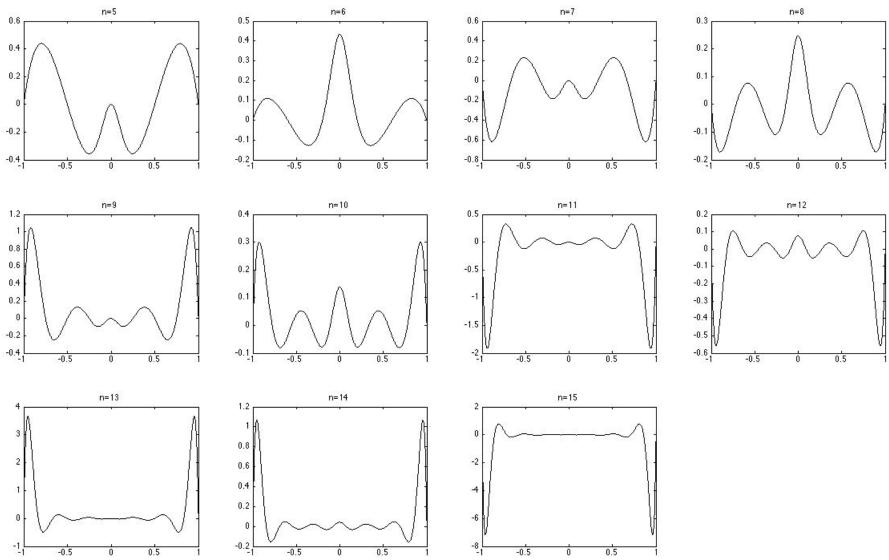
Computational Economics: Problem Set 4

Yangming Bao, ID: 5601239
Cheung Ying Lun, ID: 5441897

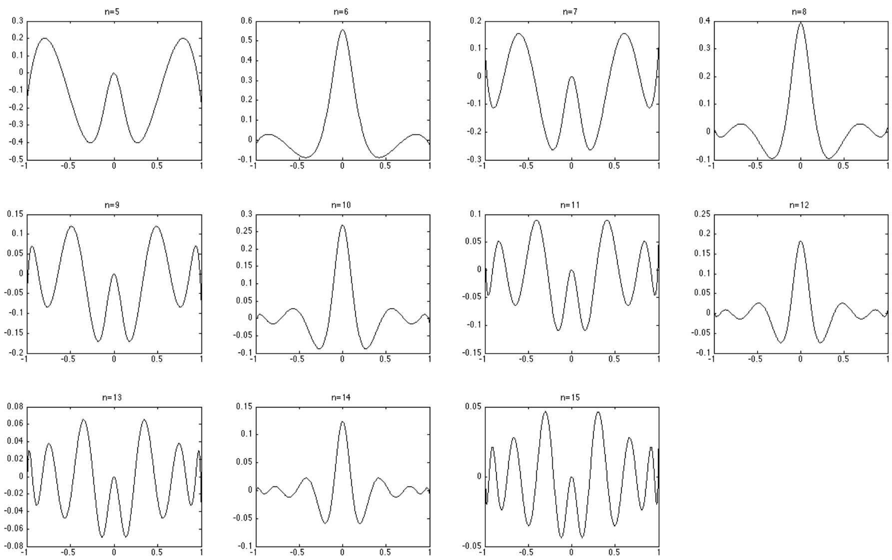
Problem 1: Interpolation of Simple Function

- If using Chebychev polynomials with equidistant nodes, the solution for some large n is out of expectation since the matrix of basis function is close to 0 and it cause the inverse matrix unreasonable large. So if we compare the residual by comparing the solution when n increase from 5 to 15 with the function itself, we can see the pattern as follows. When n increases, the residual converge to zero except the



boundaries. It means the approximate function is more close to the exact function (except the boundaries) when the nodes increase.

- if using Chebychev nodes, the figure looks as follows. When n increase, one significant feature contrast with former one is that the residual at boundaries are small, actually smaller than the points in the middle. Also, the residual are more close to 0 when n increase, which indicates the approximate function is more close to the exact one evenly (including the boundaries).



