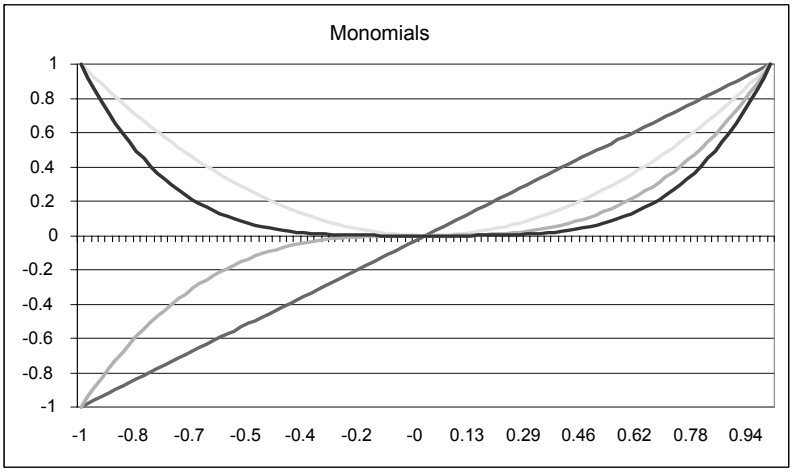

Figure 5.3: Monomials

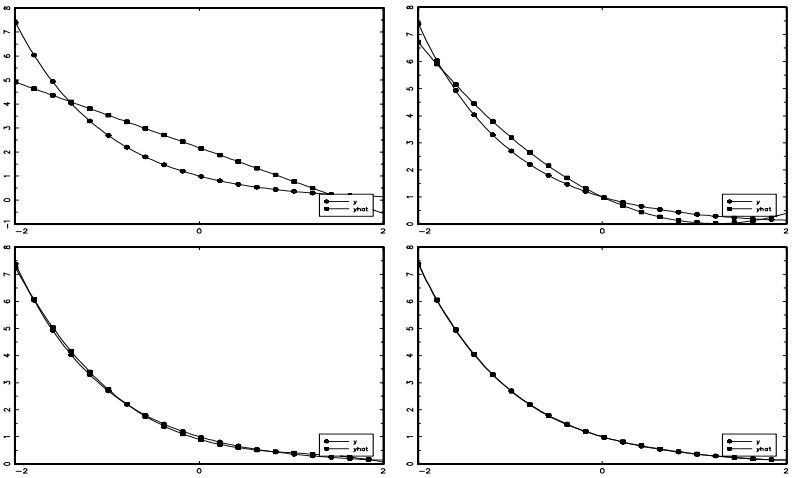

Figure 5.4: Approximating $\exp(-x), x \in [-2, 2]$ using 2, 3, 4, and 5 nodes.

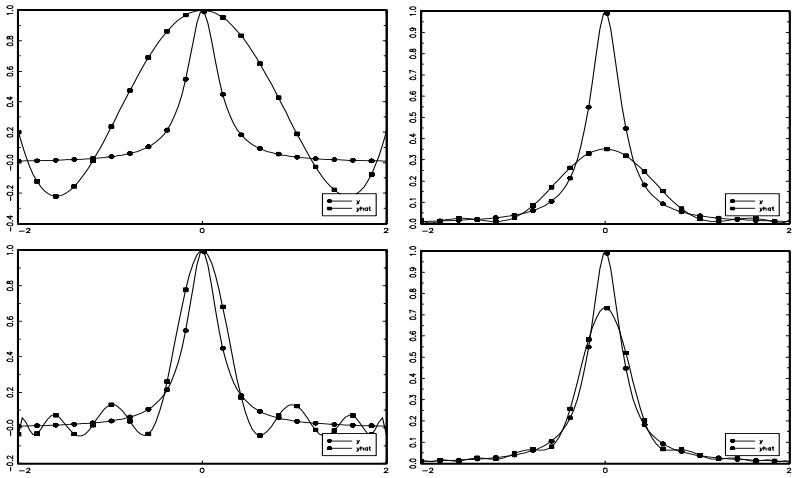

Figure 5.5: Approximating $F(x) = 1/(1+x^2), x \in [-2, 2]$ using 5, 10, 15, and 20 nodes.

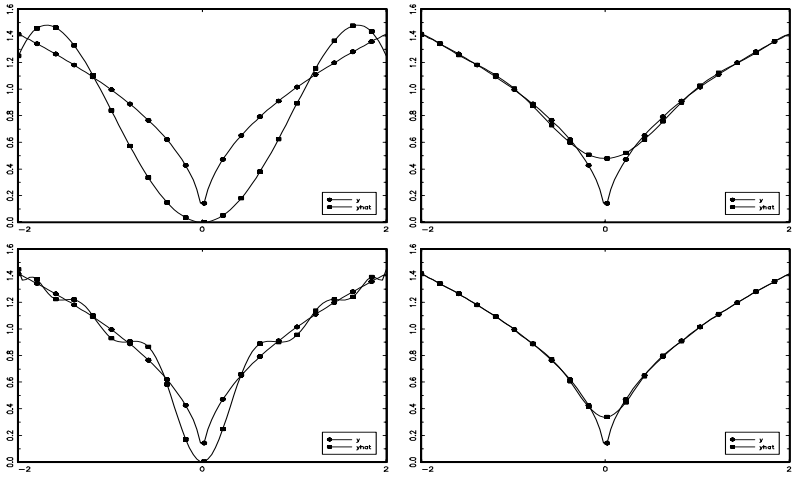

Figure 5.6: Approximating $F(x) = |x|^{1/2}, x \in [-2, 2]$ using 5, 10, 15, and 20 nodes.

```
   new;
@-------------------------------------------------------------------------@
@ Cheb_ex1.PGM. @
@ Function approximation examples with Chebyshev polynomial @
@ approximation @
@-------------------------------------------------------------------------@
output file=Cheb_ex1.out reset; output on;
format /rd 8,3;
@=========================================================================@
#include nodeeq.set; @ Equally spaced nodes @
#include nodecheb.set; @ Chebyshev nodes @
#include chebbas.set; @ Chebyshev polynomial interpolants @
@-------------------------------------------------------------------------@
exno=3; @ Choose example.  1:  exp(-x) @
@ 2:  Runge's function @
@ 3:  |x|^(1/2) @
if exno eq 1;
@---- Example 1 y=F(x)=exp(-x) -----@
n0=2; do while n0 le 5;
xl=-2; xu=2;
u=nodecheb(n0); @ Chebyshev nodes on [-1,1]@
x=(xl+xu)*0.5+0.5*(xu-xl)*u; @ Adjust to [xl,xu] @
Phi=chebbas(u); @ Chebyshev polynomial interpolants @

y=exp(-x);
b=inv(Phi'Phi)*Phi'y;

n1=100; @ points for function evaluation @
u=nodeeq(n1); @ uniform grid @
x=(xl+xu)*0.5+0.5*(xu-xl)*u;
psi=zeros(n0,1); yhat=zeros(n1,1); y=exp(-x);
i=1; do while i le n1; @ Evaluate on u (not x) @
psi[1]=1; psi[2]=u[i];
j=3; do while j le n0;
psi[j]=2*u[i]*psi[j-1]-psi[j-2];
j=j+1; endo;
yhat[i]=psi'*b;
i=i+1; endo;

if n0==2; y1=y; yhat1=yhat; endif;
if n0==3; y2=y; yhat2=yhat; endif;
if n0==4; y3=y; yhat3=yhat; endif;
if n0==5; y4=y; yhat4=yhat; endif;
n0=n0+1; endo;
endif;

if exno eq 2;
@---- Example 2 y=F(x)=1/(1+25x^2) Runge's function @
```

```
iter=0;
n0=5; do while n0 le 20;
iter=iter+1;
xl=-2; xu=2;
u=nodecheb(n0); @ Chebyshev nodes on [-1,1]@
x=(xl+xu)*0.5+0.5*(xu-xl)*u; @ Adjust to [xl,xu] @
Phi=chebbas(u); @ Chebyshev polynomial interpolants @
y=1/(1+25*x^2);
b=inv(Phi'Phi)*Phi'y;

n1=100; @ points for function evaluation @
u=nodeeq(n1); @ uniform grid @
x=(xl+xu)*0.5+0.5*(xu-xl)*u;
psi=zeros(n0,1); yhat=zeros(n1,1);
y=1/(1+25*x^2);
i=1; do while i le n1; @ Evaluate on u (not x) @
psi[1]=1; psi[2]=u[i];
j=3; do while j le n0;
psi[j]=2*u[i]*psi[j-1]-psi[j-2];
j=j+1; endo;
yhat[i]=psi'*b;
i=i+1; endo;

if iter==1; y1=y; yhat1=yhat; endif;
if iter==2; y2=y; yhat2=yhat; endif;
if iter==3; y3=y; yhat3=yhat; endif;
if iter==4; y4=y; yhat4=yhat; endif;
n0=n0+5; endo;
endif;

if exno eq 3;
@---- Example 3 y=F(x)=|x|^1/2 @
iter=0;
n0=5; do while n0 le 20;
iter=iter+1;
xl=-2; xu=2;
u=nodecheb(n0); @ Chebyshev nodes on [-1,1]@
x=(xl+xu)*0.5+0.5*(xu-xl)*u; @ Adjust to [xl,xu] @
Phi=chebbas(u); @ Chebyshev polynomial interpolants @
y=sqrt(abs(x));
b=inv(Phi'Phi)*Phi'y;

n1=100; @ points for function evaluation @
u=nodeeq(n1); @ uniform grid @
x=(xl+xu)*0.5+0.5*(xu-xl)*u;
psi=zeros(n0,1); yhat=zeros(n1,1);
y=sqrt(abs(x));
i=1; do while i le n1; @ Evaluate on u (not x) @
```

```
psi[1]=1; psi[2]=u[i];
j=3; do while j le n0;
psi[j]=2*u[i]*psi[j-1]-psi[j-2];
j=j+1; endo;
yhat[i]=psi'*b;
i=i+1; endo;

if iter==1; y1=y; yhat1=yhat; endif;
if iter==2; y2=y; yhat2=yhat; endif;
if iter==3; y3=y; yhat3=yhat; endif;
if iter==4; y4=y; yhat4=yhat; endif;
n0=n0+5; endo;
endif;
```

5.2.4 Multidimensional Interpolation

Suppose we want to approximate the function $F(x, z)$ where $x \in [x_\ell, x_u]$ and $z \in [z_\ell, z_u]$.