Problem 2: Simple Optimization Problem

1. The FOC with respect to C_0 reads

$$-(C_0 - \overline{C}) + \mathbb{E}[W_0(1+r) - C_0 - \overline{C}] \overset{!}{=} 0 \tag{1}$$
$$\implies C_0^* = \frac{1}{2}\mathbb{E}[W_0(1+r)] = \frac{W_0}{2}(1+\mathbb{E}[r]). \tag{2}$$

2. The optimal consumption at time 0 only depends on the expected return, irrespective of the variance of r. It does not make economic sense since we expect a risk-averse agent would change one's behavior when risk changes.

3. The FOC with respect to C_0 reads

$$C_0^{-\gamma} - \mathbb{E}[(W_0(1+r) - C_0)^{-\gamma}] \overset{!}{=} 0 \tag{3}$$
$$\implies C_0^* = \left(\mathbb{E}[(W_0(1+r) - C_0)^{-\gamma}]\right)^{-\frac{1}{\gamma}}. \tag{4}$$

4. The maximum percentage deviation is 7.8759e-13%. See Figure 1.

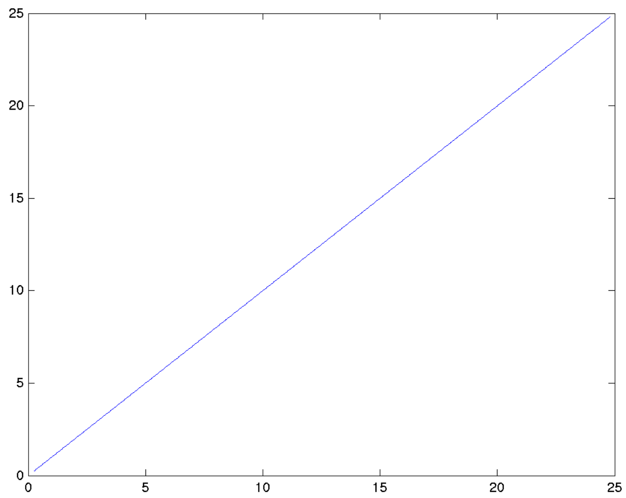


Figure 1: Solution against Chebychev Approximation

5. The maximum percentage deviation increases, but not very much.

Problem 3: Portfolio Choice

1. FOC with respect to α, we can have

$$E\left[\left(1+r^f+\alpha(r-r^f)\right)^{-\gamma_i}(r-r^f)\right]$$

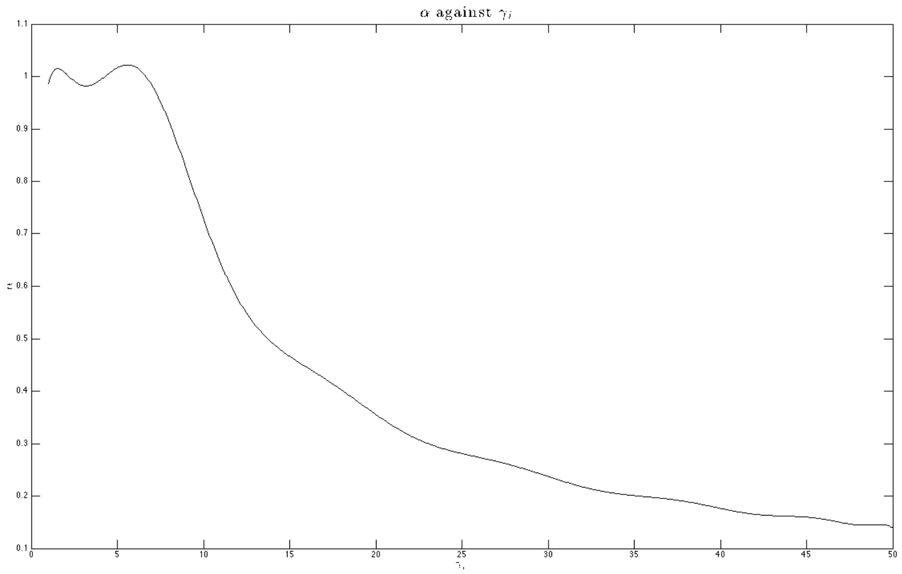
the solution of the equation is the optimal portfolio share for each agent i.