Tensor product basis

One option is to use the *tensor* product basis. To illustrate, we construct a set of $n_x = 2$ nodes for x and a set of $n_z = 2$ nodes for z in exactly the same way as we did for the one-dimensional case. The approximant is

$$\begin{aligned}\hat{F}(x,z) &= \phi_1(x)\phi_1(z)b_{11} + \phi_1(x)\phi_2(z)b_{12} + \phi_2(x)\phi_1(z)b_{21} + \phi_2(x)\phi_2(z)b_{22} \\ &= [\phi_1(x), \phi_2(x)] \otimes [\phi_1(z), \phi_2(z)]\, \underline{b}\end{aligned}$$

Let $\phi^x_{ij} = \phi_j(x_i)$ and $\phi^z_{ij} = \phi_j(z_i)$, and $\underline{b} = (b_{11}, b_{12}, b_{21}, b_{22})'$. The *interpolation conditions* are

$$\begin{aligned}y_{11} &= F(x_1, z_1) = [(\phi^x_{11}, \phi^x_{21}) \otimes (\phi^z_{11}, \phi^z_{21})]\, \underline{b} \\ y_{12} &= F(x_1, z_2) = [(\phi^x_{11}, \phi^x_{21}) \otimes (\phi^z_{12}, \phi^z_{22})]\, \underline{b} \\ y_{21} &= F(x_2, z_1) = [(\phi^x_{12}, \phi^x_{22}) \otimes (\phi^z_{11}, \phi^z_{21})]\, \underline{b} \\ y_{22} &= F(x_2, z_2) = [(\phi^x_{12}, \phi^x_{22}) \otimes (\phi^z_{12}, \phi^z_{22})]\, \underline{b}\end{aligned}$$

In matrix form, the interpolation conditions are

$$\underline{y} = \mathbf{\Phi}\underline{b}$$

and solve $\underline{b} = (\mathbf{\Phi}'\mathbf{\Phi})^{-1}\mathbf{\Phi}'\underline{y}$ where[3]

$$\begin{aligned}
\mathbf{\Phi} &= \mathbf{\Phi}^x \otimes \mathbf{\Phi}^z \\
\mathbf{\Phi}^x &= \begin{pmatrix} \phi^x_{11} & \cdots & \phi^x_{1n_x} \\ \vdots & & \vdots \\ \phi^x_{n_x1} & \cdots & \phi^x_{n_xn_x} \end{pmatrix} = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_{n_x}(x_1) \\ \vdots & & \vdots \\ \phi_1(x_{n_x}) & \cdots & \phi_{n_x}(x_{n_x}) \end{pmatrix} \\
\mathbf{\Phi}^z &= \begin{pmatrix} \phi^z_{11} & \cdots & \phi^z_{1n_z} \\ \vdots & & \vdots \\ \phi^z_{n_z1} & \cdots & \phi^z_{n_zn_z} \end{pmatrix} = \begin{pmatrix} \phi_1(z_1) & \cdots & \phi_{n_z}(z_1) \\ \vdots & & \vdots \\ \phi_1(z_{n_z}) & \cdots & \phi_{n_z}(z_{n_z}) \end{pmatrix}
\end{aligned}$$

If we use Chebyshev polynomials, we set

$$\phi_j(z_i) = T_{j-1}\left(2\left(\frac{z_i - z_\ell}{z_u - z_\ell}\right) - 1\right)$$

and

$$\phi_j(x_i) = T_{j-1}\left(2\left(\frac{x_i - x_\ell}{x_u - x_\ell}\right) - 1\right)$$

The curse of dimensionality. Using the tensor product basis is straightforward. This is a good feature. The bad feature is that the matrix $\mathbf{\Phi}$ gets big real fast. This is a bad feature. The increase in the number of nodes to evaluate and the number of basis coefficients that comes with the multivariate problem is what we mean by the curse of dimensionality.

Polynomial basis

Collard suggests a reduction in computational burden with no *asymptotic loss of accuracy* by using the *implicit polynomial basis*. The idea of the polynomial basis over the *monomials* can be seen from a second-order Taylor expansion.

$$F(x,z) = F(0,0) + F_1x + F_2z + \frac{1}{2}\left(F_{11}x^2 + F_{22}z^2 + 2F_{12}xz\right)$$

This suggests using as a basis,

$$\{1, x, z, x^2, z^2, xz\}$$

[3]`Notes: Storing a two-dimensional array in a vector. For` $i = 1,\ldots,n_i$ `and` $j = 1,\ldots,n_j$, $b(i,j) = \tilde{b}(\ell)$ `where` $\ell = j+(i-1)*n_j$. `For` $i = 1,\ldots,n_i, j = 1,\ldots,n_j, m = 1,\ldots,n_m$, `and` $b(i,j,m) = \tilde{b}(\ell)$ `where` $\ell = m + (j-1)*n_m + (i-1)*n_m*n_j$.

as opposed to the *tensor product basis*,

$$\{1, z, z^2, x, xz, xz^2, x^2, x^2z, x^2z^2\}$$

Collard's suggestion is based on Taylor's theorem which tells us that asymptotically, we get the same degree of accuracy from the polynomial basis as we get from the tensor product basis.

What is an algorithm to obtain the polynomial basis? Notice that for $n = 3$, the tensor product basis can be represented as

$$\begin{array}{cccc} 1 & x & x^2 & x^3 \\ z & zx & zx^2 & zx^3 \\ z^2 & z^2x & z^2x^2 & z^2x^3 \\ z^3 & z^3x & z^3x^2 & z^3x^3 \end{array}$$

and for the polynomial basis,

$$\begin{array}{cccc} 1 & x & x^2 & x^3 \\ z & zx & zx^2 & \\ z^2 & z^2x & & \\ z^3 & & & \end{array}$$

Product terms involve powers $x^i z^j$ such that $i + j \leq n$. Thus, using Chebyshev polynomials, we will want to form the basis as

$$\{1, T_1^x, \ldots, T_n^x, T_1^z, \ldots, T_n^z, (\cdots)\}$$

where $(\cdots) = \{\{T_i^x T_j^z\}_{j=1}^{n-i}\}_{i=1}^{n}$

Approximating the Cobb-Douglas Production Function

Let's try out the technique using the tensor product basis for approximating the Cobb-Douglas function $F(k, \ell) = k^{0.4}\ell^{0.6}$. In the program Cheb_ex2.pgm, we

1. Obtain $u_1, \ldots, u_n$: a set of Chebyshev nodes on $[-1, 1]$.
2. Adjust for nodes for capital and labor. Let $k_l = 20, k_u = 100, \ell_l = 0.1, \ell_u = 2$.

$$\begin{aligned} k_i &= \frac{k_l + k_u}{2} + \frac{k_u - k_l}{2} u_i \\ \ell_i &= \frac{\ell_l + \ell_u}{2} + \frac{\ell_u - \ell_l}{2} u_i \end{aligned}$$

3. Form $\mathbf{\Phi}^k$ and $\mathbf{\Phi}^\ell$.

4. Solve for $\underline{b}$.

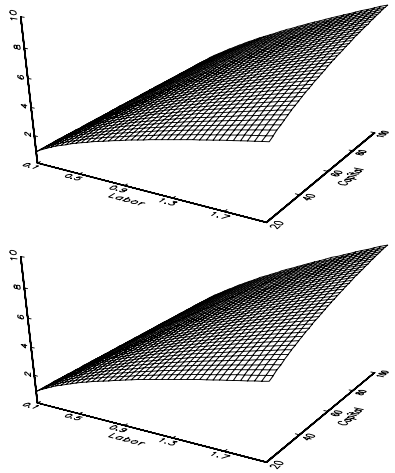

Figure 5.7: Approximated and true Cobb-Douglas function $F(k, \ell) = k^{0.4}\ell^{0.6}$

```
   new;
@-----------------------------------------------------------------------@
@ Cheb_ex2.PGM. @
@ Approximate the Cobb-Douglas production function @
@ with Chebyshev polynomial approximation and the tensor product basis @
@-----------------------------------------------------------------------@
output file=Cheb_ex2.out reset; output on;
format /rd 8,3;
@=======================================================================@
#include nodeeq.set; @ Equally spaced nodes @
#include nodecheb.set; @ Chebyshev nodes @
#include chebbas.set; @ Chebyshev polynomial interpolants @
@-----------------------------------------------------------------------@
n0=10;
u=nodecheb(n0); @ Chebyshev nodes on [-1,1]@
k_l=20; k_u=100; L_l=0.1; L_u=2;
k=(k_l+k_u)*0.5+0.5*(k_u-k_l)*u; @ Adjust to [k_l,k_u] @
L=(L_l+L_u)*0.5+0.5*(L_u-L_l)*u; @ Adjust to [L_l,L_u] @
Phi=chebbas(u); @ Chebyshev polynomial interpolants @
BP=Phi.*.Phi;
y=(k^0.4).*.(L^0.6);
b=inv(BP'BP)*BP'y;
n1=40; @ points for function evaluation @
u=nodeeq(n1); @ uniform grid @
k=(k_l+k_u)*0.5+0.5*(k_u-k_l)*u; @ Adjust to [k_l,k_u] @
L=(L_l+L_u)*0.5+0.5*(L_u-L_l)*u; @ Adjust to [L_l,L_u] @
@ Evaluation @
psi_k=zeros(1,n0); psi_l=zeros(1,n0);
yhat=zeros(n1*n1,1); y=(k^0.4).*.(L^0.6);
i=1; do while i le n1; @ Evaluate on u (not x) @
j=1; do while j le n1;
psi_k[1,1]=1; psi_k[1,2]=u[i];
psi_l[1,1]=1; psi_l[1,2]=u[j];
m=3; do while m le n0;
psi_k[1,m]=2*u[i]*psi_k[1,m-1]-psi_k[1,m-2];
psi_l[1,m]=2*u[j]*psi_l[1,m-1]-psi_l[1,m-2];
m=m+1; endo;
psi=psi_k.*.psi_l;
yhat[j+(i-1)*n1]=psi*b;
j=j+1; endo;
i=i+1; endo;
zhat=zeros(n1,n1); z=zeros(n1,n1);
i=1; do while i le n1;
j=1; do while j le n1;
zhat[i,j]=yhat[j+(i-1)*n1];
z[i,j]=k[i]^0.4*L[j]^0.6;
j=j+1; endo;
i=i+1; endo;
```

```
@-------------------------------------------------------@
library pgraph;
graphset;
_pframe = 0; _psurf =  0,0 ; _pdate = ""; _pzclr = 10;
volume(2,2,1.5);
view(4,-4,3);
xlabel("Labor"); ylabel("Capital");
begwind; window(2,1,0); surface(L',k,zhat);
nextwind; surface(L',k,z);endwind;
@=======================================================@
```

5.3 Essential Topics, not covered

1. Numerical differentiation and integration
2. Solving systems of nonlinear equations

5.4 Inessential topics, not covered

Pertubation methods (second order Taylor expansion).

Chapter 6

Solving the Deterministic Growth Model by Collocation

6.1 Minimum Weighted Residual Methods

Our method of choice is called *collocation*. This is but one of a general class of approximation methods called *minimum weighted residual methods*. The objective is to find that unknown function $g(\cdot)$ that solves the functional equation

$$R(k; g) = 0 \quad \forall k \in [k_\ell, k_u]$$

This is called the *residual* function. The strategy is to *parameterize* an approximant to $g(\cdot)$ with a finite set of coefficients $\underline{b}$. Call the approximant $\hat{g}(\underline{b})$. We choose $\underline{b}$ to make $R(k; \hat{g}(\underline{b}))$ small in the sense that for $n = \dim(\underline{b})$ and $i = 1, \ldots, n$, $\underline{b}^*$ is the solution to

$$\int R(k; \hat{g}(\underline{b}))\omega_i(k)dk = 0 \tag{6.1}$$

for $i = 1, \ldots, n$. (Note: this $\omega_i(k)$ function is *different* from the weighting functions of the previous chapter.) (6.1) is a system of n (usually nonlinear) equations in n unknowns. At this point, we are *confronted by two choices*. They are,

1. Choice of approximant.
   - Finite element methods (e.g., cubic spline).
   - Spectral methods (e.g., Chebyshev polynomials).
2. Choice of weighting function $\omega_i(k)$.

- *Least squares.* Set the weighting function to be,

$$\omega_i(k) = \frac{\partial R(k; g(\underline{b}))}{\partial b_i}$$

Then the criterion is just the least-squares normal equations. We can rewrite the problem as

$$\underline{b}^* = \text{argmin} \left\{ \int R^2(k; \hat{g}(\underline{b}))dk \right\}$$

- *Collocation.* Set the weighting function to be

$$\omega_i(k) = \begin{cases} 1 & \text{if} \quad k = k_i \\ 0 & \text{otherwise} \end{cases}$$

Here $\omega_i(k)$ is the *Dirac delta* function. The residuals are set identically to zero at the collocation nodes $\{k_1, \ldots, k_n\}$ but nothing is imposed outside of the nodes.

Under collocation, we don't actually have to compute the integral (6.1) because the weighting function reduces it to a system of n equations in n unknowns.

- *Galerkin.* Set

$$\hat{g}(\underline{b}) = \sum_{j=1}^{n} b_j \phi_i(k_j)$$

$$\omega_i(k) = \phi_i(k)$$

Choose $\underline{b}$ to set

$$\int R(k, \hat{g}(\underline{b}))\phi_i(k)dk = 0$$

Remarks.
If the residual function is *nonlinear*, the integrals cannont be solved explicitly and we must resort to numerical quadrature. This is true also for the least squares method, but as mentioned above, under collocation we do not have to compute the integral. Because collocation is simpler in this regards, we follow Fackler and Miranda in emphasizing collocation.

6.2 The Collocation Method

The collocation method is a way to solve *functional equations*. The strategy is based on a generalization of function approximation methods.

The basic idea. Let $R : \mathcal{R}^2 \to \mathcal{R}$ be a known function, and $g : [a, b] \to \mathcal{R}$ be an *unknown* function that we would like to know. We know that

$$R(k, g(k)) = 0, \quad k \in [a, b]$$

and we want to find the unknown function g. The *strategy* then is to construct the *collocation* nodes $\{k_1, \ldots, k_n\}$, approximate the function by

$$\hat{g}(k_i) = \sum_{j=1}^{n} b_j \phi_j(k_i) = \sum_{j=1}^{n} b_j T_{j-1}\left(2\left(\frac{k_i - k_\ell}{k_u - k_\ell}\right) - 1\right).$$

Now solve the n nonlinear equations

$$\begin{aligned} R\left(x_1, \sum_{j=1}^{n} b_j T_{j-1}\left(2\left(\frac{k_1 - k_\ell}{k_u - k_\ell}\right) - 1\right)\right) &= 0 \\ &\vdots \\ R\left(x_n, \sum_{j=1}^{n} b_j T_{j-1}\left(2\left(\frac{k_n - k_\ell}{k_u - k_\ell}\right) - 1\right)\right) &= 0 \end{aligned}$$

for $\{b_1, \ldots, b_n\}$.

Define the *residual function* to be

$$r(k) = R(k, \hat{g}(k))$$

The approximant $\hat{g}(k)$ is exact when evaluated on the collocation nodes but will not be exact when evaluated at points in between. One assesses the accuracy of the approximant by looking at the residual function. We can compute its maximum absolute value over the domain and if it is insufficiently small, we might want to increase the degree of interpolation (number of nodes).

6.3 Value function iteration

Let's put the collocation idea to work by solving the *deterministic* optimal growth problem by *parametrically* specifiying the value function as a polynomial and then iterating on the value function.

Why does function iteration work?

We know that by the contraction mapping theorem that we are looking for the function V that solves

$$V = TV$$

where the operator $TV = \max\{U(c) + \beta V(k^\alpha - c + k(1-\delta)\}$. If T is a contraction mapping (this has to be verified, but we know it to be the case in the optimal growth problem) then we know that for any sequence $\{V_n\}_{n=0}^{\infty}$,

$$V_{n+1} = TV_n$$

converges to $V = TV$. This theoretical result tells us why iterating on the value function works.

In the collocation method, we will approximate the Bellman equation by Chebyshev polynomial interpolation. Then the Bellman equation is replaced by a system of n nonlinear equations in the n unknown basis coefficients $\{b_1, \ldots, b_n\}$, for which we solve.

Solve the deterministic optimal growth model.

We state the *problem* as,

$$\begin{aligned} V(k) &= \max_{c \in \mathcal{F}} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta V(k') \right\} \\ \mathcal{F} &\in [0, k^\alpha] \\ k' &= g(k,c) = k^\alpha - c + k(1-\delta) \end{aligned}$$

which we rewrite as

$$V(k) = \max_{c \in \mathcal{F}} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta V(k^\alpha - c + k(1-\delta)) \right\} \tag{6.2}$$

Here is the procedure we use to solve (6.2) by parametric value function iteration.

1. Obtain the Chebyshev nodes $\{u_1, \ldots, u_n\}$ on $[-1, 1]$. Map into the capital stock $\{k_1, \ldots, k_n\}$ on $[k_l, k_u]$,

$$k_i = \frac{k_u + k_l}{2} + \frac{k_u - k_l}{2} u_i$$

The $\{k_1, \ldots, k_n\}$ are now referred to as the *collocation nodes*.

2. Use the *approximant*: $\hat{V}(k) = \sum_{j=1}^{n} b_j \phi_j(k) = \sum_{j=1}^{n} b_j T_{j-1}\left(2\left(\frac{k-k_\ell}{k_u-k_\ell}\right)-1\right)$.

3. For $i = 1, \ldots, n$, solve for the *collocation coefficients* in the system of equations,

$$\sum_{j=1}^{n} b_j \phi_j(k_i) = \max_{c \in \mathcal{F}} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta \sum_{j=1}^{n} b_j \phi_j \left(k_i^\alpha - c + k_i(1-\delta)\right) \right\} \tag{6.3}$$

How to do step 3?

(a) Begin with guess values of collocation coefficients $\underline{b}^0 = \{b_1^0, \ldots, b_n^0\}$.

(b) For the grid $\{u_1, \ldots, u_n\}$ and associated $\{k_1, \ldots, k_n\} \in [k_\ell, k_u]$, construct the Chebyshev basis

$$\mathbf{\Phi} = \{\phi_j(k_i)\} = \{T_{j-1}(u_i)\}$$

(c) Use to evaluate the *RHS* of (6.3).
For each $k_i, i = 1, \ldots, n$,

i. Construct a grid of feasible consumption values, $m = 1, \ldots, n$,

$$\{c_{i,m}\}_{m=1}^{n} \to \{c_{i,1}, \ldots, c_{i,m}\} \in [c_{i,\ell}, c_{i,u}] = [0, k_i^\alpha]$$

ii. Associated grid of next period capital

$$k'_{i,m} = k_i^\alpha + (1-\delta)k_i - c_{i,m}$$

iii. Associated grid of rhs evaluations

$$\hat{G}'_{i,m} = \frac{c_{i,m}^{1-\gamma}}{1-\gamma} + \beta \sum_{j=1}^{n} b_j \phi_j(k'_{i,m}) = \frac{c_{i,m}^{1-\gamma}}{1-\gamma} + \beta \sum_{j=1}^{n} b_j T_{j-1}\left(2\left(\frac{k'_{i,m}-k_\ell}{k_u-k_\ell}\right)-1\right)$$

In vector form, stack across $m = 1, \ldots, n$.

$$\underline{\hat{G}}'_i = \frac{\underline{c}_i^{1-\gamma}}{1-\gamma} + \beta \mathbf{\Phi}_p \underline{b}^0$$

where

$$\mathbf{\Phi}_p = \{\phi_j(k'_{i,m}\} = \left\{ T_{j-1}\left(2\left(\frac{k'_{i,m}-k_\ell}{k_u-k_\ell}\right)-1\right)\right\}$$

iv. Find the largest element of $\hat{G}'_{i,m}$. Call it $\hat{G}'_{i,m^*}$. Save it as $G'_i = \hat{G}'_{i,m^*}$.

(d) Stack into a vector $\underline{G}' = \{G'_1, \ldots, G'_n\}$.

(e) Update the coefficients.[1]

$$\underline{b}^{\ell+1} = \mathbf{\Phi}^{-1}\underline{G}'$$

Repeat until convergence

We do this in COL_DET1.PGM. (The graphing code has been omitted.) Here are the basis coefficients obtained with $n = 16$.

b_0	-30.194
b_2	5.109
b_3	-1.428
b_4	0.533
b_5	-0.232
b_6	0.108
b_7	-0.050
b_8	0.024
b_9	-0.013
b_{10}	0.007
b_{11}	-0.003
b_{12}	0.002
b_{13}	-0.001
b_{14}	0.000
b_{15}	-0.000
b_{16}	0.000

```
new;
@-----------------------------------------------------------------------@
@ COL_DET1.PGM @
@ Collocation method solution of deterministic optimal growth model @
@ by value function iteration @
@ Function approximation by Chebyshev polynomial @
@-----------------------------------------------------------------------@
output file=Col_det1.out reset; output on;
format /rd 8,3;
@=======================================================================@
#include nodeeq.set; @ Equally spaced nodes @
#include nodecheb.set; @ Chebyshev nodes @
#include chebbas.set; @ Chebyshev polynomial interpolants @
#include chebeval.set; @ Evaluation of Cheb.  approximated function @
@=======================================================================@
```

[1]We set $V_\ell = \mathbf{\Phi}\underline{b}^\ell$, and $V_{\ell+1} = TV_\ell$, where $V_\ell = \underline{G}'$. Now the updating equation follows directly.

```
@:::::::::::::  PARAMETER VALUES ::::::::::::::::::::::::::::::::::@
@------------- CRRA Utility ------------------------------------@
alpha=0.4; @ CAPITAL'S SHARE @
delta=0.025; @ QUARTERLY DEPRECIATION RATE @
beta=0.99; @ QUARTERLY DISCOUNT FACTOR-NORMALIZED BY EFFICIENCY UNITS @
gam=2.0; @ COEFFICIENT OF RELATIVE RISK AVERSION @
sto1=1-beta*(1-delta); @--------------@
sto2=beta*alpha; @ Steady State @
sto3=1/(alpha-1); @--------------@
Kss=(sto1/sto2)^sto3; @ Steady state capital @
@-----------------------------------------------------------------@
crit=1e-6; @ convergence criteria @
n0=16; @ degree of interpoloation @
kl=Kss*0.1; ku=Kss*1.4;
let sto= -2.74569510e+01 5.64583848e+00 -2.47211775e+00 6.63931266e-01 ;
b0=zeros(n0,1); b0[1:rows(sto)]=sto; @ Starting values @
output off;
iter=0; @-----------------Update basis coefficients ------@
retry:
iter=iter+1;
u=nodecheb(n0); @ Chebyshev nodes on [-1,1]@
Phi=chebbas(u); @ Chebyshev polynomial interpolants @
k=(kl+ku)*0.5+0.5*(ku-kl)*u; @ Adjust to [kl,ku] @
Vb0=zeros(n0,1); c=zeros(n0,1); kp=zeros(n0,1);
i=1; do while i le n0;
cl=1e-6; cu=k[i]^(alpha);
cgrid=(cl+cu)*0.5+0.5*(cu-cl)*u; @ grid c[i,m], m=1,...,n0 @
kpgrid=k[i]^(alpha)+(1-delta)*k[i]-cgrid; @ k' grid m=1,...,n0 @
Phip=chebbas(2*(kpgrid-kl)/(ku-kl)-1);
ghat=((1-gam)^(-1))*cgrid^(1-gam)+beta*Phip*b0;
mstar=maxindc(ghat);
Vb0[i]=ghat[mstar];
c[i]=cgrid[mstar];
kp[i]=k[i]^(alpha)+(1-delta)*k[i]-c[i];
i=i+1; endo;
b1=inv(Phi)*Vb0; Vb1=Phi*b0;
d=Vb1-vb0;
b1'; "iter";;iter;
if sqrt(d'd) gt crit; b0=b1; goto retry; endif;
output on;
b=b1; @ FINAL VALUES @
"collocation coefficients "; b;
@---------------- Study the results -----------------@
thetak=inv(phi'phi)*phi'kp; @ approximate optimal investment (k') @
n=100; @ points for function evaluation @
u=nodeeq(n); @ uniform grid @
Vhat=chebeval(b,n0,u); @ The value function @
k=(kl+ku)*0.5+0.5*(ku-kl)*u; @ equally spaced grid of k @
```

```
c=zeros(n,1); kp=zeros(n,1);
i=1; do while i le n;
kp[i]=chebeval(thetak,n0,2*(k[i]-kl)/(ku-kl)-1);
i=i+1; endo;
c=k^(alpha)-kp+k*(1-delta);
output off;
@GRAPHINC CODE OMITTED@
```

What to do with the results?

1. *Plot the value function.* Select a uniform grid of N feasible nodes for k_i, and plot

$$\hat{V}(k_i) = \sum_{j=1}^{N} b_j T_{j-1}\left(2\left(\frac{k_i - k_\ell}{k_u - k_\ell}\right) - 1\right) = \mathbf{\Phi}_{[i,\cdot]}\underline{b}$$

against k_i.