2. If we use the parametrization of problem 2, which is $r^f = 0.02$, $r^{\min} = -0.08$ and $r^{\max} = 0.12$ with the probability of 0.5, respectively, then the problem can be written as

$$\begin{aligned} 0.5 \times \left[(1.02+0.1\alpha)^{-\gamma_i} \times 0.1\right] + 0.5 \times \left[(1.02+(-0.1)\alpha)^{-\gamma_i} \times 0.1\right] = 0 \\ \Leftrightarrow \quad (1.02+0.1\alpha)^{-\gamma_i} = (1.02-0.1\alpha)^{-\gamma_i} \end{aligned}$$

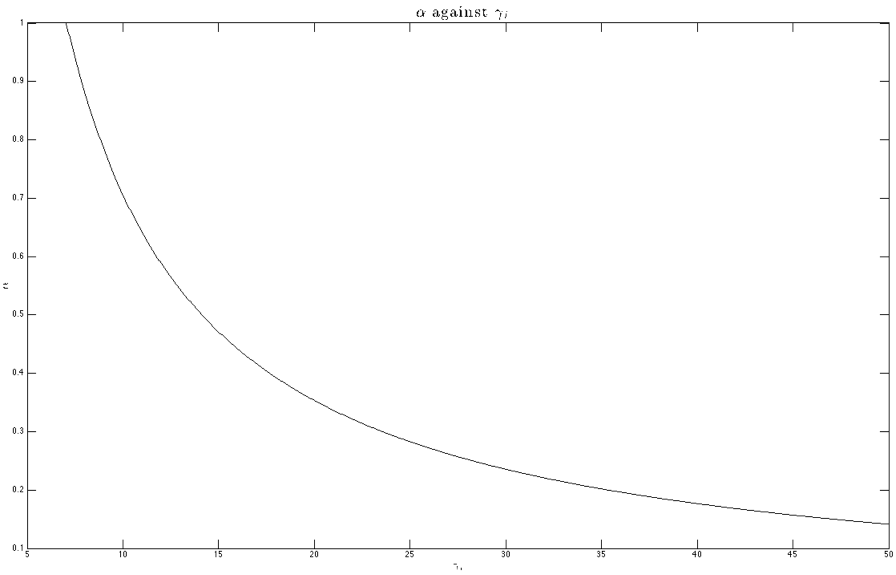
Then it will lead to $\alpha = 0$, regardless of γ_i.

When I set $p = 0.8$, the plot of α against γ_i looks as follows, From the figure, we



can observe that in the neighborhood of the point where α is binding, the points are not exactly smooth at 1 due to the characteristics of the basic function.

3. $\bar{\gamma} = 7.04$ when α is just binding.
The graph now looks smoother than before.



Problem 4: Policy Function Approximation in the Neoclassical Growth Model

1. The Lagrangian for households is

$$\mathcal{L} = \sum_{t=0}^{\infty} \beta^t \ln C_t - \lambda_t [C_t + K_{t+1} - (1 + r_t - \delta) K_t + w_t] \tag{5}$$

The household's FOCs are

$$\beta^t C_t^{-1} - \lambda_t \overset{!}{=} 0 \implies \lambda_t = \beta^2 C_t^{-1} \tag{C_t}$$

$$-\lambda_t + (1 + r_{t+1} - \delta)\lambda_{t+1} \overset{!}{=} 0 \implies \lambda_t = (1 + r_{t+1} - \delta)\lambda_{t+1} \tag{K_{t+1}}$$

Thus we have

$$C_{t+1} = \beta(1 + r_{t+1} - \delta) C_t. \tag{6}$$

For firms, the FOC is

$$\alpha K_t^{\alpha - 1} - r_t \overset{!}{=} 0 \implies r_t = \alpha K_t^{\alpha - 1}. \tag{7}$$