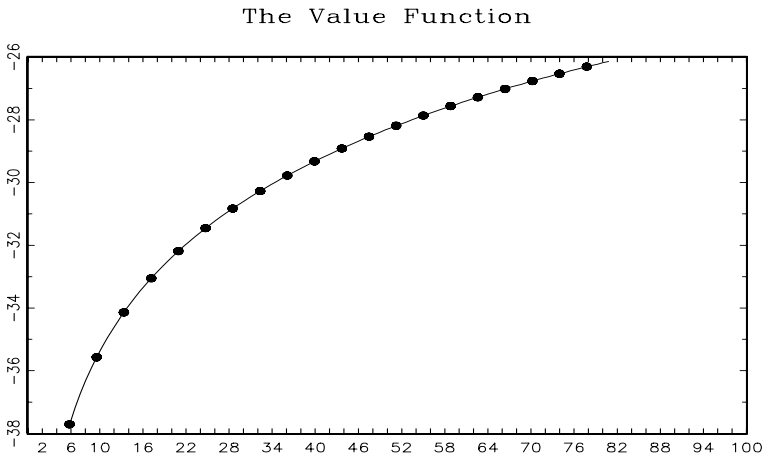

Figure 6.1: Value Function: Deterministic Optimal Growth

2. *Plot c against k and k' against k.* For $i = 1, \ldots, n*$, set up a grid of feasible $c_{i,m} \in [0, k_i^\alpha]$ and find

$$c_{i,m^*} = \text{argmax}\left\{\frac{c_{i,m}^{1-\gamma}}{1-\gamma} + \beta\mathbf{\Phi}_{p[m,\cdot]}\underline{b}\right\}$$

Then obtain

$$k_i' = k_i^\alpha - c_{i,m^*} + k_i(1-\delta).$$

Now we have the pairs (k_i', k_i) and the pairs (k_i, c_{i,m^*}) available to plot.

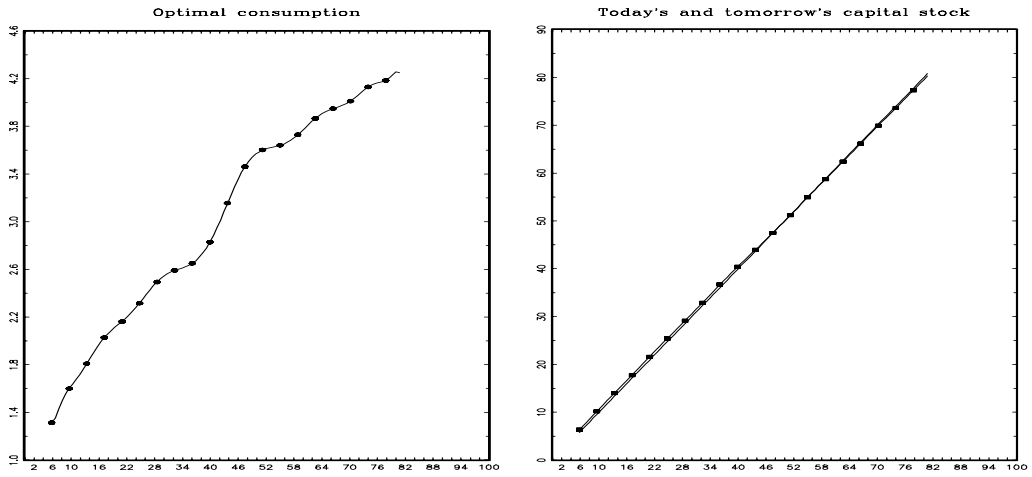

Figure 6.2: Optimal consumption, k_{t+1} and k_t.

Policy Function Iteration

Instead of approximating and iterating on the value function, we now approximate the unknown optimal policy (investment or consumption) function and iterate on it as instructed by the Euler equation.
The *Euler equation* is

$$c^{-\gamma} = (c')^{-\gamma}\left(\alpha(k')^{\alpha-1} + 1 - \delta\right)$$

We'll solve this by approximating the optimal consumption rule,

$$c = \exp[g(k)].$$

(Exponentiation keeps consumption positive.) Instead of approximating the value function, we approximate optimal consumption by $c_i \simeq \exp[\hat{g}(k_i)] = \exp\left(\sum_{j=1}^{n} b_j \phi_j(k_i)\right)$. Then next period comes from the *transition equation*,

$$k' = k^{\alpha} - \exp[g(k)] + k(1-\delta)$$

Solution Method

We can use a nonlinear equation solver to obtain $\underline{b}$, or update as we did for the value function. Let's update.

1. Obtain Chebyshev nodes, adjust for capital stocks and Chebyshev polynomial basis.

$$\{u_1, \ldots, u_n\} \rightarrow \{k_1, \ldots, k_n\} \in [k_\ell, k_u]$$

$$\mathbf{\Phi} = \{\phi_j(k_i)\} = \left\{T_{j-1}\left(2\left(\frac{k_i - k_\ell}{k_u - k_\ell}\right) - 1\right)\right\} = \{T_{j-1}(u_i)\}$$

2. For each $k_i, i = 1, \ldots, n$ compute

$$\begin{aligned}
g_i &= \sum_{j=1}^{n} b_j T_{j-1}\left(2\left(\frac{k_i - k_\ell}{k_u - k_\ell}\right) - 1\right) \\
c_i &= \exp(g_i) \\
k_i' &= k_i^\alpha + (1-\delta)k_i - c_i \\
g_i' &= \sum_{j=1}^{n} b_j T_{j-1}\left(2\left(\frac{k_i' - k_\ell}{k_u - k_\ell}\right) - 1\right) \\
c_i' &= \exp(g_i') \\
R_i &= c_i'^{-\gamma}\left[\alpha k_i'^{\alpha-1} + 1 - \delta\right]
\end{aligned}$$

3. Stack the RHS evaluations R_i into a vector, $\underline{R}$.

4. Update the coefficients

$$\underline{b}^{\ell+1} = \mathbf{\Phi}^{-1}\left(\frac{\ln \underline{R}}{-\gamma}\right)$$

We do this in COL_DET3.PGM. The shape of optimal consumption differs quite a bit from that obtained under value function iteration.

```
new;
@-----------------------------------------------------------------------@
@ COL_DET3.PGM @
@ Collocation method solution of deterministic optimal growth model @
@ by policy function approach @
@-----------------------------------------------------------------------@
output file=col_det3.out reset ; output on; format /rd 8,3;
@=======================================================================@
#include nodeeq.set; @ Equally spaced nodes @
#include nodecheb.set; @ Chebyshev nodes @
#include chebbas.set; @ Chebyshev polynomial interpolants @
#include chebeval.set; @ Evaluation of Cheb.  approximated function @
#include nlsys.set; @ Systems of Nonlinear equations solver @
@=======================================================================@
@:::::::::::::  PARAMETER VALUES ::::::::::::::::::::::::::::::::::::::@
@------------- CRRA Utility ------------------------------------------@
```

```
alpha=0.4; @ CAPITAL'S SHARE @
delta=0.025; @ QUARTERLY DEPRECIATION RATE @
beta=0.99; @ QUARTERLY DISCOUNT FACTOR-NORMALIZED BY EFFICIENCY UNITS @
gam=2.0; @ COEFFICIENT OF RELATIVE RISK AVERSION @
sto1=1-beta*(1-delta); @--------------@
sto2=beta*alpha; @ Steady State @
sto3=1/(alpha-1); @--------------@
Kss=(sto1/sto2)^sto3; @ Steady state capital @
@-------------------------------------------------------------@
clear c,kp,kpp,Vb0,up,Phip,cp,stox,u,k,kpu,kpl,i;
crit=1e-6;
clear sto,ghat,ghatp;

   n0=20; @ Degree of interpolation @
kl=Kss*.1; ku=Kss*1.4;
u=nodecheb(n0); Phi=chebbas(u); k=(kl+ku)*0.5+0.5*(ku-kl)*u;
b0=zeros(n0,1);
let sto= 1.26000141e+00 4.25820838e-01 -1.44585625e-01 5.42740266e-02 ;
b0[1:rows(sto)]=sto;
output off;

   iter=0; retry:  iter=iter+1;
u=nodecheb(n0);
k=(kl+ku)*0.5+0.5*(ku-kl)*u;
Phi=chebbas(u); g=Phi*b0; c=exp(g); kp=k^(alpha)+(1-delta)*k-c;
Phip=chebbas(2*(kp-kl)/(ku-kl)-1); gp=Phip*b0; cp=exp(gp);
kpp=kp^(alpha)+(1-delta)*kp-cp;
RHS1=cp.*(beta*(alpha*k^(alpha-1)+1-delta))^(-1/gam);
b1=inv(Phi)*ln(RHS1);
LHS1=c;
d=abs(LHS1-RHS1);
"iter";;iter; b1';
if sqrt(d'd) gt crit; b0=b1; goto retry; endif;
output on;
b=b1; @ FINAL VALUES @
"collocation coefficients "; format /re 18,8; b; format /rd 9,3;
@---------------- Study the results -----------------@
n=100; @ points for function evaluation @
u=nodeeq(n); @ uniform grid @
k=(kl+ku)*0.5+0.5*(ku-kl)*u; @ equally spaced grid of k @
ghat=chebeval(b,n0,u);
c=exp(ghat); @ optimal investment policy function @
kp=k^(alpha)-c+k*(1-delta);
```

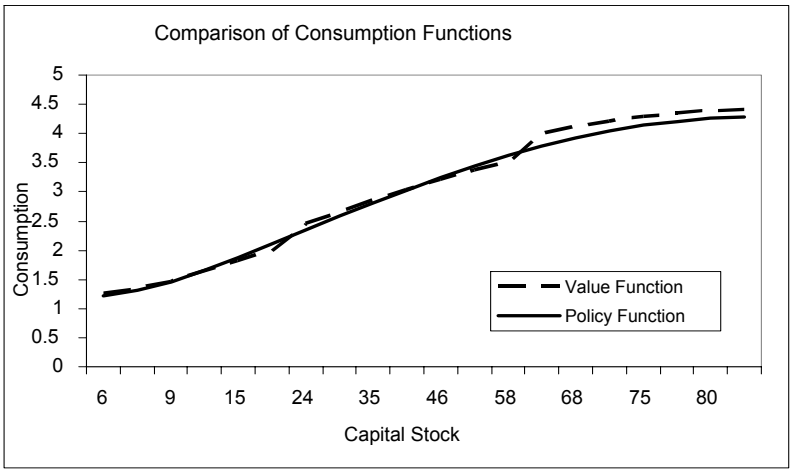

Figure 6.3: Comparison of consumption policy function from Value Function iteration and Policy Function approach

Chapter 7

Solving the Stochastic Growth Model by Collocation

We'll now work on solving the stochastic growth model by the collocation method. This is the same model that we solved earlier by log-linearization. Replicating the essential elements of the model here for convenience, the problem is

$$\max \mathrm{E}_0 \sum_{t=0}^{\infty} \beta^t U(c_t)$$

subject to

$$\begin{aligned} k_{t+1} &= A_t k_t^{\alpha} - c_t + (1-\delta)k_t \\ A_{t+1} &= a(A_t, \epsilon_{t+1}) \end{aligned}$$

A_t is a stationary Markov technology shock sequence and ϵ_t is iid.

The problem rewrites in recursive form by noting that the current state is (k, A). Given the current state, we then know y and need to choose k'. In recursive form, the functional equation problem is,

$$V(k, A) = \max \left\{ U(g(k, A; k')) + \beta \mathrm{E} V(k', A') \right\} \tag{7.1}$$

and the Euler equation associated with optimality is,

$$U_c(k, A; k') = \beta \mathrm{E} U_c(k', A'; k'') \left[\alpha A' k'^{\alpha} + 1 - \delta \right] \tag{7.2}$$

7.0.1 Discretizing the space spanned by the shocks

We need to deal with these random shocks hitting the economy. We proceed in the simplist way and discretize the shocks by assuming that A_t evolves according to a finite element Markov chain. Tauchen and Hussey (1991) show how to discretize continuous time processes, including vector autoregressions.