2. At equilibrium, both firm's and household's problems are solved. Moreover, at

steady state $C_t = C_{t+1} = C$ and $K_t = K_{t+1} = K$. Thus, we have

$$1 = \beta(1 + \alpha K^{\alpha-1} - \delta) \implies K^* = \left(\frac{\beta^{-1} + \delta - 1}{\alpha}\right)^{\alpha-1} \approx 7.2112. \tag{8}$$

Due to the market clearing, consumption equals production and

$$\widetilde{C}(K) = K^\alpha. \tag{9}$$

3. See code file `Q4.m`.

4. See code file `Q4.m`.

5. See Figure 2.

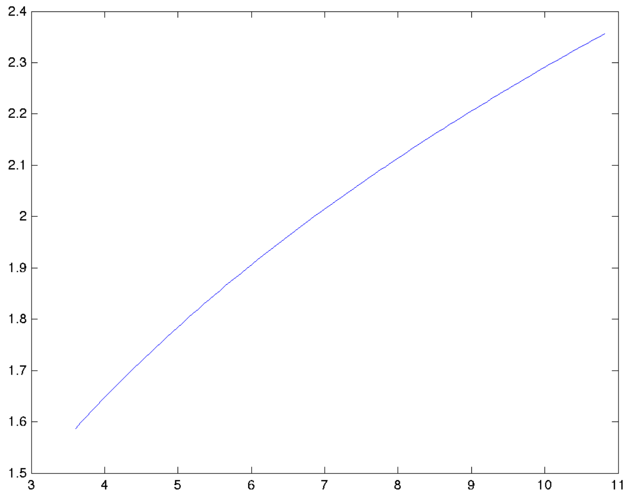


Figure 2: Consumption Policy

6. Capital grows monotonically with decreasing speed towards the steady state. See Figure 3.

7. Maximum absolute error = 3.8648e-07.
   Average absolute error = 7.315e-08.

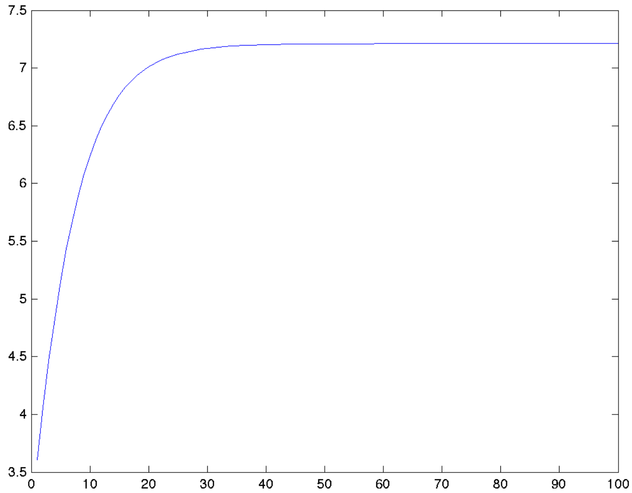


Figure 3: Capital

Question 1

`Q1.m`

```
clc;clear;
% cepath='/Users/baoyangming/Dropbox/gsefm/2015SoSe/Computational Economics/Applied Computational Economics and Finance/compecon/';
% path([cepath 'cetools;' cepath 'cedemos'],path);

% Set domain of interpolation

a =    -1;
b =    1;

%% using Chebychev polynomials for n equidistant nodes.
% for some large n
n_big=1000;

% define a vector of equidistant nodes, x
x_big = nodeunif(n_big,a,b);
% define the function space for Chebychev polynomials and
% calculate the matrix of basis functions, T
fspace_big = fundefn('cheb',n_big,a,b);
```

```
T_big = funbas(fspace_big,x_big);
% calculate the function values at x
y_big = feval(@func,x_big');
% finally get the polynomial coefficients.
c_big=(T_big'*T_big)^(-1)*(T_big'*y_big');
y_bar = T_big*c_big;

figure('Name','Comparison using equidistant nodes');

for n=5:15

    xnode = nodeunif(n,a,b);
    fspace = fundefn('cheb',n,a,b);
    T = funbas(fspace,xnode);
    % calculate the function values at x
    y = feval(@func,xnode);
    c=(T'*T)^(-1)*(T'*y);

    % plot the residual
    T_use=funbas(fspace,x_big);
    y_tilda=T_use*c;
    res = y_big'-y_tilda;

    subplot(3,4,n-5+1);
    plot(x_big,res,'k','LineWidth',1.2);
    title(strcat('n=',num2str(n)));
end



%% Repeat the exercise using Chebychev nodes
clearvars -except a b;
% for some large n
n_big = 1000;
% define the function space for Chebychev polynomials and
% calculate the matrix of basis functions, T
fspace_big = fundefn('cheb',n_big,a,b);
x_big = funnode(fspace_big);
T_big = funbas(fspace_big,x_big);

% calculate the function values at x
```