Markov Chains

Let A_t be a random variable and a_t be a particular realization of A_t. A Markov chain is a stochastic process $\{A_t\}_{t=0}^{\infty}$ with the property that the information in the current realized value of $A_t = a_t$ summarizes the entire past history of the process. That is,

$$\mathrm{P}[A_{t+1} = a_{t+1} | A_t = a_t, A_{t-1} = a_{t-1}, \ldots, A_0 = a_0] = \mathrm{P}[A_{t+1} = a_{t+1} | A_t = a_t]. \quad (7.3)$$

A key Proposition that simplifies probability calculations of Markov chains is,

Proposition 1 *If $\{A_t\}_{t=0}^{\infty}$ is a Markov chain, then*

$$P[A_t = a_t \cap A_{t-1} = a_{t-1} \cap \cdots \cap A_0 = a_0] =$$

$$P[A_t = a_t | A_{t-1} = a_{t-1}] \cdots P[A_1 = a_1 | A_0 = a_0] P[A_0 = a_0]. \quad (7.4)$$

We now state (without proof) some useful facts about Markov chains.

Useful facts about Markov chains with stationary probabilities

1. Let a_j, $j = 1, \ldots, N$ denote the possible states for A_t. A Markov chain has stationary probabilities if the transition probabilities from state a_i to a_j are time-invariant. That is,

   $$\mathrm{P}[A_{t+1} = a_j | A_t = a_i] = p_{ij}$$

   Notice that in Markov chain analysis the first subscript denotes the state that you condition on. For concreteness, consider a Markov chain with two possible states, a_1 and a_2, with transition matrix,

   $$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix},$$

   where the rows of $\mathbf{P}$ sum to 1.

2. The transition matrix over k steps is $\mathbf{P}^k = \underbrace{\mathbf{PP}\cdots\mathbf{P}}_{\text{k}}$.

3. The initial (unconditional) probability vector, π is given by $\pi_j = \sum_{i=1}^{k} \pi_i p_{ij}$. In matrix form, $\underline{\pi} = \mathbf{P}\underline{\pi}$.

Matching the Markov Chain to an AR(1)

Let A_t take on two possible values:

$$A_t = \begin{cases} 1+\mu \\ 1-\mu \end{cases}$$

and let the transition matrix be

$$\mathbf{P} = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix} = \begin{pmatrix} p & 1-p \\ 1-q & q \end{pmatrix}$$

Then the initial probability vector is seen to be $\pi_1 = \pi_2 = 1/2$.

Recall that we used as the law of motion for the productivity shock:

$$A_t = (1-\rho) + \rho A_{t-1} + \epsilon_t \tag{7.5}$$

where $\rho = 0.93$, $\epsilon_t \overset{NID}{\sim} (0, 0.010224^2)$.

We want the Markov chain to have the following properties.

1. $\mathrm{E}(A_t) = 1$
2. $\mathrm{E}(A_t - 1)(A_{t-1} - 1) = \rho$
3. $\mathrm{sd}(A) = \sigma/\sqrt{1-\rho^2}$.

We want to match 3 moments of the process. We have at our disposal 3 parameters, μ, p, q. We've already satisfied the first condition. Since

$$\mathrm{P}\left(A' = a_i \cap A = a_j\right) = p_{ji}\pi_j = p_{ji}\frac{1}{2}$$

Since $\mathrm{Var}(A) = \mathrm{E}(A^2) - 1 = \pi_1 a_1^2 + \pi_2 a_2^2 - 1 = \mu^2$, condition 3 is satisfied if

$$\mathrm{sd}(A) = 0.076 = \mu$$

Condition 2 will be satisfied if

$$\rho = \frac{\mathrm{E}(A'A) - 1}{\mathrm{Var}(A)}$$

Now

$$\begin{aligned} \mathrm{E}(A'A) &= \sum_i \sum_j \mathrm{P}(A' = a_i \cap A = a_j) a_i a_j \\ &= \frac{1}{2}\left\{p(1+\mu)^2 + (2-p-q)(1+\mu)(1-\mu) + q(1-\mu)^2\right\} \end{aligned}$$

We can accomplish this by setting $p = q = 0.965$.

Impulse responses.

1. *Method 1.* $A_0 = 1$. Then we get shocked by size μ which sets $A_1 = 1 + \mu$. $\rho\mu$ persists into the next period so set $A_2 = 1 + \rho\mu$. $\rho^2\mu$ persists into period 3 so set $A_3 = 1 + \rho^2\mu$, etc. That is,

   $$A_t = 1 + \rho^{t-1}\mu$$

   This is our intuition about how an impulse is propagated into the future. The problem for us here is that if the shock is a two-element Markov chain, A_t cannot take on these values. The stochastic growth model won't admit them. Thus, we need to think of something else. (method 2).

2. *Method 2.* Begin with $A_1 = 1, A_2 = 1 + \mu$. Then conditional on $A_2 = 1 + \mu$, simulate the sequence $\{A_t\}_{t=3}^T$ according to the transition probabilities. Repeat 10,000 times and take the mean values across i. The figure shows a comparison of the two methods. (mrkv_1.pgm)

7.1 Value Function Approach

The problem.

$$V(k, A) = \max_{c \in \mathcal{F}} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta \mathrm{E}_A V(k', A') \right\}$$

where

$$k' = Ak^\alpha - c + k(1-\delta)$$

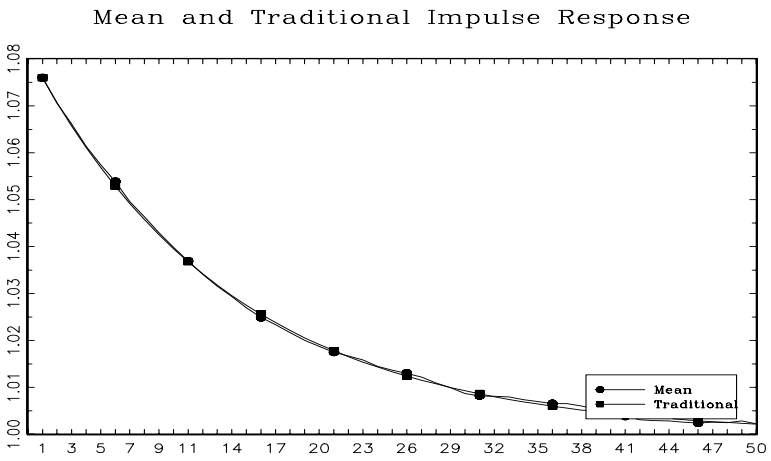

Figure 7.1: Impulse Responses from Markov Chain

Substitute out k'. Rewrite the problem as

$$V(k,A) = \max_{c \in \mathcal{F}} \left\{ \frac{c^{1-\gamma}}{1-\gamma} + \beta \mathrm{E}_A V(Ak^\alpha - c + k(1-\delta), A') \right\}$$

where

$$A = \begin{cases} a_1 = 1+\mu \\ a_2 = 1-\mu \end{cases}$$

We use a_1, a_2 to indicate the shock state in an argument list. In a subscript, the shock state is indexed by $s = 1, 2$ to indicate whether $A = a_1$ or $A = a_2$.

We will use the Chebyshev approximation to the value function, which is a function of two variables (k, A). One option is to use the tensor product basis. A second option is to view A as a function shifter and to allow such shifts to occur by assigning a different set of basis coefficients to each shock state, $s = 1, 2$,

$$\underline{b}_s = \begin{cases} \underline{b}_1 & \text{if} \quad A = a_1 = 1+\mu \\ \underline{b}_2 & \text{if} \quad A = a_2 = 1-\mu \end{cases}$$

The computational problem is to solve for the $2n$ coefficients $\mathbf{B} = (\underline{b}_1, \underline{b}_2)$.

Overview.

1. Form the approximants

$$V(k_i, A) = \begin{cases} V(k_i, a_1) \simeq \hat{V}(k_i, \underline{b}_1) = \sum_{j=1}^{n} b_{j1}\phi_j(k_i) \\ V(k_i, a_2) \simeq \hat{V}(k_i, \underline{b}_2) = \sum_{j=1}^{n} b_{j2}\phi_j(k_i) \end{cases}$$

2. Form the approximant to $V(k', A')$. To do this, we need to compute k', as a function of (k, A). To do this, for each k_i and a_s form a choice set of consumption

$$\{c_{i,s,m}\}_{m=1}^{n} = \{c_{i,s,1}, \ldots, c_{i,s,n}\}$$

For each one of these consumption values there is an implied next period capital stock

$$\{k'_{i,s,m}\}_{m=1}^{n} = \{a_s k_i^{\alpha} - c_{i,s,m} + k_i(1-\delta)\}_{m=1}^{n}$$

3. Form the approximant to $V(k', A')$.

$$V(k'_{i,s,m}, A') = \hat{V}(k'_{i,s,m}, \underline{b}_{s'}) = \sum_{j=1}^{n} b_{j,s'}\phi_j\left(k'_{i,s,m}\right) = \sum_{j=1}^{n} b_{j,s'} T_{j-1}\left(2\left(\frac{k'_{i,s,m} - k_\ell}{k_u - k_\ell}\right) - 1\right)$$

4. We now have the following $2n$ equations in $2n$ coefficients

$$\begin{aligned} (A = a_1) : \sum_{j=1}^{n} b_{j,1}\phi_j(k_i) &= \max\left\{\frac{c_{i,1,m}^{1-\gamma}}{1-\gamma} + \beta \sum_{s=1}^{2} p_{1,s} \sum_{j=1}^{n} b_{j,s}\phi_j\left(a_1 k_i^{\alpha} - c_{i,1,m} + k_i(1-\delta)\right)\right\}_{m=1}^{n} \\ (A = a_2) : \sum_{j=1}^{n} b_{j,2}\phi_j(k_i) &= \max\left\{\frac{c_{i,2,m}^{1-\gamma}}{1-\gamma} + \beta \sum_{s=1}^{2} p_{2,s} \sum_{j=1}^{n} b_{j,s}\phi_j\left(a_2 k_i^{\alpha} - c_{i,2,m} + k_i(1-\delta)\right)\right\}_{m=1}^{n} \end{aligned}$$

Procedure

1. $\underline{b}_1^0, \underline{b}_2^0$ given (guess values). Construct a grid of Chebyshev nodes and map into capital stocks

$$\{u_i\}_{i=1}^{n} \to \{k_i\}_{i=1}^{n} \in [k_\ell, k_u], \qquad k_i = \frac{k_u + k_\ell}{2} + \frac{k_u - k_\ell}{2} u_i$$

2. For $s = 1, 2$ and $k_i, i = 1, \ldots, n$, compute

   - A feasible grid of consumption values built off of the Chebyshev nodes

$\{u_m\}_{m=1}^n$.

$$\{c_{i,s,m}\}_{m=1}^n = \left\{ \frac{a_s k_i^\alpha}{2}(1+u_m) \right\}_{m=1}^n \in [0, a_s k_i^\alpha].$$

- Associate each $c_{i,s,m}$ with

$$k'_{i,s,m} = a_s k_i^\alpha - c_{i,s,m} + k_i(1-\delta)$$

- Construct the polynomial basis associated with k',

$$\phi_j(k'_{i,s,m}) = T_{j-1}\left(2\left(\frac{k'_{i,s,m} - k_\ell}{k_u - k_\ell}\right) - 1\right)$$

- Construct the sequence (vector)

$$\left\{\tilde{V}_m\right\}_{m=1}^n = \left\{ \frac{c_{i,s,m}^{1-\gamma}}{1-\gamma} + \beta \sum_{s'=1}^{2} p_{s,s'} \sum_{j=1}^{n} b_{j,s'} T_{j-1}\left(2\left(\frac{k'_{i,s,m} - k_\ell}{k_u - k_\ell}\right) - 1\right) \right\}_{m=1}^n$$

Let entry m^* be the largest value in this sequence. Call it $\tilde{V}_{m^*}$ and assign it to $V_{i,s}$,

$$V_{i,s}(\underline{b}_1^0, \underline{b}_2^0) = \tilde{V}_{m^*}$$

3. The system of $2n$ equations are for $s = 1, 2$, and $i = 1, \ldots, n$:

$$\sum_{j=1}^{n} b_{j,s}\phi_j(k_i) = V_{i,s}(\underline{b}_1^0, \underline{b}_2^0)$$

In vector notation

1. Define

$$\mathbf{\Phi} = \{\phi_{ij}\} = \{\phi_j(k_i)\} \qquad \mathbf{B} = (\underline{b}_1, \underline{b}_2)$$