```
y_big = feval(@func,x_big);

% finally get the polynomial coefficients.
c_big=(T_big'*T_big)^(-1)*T_big'*y_big;
y_bar = T_big*c_big;

figure('Name','Comparison using Chebychev nodes');
for n=5:15
    % calculate the matrix of basis functions, B
    fspace = fundefn('cheb',n,a,b);
    xnode = funnode(fspace);
    T = funbas(fspace,xnode);

    % calculate the function values at x
    y = feval(@func,xnode);

    % finally get the polynomial coefficients.
    c=(T'*T)^(-1)*(T'*y);


    % plot the residual
    T_use=funbas(fspace,x_big);
    y_tilda=T_use*c;
    res = y_bar-y_tilda;

    subplot(3,4,n-5+1);
    plot(x_big,res,'k','LineWidth',1.2);
    title(strcat('n=',num2str(n)));
end
```

func.m

```
function y=func(x)
%function in Q1
y = 1./(1+25*x.^2);
end
```

Question 2

Q2.m

```
clc;clear;
cepath='/Users/YingLun/Documents/Dropbox/Academic/Postgraduate/GSEFM/PhD/Year 2/Summersemester/computational economics/Applied Computational Economics and Finance/compecon/';
path([cepath 'cetools;' cepath 'cedemos'],path);

%% Parameters

r_min    = -0.08;
r_max    = 0.12;
gamma    = 2;
p        = 0.5;

func     = @(W0)Opt_Comp(W0,r_min,r_max,gamma,p);

W_max    = 50;
W_min    = 0.5;

n_node   = 15;

%% Compute the Chebyshev approximation

fspace   = fundefn('cheb',n_node,W_min,W_max);
W_grid   = funnode(fspace);

C0_hat   = func(W_grid);
T_hat    = funbas(fspace,W_grid);
alpha    = (T_hat'*T_hat)\(T_hat'*C0_hat);

W_new    = linspace(0.5,50,1000)';
T_new    = funbas(fspace,W_new);

C0_new   = T_new*alpha;

true_C   = func(W_new);
fig      = plot(true_C,C0_new);
% print('-dpng',fig)

disp(['Max. percentage error = ',num2str(max((true_C-C0_new)./true_C)*100),'%'])
```

`Obj.m`

```
function C0 = Opt_Comp(W0,r_min,r_max,gamma,p)
%This function computes the optimal consumption C0*
%   INPUT:
%        W0: initial wealth
%     r_min: return at bad state
%     r_max: return at good state
%     gamma: risk aversion coefficient
%         p: probability of good state
%
%   OUTPUT:
%        C0: optimal consumption at t=0

%% Initialization
eps         = 1e-5;
del         = 1e-5;
max_it      = 1e6;
ini_Jac     = eye(length(W0));
ini_val     = W0/2;
func        = @(C)((p*(W0*(1+r_max)-C).^(-gamma)+(1-p)*(W0*(1+r_min)-C).^(-gamma)).^(-1/gamma)-C);
stop_crit   = [eps,del,max_it];

%% Compute C0
C0          = Inverse_Broyden_Method(func,ini_Jac,ini_val,stop_crit);

end
```