2. Chebyshev nodes to capital grid:

$$\underline{u} \to \underline{k}$$

3. Feasible consumption grid and next period capital stock for given (k_i, a_s).

$$\underline{c}_{i,s} = \frac{a_s k_i^\alpha}{2}(\underline{\iota} + \underline{u})$$

$$\underline{k}'_{i,s} = \left(a_s k_i^{\alpha} + k_i(1-\delta)\right) \underline{\iota} - \underline{c}_{i,s}$$

4. Vector of candidate value function evaluations

$$\mathbf{\Phi}_{ps} = \{\phi_j(k'_{i,s,m})\} = \left\{T_{j-1}\left(2\left(\frac{k'_{i,s,m} - k_\ell}{k_u - k_\ell}\right) - 1\right)\right\}$$

$$\underline{\tilde{V}}_{i,s} = \frac{\underline{c}_{i,s}^{1-\gamma}}{1-\gamma} + \beta\left[p_{s,1}\mathbf{\Phi}_{ps}\underline{b}_1^0 + p_{s,2}\mathbf{\Phi}_{ps}\underline{b}_2^0\right]$$

5. Set $V_{i,s}(\underline{b}_1^0, \underline{b}_2^0)$ equal to the maximal entry of $\underline{\tilde{V}}_{i,s}$. For each state $s = 1, 2$, stack into an $n \times 1$ vector $\underline{V}_s(\mathbf{B})$

6. Update coefficients according to

$$\mathbf{B}^{\ell+1} = \mathbf{\Phi}^{-1}\left[\underline{V}_1(\mathbf{B}^\ell), \underline{V}_2(\mathbf{B}^\ell)\right]$$

Simulating the Model (brute force)

1. k_1 given. Generate a sequence

$$\{A_t\}_{t=1}^T$$

according to the Markov chain transition probabilities.

2. For $t = 1$, construct an $n \times 1$ grid of feasible consumption values

$$\{\tilde{c}_{t,m}\}_{m=1}^n \in [0, a_1 k_t^{\alpha}] \quad \text{if} \quad A_t = a_1 = 1 + \mu$$

$$\{\tilde{c}_{t,m}\}_{m=1}^n \in [0, a_2 k_t^{\alpha}] \quad \text{if} \quad A_t = a_2 = 1 - \mu$$

Let $\underline{\iota}$ be a vector of 1s. $\underline{u}$ can be a vector of Chebyshev nodes. Stack in vector

$$\underline{\tilde{c}}_t = \frac{A_t k_t^{\alpha}}{2}\left(\underline{\iota} + \underline{u}\right)$$

3. Use to construct grid of feasible next period capital stocks,

$$\underline{\tilde{k}}_{t+1} = \left[A_t k_t^{\alpha} + k_t(1-\delta)\right] \underline{\iota} - \underline{\tilde{c}}_t$$

4. Find the largest element of the sequence

$$\left\{\frac{\tilde{c}_{t,m}^{1-\gamma}}{1-\gamma} + \beta\left(p_{s,1}\sum_{j=1}^n b_{j,1}\phi_j(\tilde{k}_{t+1,m}) + p_{s,2}\sum_{j=1}^n b_{j,2}\phi_j(\tilde{k}_{t+1,m})\right)\right\}_{m=1}^n$$

Let it be element m^*.

5. Set

$$c_t = \tilde{c}_{t,m^*}$$
$$k_{t+1} = A_t k_t^{\alpha} + k_t(1-\delta) - c_t$$

Repeat for $t = t+1$.

Impulse Response

Follow the procedure outlined above except that the sequence $\{A_t\}_{t=1}^T$ is generated by *method 2*. Repeat N times, and report for each t, the mean values across the simulations.

Simulating the model (Plan B)

We've already solved the model. $\mathbf{B} = [\underline{b}_1, \underline{b}_2]$ are the collocation coefficients. Using the orignial grid of capital stocks,

$$\{k_1, \ldots, k_n\} \in [k_\ell, k_u], \quad u_i = 2\left(\frac{k_i - k_\ell}{k_u - k_\ell}\right) - 1$$

the solution of the model gives an (implicit) decision (transition) rule for $i = 1, \ldots, n, s = 1, 2$,

$$k'_{i,s} = g(k_i, A_s)$$

That is, in the process of solving the model, we compute and obtain $k'_{i,s}, i = 1, \ldots, n, s = 1, 2$.

So let's use these values to estimate (approximate) the transition rule,

$$k'_{i,s} = \sum_{j=1}^{n} \theta_{j,s}\phi_j(k_i) = \sum_{j=1}^{n} \theta_{j,s} T_{j-1}\left(2\left(\frac{k_i - k_\ell}{k_u - k_\ell}\right) - 1\right)$$

Obtain the θ coefficients by

$$\underline{\theta}_s = \left(\mathbf{\Phi}'\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}'\underline{k}'_s$$

Now for $t = 1, \ldots, T$, generate the seqnence $\{A_t\}$ and

$$k_{t+1} \;=\; \sum_{j=1}^{n} \theta_{j,s} T_{j-1}\left(2\left(\frac{k_t - k_\ell}{k_u - k_\ell}\right) - 1\right)$$

$$c_t \;=\; A_t k_t^{\alpha} + (1-\delta)k_t - k_{t+1}$$

```
new;
@-----------------------------------------------------------------------@
@ SGM_VAL.PGM @
@ Collocation method solution of stochastic optimal growth model @
@ by value function iteration @
@ Function approximation by Chebyshev polynomials @
@-----------------------------------------------------------------------@
output file=sgm_val.out reset; output on;
format /rd 8,3;
@=======================================================================@
#include nodeeq.set; @ Equally spaced nodes @
#include nodecheb.set; @ Chebyshev nodes @
#include chebbas.set; @ Chebyshev polynomial interpolants @
#include chebeval.set; @ Evaluation of Cheb.  approximated function @
#include nlsys.set; @ Systems of Nonlinear equations solver @
@=======================================================================@
@:::::::::::::  PARAMETER VALUES ::::::::::::::::::::::::::::::::::::::@
@------------- CRRA Utility ------------------------------------------@
alpha=0.4; @ CAPITAL'S SHARE @
delta=0.012; @ QUARTERLY DEPRECIATION RATE @
beta=0.978; @ QUARTERLY DISCOUNT FACTOR-NORMALIZED BY EFFICIENCY UNITS @
gam=2.0; @ COEFFICIENT OF RELATIVE RISK AVERSION @
rho=0.95; @ AR(1) COEFFICIENT FOR TECHNOLOGY SHOCKS. @
sigma=0.007; @ Innovation Standard Deviation @

p=0.965 0.035,0.035 0.965; @ Transition Matrix @
mu=0.076;
a=zeros(2,1); a[1]=1+mu; a[2]=1-mu; @ Productivity shock @
seed=52093;

sto1=1-beta*(1-delta); @--------------@
sto2=beta*alpha; @ Steady State @
sto3=1/(alpha-1); @--------------@
Kss=(sto1/sto2)^sto3; @ Steady state capital @
Yss=Kss^alpha; @ Steady state output @
Iss=delta*Kss; @ Steady state investment @
Css=Yss-Iss; @ Steady state consumption @
@-----------------------------------------------------------------------@
simtype=1; @ 1:  Single simulation.  2:  Impulse response @
nsim=1000;
crit=0.001; @ convergence criteria @
n0=20; @ Degree of interpolation @
kl=Kss*0.1; ku=Kss*1.4;
b0=zeros(n0,2);
```

```
/*
b0[.,1]=exp(-seqa(1,1,n0)); @ Guess (Initial) collocation coefficients @
b0[.,2]=exp(-seqa(1,1,n0));
/
let sto1= -11.947 3.681 -1.129 0.436 -0.189 0.092
-0.045 0.018 -0.009 0.005 -0.003 0.001 -0.001
0.001 -0.000 ;

let sto2= -12.603 4.085 -1.287 0.502 -0.220 0.102
-0.044 0.020 -0.012 0.007 -0.004 0.002 -0.002
0.001 -0.000 ;
b0[1:rows(sto1),.]=sto1 sto2;
output off;

iter=0; @----------------------Begin updating iterations ------@
retry:
iter=iter+1;
u=nodecheb(n0); @ Chebyshev nodes on [-1,1]@
Phi=chebbas(u); @ Chebyshev polynomial interpolants @
k=(kl+ku)*0.5+0.5*(ku-kl)*u; @ Adjust to [kl,ku] @
kp=zeros(n0,2);
Vb0=zeros(n0,2); @ Store Maximized rhs tildeV_{i,s}@

s=1; do while s le 2;
i=1; do while i le n0;
kpu=a[s]*k[i]^(alpha)+(1-delta)*k[i];
kpl=(1-delta)*k[i];
kpgrid=(kpu+kpl)*0.5+(kpu-kpl)*0.5*u; @ grid k' m=1,...n @
cgrid=a[s]*k[i]^(alpha)+(1-delta)*k[i]-kpgrid; @ grid of consumption @
up=2*(kpgrid-kl)/(ku-kl)-1;
Phip=chebbas(up);
Vhat=((1-gam)^(-1))*cgrid^(1-gam)
+beta*(p[s,1]*Phip*b0[.,1]+p[s,2]*Phip*b0[.,2]);
Vb0[i,s]=maxc(Vhat);
sxx=maxindc(Vhat);
kp[i,s]=kpgrid[sxx]; @ Storing next period capital @
i=i+1; endo;
s=s+1; endo;
b1=inv(Phi)*Vb0;
d=abs(b1-b0);
"iter";;iter;
if sqrt(d'd) gt crit; b0=b1; goto retry; endif;
b=b1; @ FINAL VALUES @
outwidth 80; output on; "Collocation coefficients"; b'; output off;
@---------------- Study the results -----------------@
if simtype eq 1;
theta=inv(phi'phi)*phi'kp;
t1=100; A=zeros(t1,1); e=rndus(t1,1,seed);
```

```
c=zeros(t1,1); k=zeros(t1,1); k[1]=kss;
if e[1] gt 0.5; A[1]=1+mu; else; A[1]=1-mu; endif;
t=2; do while t le t1; @ SEQUENCE OF TECHNOLOGY SHOCKS @
if A[t-1] eq 1+mu;
if e[t] lt p[1,1]; A[t]=1+mu; else; A[t]=1-mu; endif;
endif;
if A[t-1] eq 1-mu;
if e[t] lt p[2,2]; A[t]=1-mu; else; A[t]=1+mu; endif;
endif;
t=t+1; endo;

t=1; do while t le t1-1;
uu=2*(k[t]-kl)/(ku-kl)-1;
if a[t] == 1+mu; k[t+1]=chebeval(theta[.,1],n0,uu); endif;
if a[t] == 1-mu; k[t+1]=chebeval(theta[.,2],n0,uu); endif;
c[t]=a[t]*k[t]^alpha+k[t]*(1-delta)-k[t+1];
t=t+1; endo;
c=trimr(c,0,1); k=trimr(k,0,1);

if simtype eq 2;
theta=inv(phi'phi)*phi'kp;
t1=100; c=zeros(t1,nsim); k=zeros(t1,nsim);
sim=1; do while sim le nsim;
A=zeros(t1,1); A[1:4]=(1+mu)*ones(4,1); e=rndus(t1,1,seed);
t=5; do while t le t1; @ SEQUENCE OF TECHNOLOGY SHOCKS @
if A[t-1] eq 1+mu;
if e[t] lt p[1,1]; A[t]=1+mu; else; A[t]=1-mu; endif;
endif;
if A[t-1] eq 1-mu;
if e[t] lt p[2,2]; A[t]=1-mu; else; A[t]=1+mu; endif;
endif;
t=t+1; endo;

k[1:4,sim]=kss*ones(4,1); c[1:4,sim]=css*ones(4,1);
t=1; do while t le t1-1;
uu=2*(k[t,sim]-kl)/(ku-kl)-1;
if a[t] == 1+mu; k[t+1,sim]=chebeval(theta[.,1],n0,uu); endif;
if a[t] == 1-mu; k[t+1,sim]=chebeval(theta[.,2],n0,uu); endif;
c[t,sim]=a[t]*k[t,sim]^alpha+k[t,sim]*(1-delta)-k[t+1,sim];
t=t+1; endo;
"sim";;sim;
sim=sim+1; endo;
csto=trimr(c,0,1); ksto=trimr(k,0,1);
c=meanc(csto'); k=meanc(ksto');
```

7.2 Policy Function Approach

Here, we want to solve the Euler equation

$$c^{-\gamma} = \beta \mathrm{E}_A \left(c^{'-\gamma} \left[\alpha A' k^{'\alpha-1} + 1 - \delta \right] \right)$$

by finding optimal next period capital stock,

$$k' = g(k, A).$$

Once this is determined, we get consumption,

$$c = Ak^{\alpha} + (1-\delta)k - k'.$$

Procedure in vector notation

1. Create the collocation nodes and Chebysyev polynomial basis

$$\underline{u} = \{u_1, \ldots, u_n\} \rightarrow \underline{k} = \{k_1, \ldots, k_n\}$$

$$\mathbf{\Phi} = \{\phi_j(k_i)\} = \{T_{j-1}(u_i)\}$$

2. Approximant to optimal policy function.

$$\hat{g}(k_i, \underline{b}_s) = \sum_{j=1}^{n} b_{j,s} \phi_j(k_i) = \sum_{j=1}^{n} b_{j,s} T_{j-1}(u_i) \rightarrow \underline{\hat{g}}(\underline{k}, \underline{b}_s) = \mathbf{\Phi} \underline{b}_s$$

$$\underline{k}'_s = \begin{cases} \underline{k}'_1 = \exp\left(\underline{\hat{g}}(\underline{k}, \underline{b}_1)\right) = \exp\left(\mathbf{\Phi}\underline{b}_1\right) & \text{if} \quad A = a_1 (s=1) \\ \underline{k}'_2 = \exp\left(\underline{\hat{g}}(\underline{k}, \underline{b}_2)\right) = \exp\left(\mathbf{\Phi}\underline{b}_2\right) & \text{if} \quad A = a_2 (s=2) \end{cases}$$

   This gives

$$\underline{c}_s = \begin{cases} \underline{c}_1 = a_1 \underline{k}^{\alpha} + (1-\delta)\underline{k} - \underline{k}'_1 & \text{if} \quad A = a_1 (s=1) \\ \underline{c}_2 = a_2 \underline{k}^{\alpha} + (1-\delta)\underline{k} - \underline{k}'_2 & \text{if} \quad A = a_2 (s=2) \end{cases}$$

3. We need c'. In order to get c' we need k''. Given (k_i, a_s) we have $k'_{i,s} = \exp(\hat{g}(k_i, \underline{b}_s)$. Thus, $k''_{i,s,s'} = \exp(\hat{g}(k'_{i,s}, \underline{b}_{s'}))$. So for each k_i, there are 4 possible values of k''. They need to be evaluated.

Construct the the Chebyshev polynomial basis associated with k'.

$$\mathbf{\Phi}_{p,s} = \begin{cases} \mathbf{\Phi}_{p,1} = \left\{\phi_j(k'_{i,1})\right\} = \left\{T_{j-1}\left(2\left(\frac{k'_{i,1}-k'_{\ell,1}}{k'_{u,1}-k'_{\ell,1}}\right)-1\right)\right\} & (A=a_1, s=1) \\ \mathbf{\Phi}_{p,2} = \left\{\phi_j(k'_{i,2})\right\} = \left\{T_{j-1}\left(2\left(\frac{k'_{i,2}-k'_{\ell,1}}{k'_{u,2}-k'_{\ell,2}}\right)-1\right)\right\} & (A=a_2, s=2) \end{cases}$$

$$\underline{k}''_{s,s'} = \begin{cases} \underline{k}''_{1,1} = \exp\left(\mathbf{\Phi}_{p,1}\underline{b}_1\right) & (A=a_1, A'=a_1) \\ \underline{k}''_{1,2} = \exp\left(\mathbf{\Phi}_{p,1}\underline{b}_2\right) & (A=a_1, A'=a_2) \\ \underline{k}''_{2,1} = \exp\left(\mathbf{\Phi}_{p,2}\underline{b}_1\right) & (A=a_2, A'=a_1) \\ \underline{k}''_{2,2} = \exp\left(\mathbf{\Phi}_{p,2}\underline{b}_2\right) & (A=a_2, A'=a_2) \end{cases}$$

Now we can get next period consumption,

$$\underline{c}'_{s,s'} = a_{s'}\underline{k}_s^{'\alpha-1} + (1-\delta)\underline{k}'_s - k''_{s,s'}$$

4. The $2n$ equations to solve are

$$\begin{aligned} (A=a_1): \quad \underline{c}_1^{-\gamma} &= \beta\left\{p_{11}(\underline{c}'_{1,1})^{-\gamma}\left[\alpha a_1\underline{k}_1^{'\alpha-1}+1-\delta\right] + p_{12}(\underline{c}'_{2,1})^{-\gamma}\left[\alpha a_2\underline{k}_2^{'\alpha-1}+1-\delta\right]\right\} \\ (A=a_2): \quad \underline{c}_2^{-\gamma} &= \beta\left\{p_{21}(\underline{c}'_{2,1})^{-\gamma}\left[\alpha a_1\underline{k}_2^{'\alpha-1}+1-\delta\right] + p_{22}(\underline{c}'_{2,2})^{-\gamma}\left[\alpha a_2\underline{k}_2^{'\alpha-1}+1-\delta\right]\right\} \end{aligned}$$

7.3 Parameterized Expectations Algorithm (PEA)

The *parameterized expectations algorithm* was introduced by Marcet (1988). The method can be thought of as a generalized method of undetermined coefficients and has an interpretation in terms of *learning behavior*.

The underlying strategy is to obtain an approximant to the expectation function instead of the policy or value function. We want to solve

$$c_t^{-\gamma} = \beta \mathrm{E}_t\left(c_{t+1}^{-\gamma}\left[\alpha A_{t+1}k_{t+1}^{\alpha-1}+(1-\delta)\right]\right),$$

$$k_{t+1} - A_t k_t^{\alpha} + c_t - (1-\delta)k_t = 0$$

$$\ln(A_{t+1}) - \rho\ln(A_t) - \epsilon_{t+1} = 0$$

Previously, in the policy function approach, we parameterized consumption, which is essentially parameterizing the *lhs* of the Euler equation. Since the *lhs* must equal the *rhs*, we can also think about parameterizing the *rhs*. That is what the PEA algorithm is. We parameterize the expectation function

$$\beta \mathrm{E}_t\left[c_{t+1}^{-\gamma}\left(\alpha A_{t+1}k_{t+1}^{\alpha-1}+1-\delta\right)\right]$$

There are 2 state variables, (k, A). The expectation function is state dependent and must be a function of k_t and A_t.

1. Define
$$x_t \equiv c_{t+1}^{-\gamma}\left(\alpha A_{t+1}k_{t+1}^{\alpha-1} + 1 - \delta\right)$$

2. (Purely for illustrative purposes) Form the approximant using the *complete polynomial basis*,
$$\begin{aligned}\hat{g}(k_t, A_t; \underline{b}) &= \hat{\mathrm{E}}_t(x_{t+1}) \\ &= \hat{\mathrm{E}}_t\left(c_{t+1}^{-\gamma}\left[\alpha A_{t+1}k_{t+1}^{\alpha-1} + 1 - \delta\right]\right) \\ &= \exp\left(b_0 + b_1 \ln k_t + b_2 \ln A_t + b_3 (\ln k_t)^2 + b_4 (\ln A_t)^2 + b_5 (\ln k_t)(\ln A_t)\right)\end{aligned}$$

3. Now from the Euler equation we have
$$c_t(\underline{b}) = (\beta \hat{g}(k_t, A_t; \underline{b}))^{-1/\gamma}$$
and by the transition equation,
$$k_{t+1}(\underline{b}) = A_t k_t(\underline{b})^\alpha - c_t(\underline{b}) + k_t(\underline{b})(1-\delta)$$

4. Generate, according to the model, the sequences
$$\{k_t(\underline{b})\}_{t=1}^T, \quad \{A_t\}_{t=1}^T, \quad \{c_t(\underline{b})\}_{t=1}^T$$
$$\{x_{t+1}(\underline{b})\}_{t=1}^T = \left\{c_{t+1}(\underline{b})^{-\gamma}\left[\alpha A_{t+1}k_{t+1}(\underline{b})^{\alpha-1} + 1 - \delta\right]\right\}_{t=1}^T$$
That is, for $t = 1, \ldots, T$, we have A_t, k_t given. Form $g_t = \exp(z_t'\underline{b}^0)$, where $z_t = (1, \ln k_t, \ln A_t, (\ln k_t)^2, (\ln A_t)^2, (\ln k_t)(\ln A_t))$. Form $c_t(\underline{b}^0)$. This gives k_{t+1}. So we can generate the entire sequence of observations of capital and consumption in the model recursively. Form the sequence $\{x_t(\underline{b}^0)\}$.

5. Regress $x_{t+1}(\underline{b})$ on z_t. Use the estimated value of $\underline{b}$ to update the coefficient vector.

We could be at this for a long time. No matter how large you make T, there will be some sampling error in the OLS estimates. Prior to convergence, the regressors, which depend on $\underline{b}$, will be the 'wrong' values. It is understandable that convergence may be slow.

7.4 Christiano and Fisher's Modified PEA

Instead of generating pseudo time series and running regressions to estimate the basis coefficients, Christiano and Fisher (1999) show how to combine the minimum weighted residual method with Marcet's PEA and to parameterize the expectation function using Chebyshev polynomial approximations. Their modification is useful because it allows us to impose occasionally binding constraints.

Recall thatin the policy function approach, it was difficult to ensure that consumption (or next period capital) would remain positive for all guess values of the coefficients? The modified PEA approach takes care of this problem. We'll proceed in steps first by illustrating the modified PEA approach without imposing the nonnegativity constraint.

7.4.1 Modified PEA–without additional constraints

The model.

$$c^{-\gamma} - \beta \mathrm{E}_A \left((c')^{-\gamma} \left[\alpha A' k'^{\alpha-1} + 1 - \delta \right] \right) = 0 \tag{7.6}$$

$$k' - Ak^{\alpha} + c - (1-\delta)k = 0 \tag{7.7}$$

Parameterize the expectation function by $g(k, A) \simeq \hat{g}(k, \underline{b}_s)$.

$$\mathrm{E} \left\{ \left((c')^{-\gamma} \left[\alpha A' k'^{\alpha-1} + 1 - \delta \right] \right) | A = a_s \right\} = \exp\left(\hat{g}(k, \underline{b}_s) \right) \tag{7.8}$$

The PEA admits a *recursive representation.*

$$c^{-\gamma} = \beta \mathrm{E}_A \left[c'^{-\gamma} \left(\alpha A' k'^{\alpha-1} + 1 - \delta \right) \right] \qquad \text{Euler equation}$$

$$e^{g(k,A)} = \mathrm{E}_A \left(c'^{-\gamma} \left(\alpha A' k'^{\alpha-1} + 1 - \delta \right) \right) \qquad \text{Definition of PEA}$$

$$c^{-\gamma} = \beta e^{g(k,A)} \qquad \text{Sub PEA into Euler}$$

$$k' = Ak^{\alpha} + (1-\delta)k - \left[\beta e^{g(k,A)} \right]^{-1/\gamma} \qquad \text{Transition}$$

$$c'^{-\gamma} = \beta e^{g(k',A')} \qquad \text{Update Euler}$$

Substitute the last three of these equations into the definition of $g(k, A)$ to get the *recursive formulation*,

$$e^{g(k,A)} = \mathrm{E}_A \left\{ \beta e^{g(k',A')} \left[\alpha A' \left(Ak^{\alpha} + (1-\delta)k - \left(\beta e^{g(k,A)} \right)^{-1/\gamma} \right) \right] + 1 - \delta \right\}$$

Procedure (vectorized)

1. For $s = 1, 2$, compute

$$\begin{aligned}
\underline{g}_s &= \mathbf{\Phi}\underline{b}_s \\
\underline{c}_s &= \beta \exp(\underline{g}_s) \\
\underline{k}'_s &= a_s \underline{k}^{\alpha} + (1-\delta)\underline{k} - \underline{c}_s \\
\mathbf{\Phi}_{p,s} &= \left\{\phi_j(k'_{i,s})\right\} = \left\{T_{j-1}\left(2\left(\frac{k'_{i,s} - k_\ell}{k_u - k_\ell}\right) - 1\right)\right\} \\
\underline{g}_{s,s'} &= \mathbf{\Phi}_{p,s}\underline{b}_{s'} \\
\underline{c}'_{s,s'} &= \beta \exp\left(\underline{g}_{s,s'}\right)
\end{aligned}$$

2. Evaluation of *rhs*, for $s = 1, 2$,

$$\underline{R}_s = \sum_{s'=1}^{n} p_{s,s'} \left\{\beta e^{\underline{g}_{s,s'}} \left[\alpha a_{s'} \left(a_s \underline{k}^{\alpha} + (1-\delta)\underline{k} - [\beta e^{\underline{g}_s}]^{-1/\gamma}\right)\right]\right\}$$

3. Update coefficients

$$\mathbf{B}^{\ell+1} = \mathbf{\Phi}^{-1}\left[\ln \underline{R}_1, \underline{R}_2\right]$$

7.4.2 Modified PEA and occasionally binding constraints

The problem.

$$\max \mathrm{E}_0 \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma}$$

subject to

$$\begin{aligned}
c_t + k_{t+1} - (1-\delta)k_t &= A_t k_t^{\alpha} \\
k_{t+1} - (1-\delta)k_t &\geq 0 \quad \text{(Irreversibility)}
\end{aligned}$$

Set up the *Lagrangian.*

$$L = \mathrm{E}_0 \sum_{t=0}^{\infty} \beta^t \left\{\frac{c_t^{1-\gamma}}{1-\gamma} + h_t\left[k_{t+1} - (1-\delta)k_t\right]\right\}$$

$\{h_t\}$ is a sequence of Lagrange multipliers for the irreversibility constraint.

First-order conditions.

$$c^{-\gamma} - h - \beta \mathrm{E}\left\{(c')^{-\gamma}\left[\alpha A' k'^{\alpha-1} + 1 - \delta\right] - h'(1-\delta)\right\} = 0$$

$$
\begin{aligned}
& k' - (1-\delta)k \geq 0 \\
& h \geq 0 \\
& h(k' - (1-\delta)k) = 0
\end{aligned}
$$

The solution is a pair of functions. One for the optimal policy (consumption or next period capital) and the other for the Lagrange multipler.

Parameterize the expectations

$$
e^{g(k,A)} = \mathrm{E}_A \left\{ c'^{-\gamma} \left[\alpha A' k'^{\alpha-1} + 1 - \delta \right] - h'(1-\delta) \right\}
$$

By the Euler equation,

$$
\begin{aligned}
h &= c^{-\gamma} - \beta e^{g(k,A)} \\
k' &= \max \left\{ (1-\delta)k, Ak^{\alpha} + (1-\delta)k - \left(\beta e^{g(k,A)} \right)^{-1/\gamma} \right\} \\
h' &= c'^{-\gamma} - \beta e^{g(k',A')} \\
k'' &= \max \left\{ (1-\delta)k', A'k'^{\alpha} + (1-\delta)k' - \left(\beta e^{g(k',A')} \right)^{-1/\gamma} \right\}
\end{aligned}
$$

Substitute these expressions back into the Euler equation to eliminate the Lagrange multiplers. The result is the system of equations that we want to solve.

$$
e^{g(k,A)} = \mathrm{E} \left\{ c'^{-\gamma} \left(\alpha A' k'^{\alpha-1} \right) + (1-\delta)\beta e^{g(k',A')} \right\} \tag{7.9}
$$

where

$$
\begin{aligned}
c' &= A'k'^{\alpha} + (1-\delta)k' - k'' \\
k' &= \max \left\{ (1-\delta)k, Ak^{\alpha} + (1-\delta)k - \left(\beta e^{g(k,A)} \right)^{-1/\gamma} \right\} \\
k'' &= \max \left\{ (1-\delta)k', A'k'^{\alpha} + (1-\delta)k' - \left(\beta e^{g(k',A')} \right)^{-1/\gamma} \right\}
\end{aligned}
$$

Detailed Procedure

Solve the equation

$$
\exp(g_{i,s}) = \sum_{s'=1}^{2} p_{s,s'} \left(c'^{-\gamma}_{i,s,s'} \alpha A_{s'} k'^{\alpha-1}_{i,s} + (1-\delta)\beta \exp(g'_{i,s,s'}) \right)
$$

where

$$\begin{aligned} k'_{i,s} &= \max\left\{(1-\delta)k_i, A_s k_i^{\alpha} + (1-\delta)k_i - (\beta\exp(g_{i,s}))\right\} \\ k''_{i,s,s'} &= \max\left\{(1-\delta)k'_{i,s}, A_{s'}k'^{\alpha}_{i,s} + (1-\delta)k'_{i,s} - \left(\beta\exp(g'_{i,s,s'})\right)\right\} \\ c'_{i,s,s'} &= A_{s'}k'^{\alpha}_{i,s} + (1-\delta)k'_{i,s} - k''_{i,s,s'} \end{aligned}$$

Compute in this order

1. The collocation nodes: $\{k_1, \ldots, k_n\} \in [k_\ell, k_u]$.
2. $g_{i,s} = g(k_i, A_s) = \sum_{j=1}^{n} b_{j,s}\phi_j(k_i) = \sum_{i=1}^{n} b_{j,s}T_{j-1}\left(2\left(\frac{k_i - k_\ell}{k_u - k_\ell}\right) - 1\right)$.
3. $k_{i,s} = \max\left\{(1-\delta)k_i, A_s k_i^{\alpha} + (1-\delta)k_i - (\beta\exp(g_{i,s}))^{-1/\gamma}\right\}$.
   This produces a grid of $k'_{i,s}$ for $i = 1, \ldots, n, s = 1, 2$.
4. $g'_{i,s,s'} = g(k'_{i,s}, A_{s'}) = \sum_{j=1}^{n} b_{j,s'}\phi_j(k'_{i,s}) = \sum_{j=1}^{n} b_{j,s'}T_{j-1}\left(2\left(\frac{k'_{i,s} - k_\ell}{k_u - k_\ell}\right) - 1\right)$.
5. $k'_{i,s,s'} = \max\left\{(1-\delta)k'_{i,s}, A_{s'}k'^{\alpha}_{i,s} + (1-\delta)k'_{i,s} - \left(\beta\exp(g'_{i,s,s'})\right)^{-1/\gamma}\right\}$
6. $k'_{i,s,s'} = A_{s'}k'^{\alpha}_{i,s} + (1-\delta)k'_{i,s} - k'_{i,s,s'}$
7. $R_{i,s} = \sum_{s'=1}^{2} p_{s,s'}\left(c'^{-\gamma}_{i,s,s'}\alpha A_{s'}k'^{(\alpha-1)}_{i,s} + (1-\delta)\beta\exp(g'_{i,s,s'})\right)$

Update coefficients

$$\mathbf{B}^{\ell+1} = \mathbf{\Phi}^{-1}\left[\ln \underline{R}_1, \ln \underline{R}_2\right]$$

where $\mathbf{B} = (\underline{b}_1, \underline{b}_2)$ and the indices are for states 1 and 2.

```
new;
@-----------------------------------------------------------------------@
@ MPEA.PGM @
@ Collocation method solution of stochastic optimal growth model @
@ Christiano-Fisher modified PEA method @
@ Function approximation by Chebyshev polynomials @
@-----------------------------------------------------------------------@
output file=mpea.out reset; output on;
format /rd 8,3;
@=======================================================================@
#include nodeeq.set; @ Equally spaced nodes @
#include nodecheb.set; @ Chebyshev nodes @
#include chebbas.set; @ Chebyshev polynomial interpolants @
#include chebeval.set; @ Evaluation of Cheb.  approximated function @
#include nlsys.set; @ Systems of Nonlinear equations solver @
@=======================================================================@
@::::::::::::::  PARAMETER VALUES ::::::::::::::::::::::::::::::::::::::@
@------------- CRRA Utility ------------------------------------------@
```

```
alpha=0.4; @ CAPITAL'S SHARE @
delta=0.012; @ QUARTERLY DEPRECIATION RATE @
beta=0.978; @ QUARTERLY DISCOUNT FACTOR-NORMALIZED BY EFFICIENCY UNITS @
gam=2.0; @ COEFFICIENT OF RELATIVE RISK AVERSION @
rho=0.95; @ AR(1) COEFFICIENT FOR TECHNOLOGY SHOCKS. @
sigma=0.007; @ Innovation Standard Deviation @

p=0.965 0.035,0.035 0.965; @ Transition Matrix @
mu=0.076;
a=zeros(2,1); a[1]=1+mu; a[2]=1-mu; @ Productivity shock @
seed=52093;

sto1=1-beta*(1-delta); @--------------@
sto2=beta*alpha; @ Steady State @
sto3=1/(alpha-1); @--------------@
Kss=(sto1/sto2)^sto3; @ Steady state capital @
Yss=Kss^alpha; @ Steady state output @
Iss=delta*Kss; @ Steady state investment @
Css=Yss-Iss; @ Steady state consumption @
@-------------------------------------------------------------@
crit=1e-4; @ convergence criteria @
n0=10; @ Degree of interpolation @
kl=Kss*0.5; ku=Kss*1.5;
let bsto1= -2.975 -0.217 0.010 -0.001 0.000 -0.000 ;
let bsto2= -2.888 -0.223 0.011 -0.001 0.000 -0.000 ;
B0=zeros(n0,2); B0[1:rows(bsto1),1]=bsto1;B0[1:rows(bsto1),2]=bsto2;
iter=0; @----------------------Begin updating iterations ------@
retry:
iter=iter+1;
u=nodecheb(n0); @ Chebyshev nodes on [-1,1]@
Phi=chebbas(u); @ Chebyshev polynomial interpolants @
k=(kl+ku)*0.5+0.5*(ku-kl)*u; @ Adjust to [kl,ku] @
kp=zeros(n0*2,1);
kpp=zeros(n0*4,1); gp=zeros(n0*4,1);
cp=zeros(n0*4,1);
rhs=zeros(n0,2);

i=1; do while i le n0;
g=Phi*B0;
s=1; do while s le 2;
l=s+(i-1)*2;
atmp=(1-delta)*k[i];
btmp=a[s]*k[i]^(alpha)+(1-delta)*k[i]-(beta*exp(g[i,s]))^(-1/gam);
kp[l]=maxc(atmp|btmp);
sp=1; do while sp le 2;
lp=sp+(s-1)*2+(i-1)*4;
gp[lp]=chebeval(b0[.,sp],n0,(2*(kp[l]-kl)/(ku-kl)-1));
atmp=(1-delta)*kp[l];
```

```
btmp=a[sp]*kp[l]^(alpha)+(1-delta)*kp[l]-(beta*exp(gp[lp]))^(-1/gam);
kpp[lp]=maxc(atmp|btmp);
cp[lp]=a[sp]*kp[l]^(alpha)+(1-delta)*kp[l]-kpp[lp];
rhs[i,s]=rhs[i,s]+p[s,sp]*(cp[lp]^(-gam)*alpha*a[sp]*kp[l]^(alpha-1)
+(1-delta)*beta*exp(gp[lp]));
sp=sp+1; endo;
s=s+1; endo;
i=i+1; endo;

B1=inv(Phi)*(ln(rhs));
d=abs(B1-B0);
"iter";;iter;
if sqrt(d'd) gt crit; B0=B1; goto retry; endif;
b=b1; @ FINAL VALUES @

outwidth 80; output on; "Collocation coefficients"; b'; output off;

end;
```