`NonLCon.m`

```
function roots = Inverse_Broyden_Method(func,ini_Jac,...
                                ini_val,...
                                stop_crit)
%This function perform the Newton's method for root-finding problem.
%         func: a function handle for value of the root-finding problem.
%          Jac: a function handle of the Jacobian function of the problem
%      ini_val: initial value
%    stop_crit: stopping criteria = [eps,del,max_it]

eps = stop_crit(1);
del = stop_crit(2);
max_it = stop_crit(3);
```

```
it = 0;
cont = true;

if length(ini_val)==size(ini_val,2)
    xold = ini_val';
else
    xold = ini_val;
end

B = inv(ini_Jac);
fold = func(xold);

while cont
    it = it+1;

    xnew = xold-B*fold;
    fnew = func(xnew);
    dx = xnew-xold;
    df = fnew-fold;
    B  = B+((dx-B*df)*dx'*B)/(dx'*B*df);
    if (norm(xold-xnew)<=eps*(1+norm(xnew))) || (it==max_it)
        cont = false;
    end
    xold = xnew;
    fold = fnew;
end

if norm(func(xnew))<=del
    disp(['Convergence after ',num2str(it),' iterations.'])
    roots = xnew;
else
    disp('Convergence failed.')
    roots = [];
end

end
```

Question 3

Q3.m

```
clc;clear;


% parameters
p = 0.8; % probability
gamma_min = 1;
gamma_max = 50;
n_node = 15;
fspace  = fundefn('cheb',n_node,gamma_min,gamma_max);
gamma_grid  = funnode(fspace);
T = funbas(fspace,gamma_grid);
y =  func(p,gamma_grid);
c = (T'*T)\(T'*y);

gamma = linspace(1,50,1000);
T_new = funbas(fspace,gamma');
alpha = T_new*c;


% plot alpha against gamma
plot(gamma,alpha,'k','LineWidth',1.2);
ylabel('$\alpha$','Interpreter','latex');
xlabel('$\gamma_i$','Interpreter','latex');
title('$\alpha$ against $\gamma_i$','Interpreter','latex');
```

func.m

```
function alpha = func(p,gamma)
% function for Q3

c = (p/(1-p)).^(-1./gamma);
alpha = max(min(10.2.*(1-c)./(1+c),1),0);

end
```

Question 4

Q4.m

```
clear , clc

%% Parameters

beta      = 0.96;
alpha     = 0.36;
delta     = 0.06;

m         = 9;
N         = 100;

%% Chebyshev approximation

Kstar     = ((1/beta+delta-1)/alpha)^(1/(alpha-1));
fspace    = fundefn('cheb',m,0.5*Kstar,1.5*Kstar);
K_grid    = funnode(fspace);
T_hat     = funbas(fspace,K_grid);

C_hat     = K_grid.^alpha;
theta     = (T_hat'*T_hat)\(T_hat'*C_hat);

K_uni     = nodeunif(N,0.5*Kstar,1.5*Kstar);
T_new     = funbas(fspace,K_uni);
C_new     = T_new*theta;

fig       = figure;
plot(K_uni,C_new);
print(fig,'-dpng','Q4_5')

%% Path of capital
K_t       = zeros(100,1);
K_t(1)    = 0.5*Kstar;
for t=1:99
    K_t(t+1) = K_Policy(K_t(t),alpha,beta,delta);
end
fig       = figure;
plot(1:100,K_t);
```

```
print(fig,'-dpng','Q4_6')

%% Approximation error
% K
K_rand   = 0.5*Kstar+Kstar.*rand(1000,1);
T_rand   = funbas(fspace,K_rand);
C_rand   = T_rand*theta;
% K'
Kprime   = K_Policy(K_rand,alpha,beta,delta);
Tprime   = funbas(fspace,Kprime);
Cprime   = Tprime*theta;
% Error
R        = Cprime-C_rand*beta.*(1+alpha*Kprime.^(alpha-1)-delta);
E        = R./C_rand;

disp(['Max. absolute error = ',num2str(max(abs(E)))])
disp(['Avg. absolute error = ',num2str(mean(abs(E)))])
```

obj_fun.m

```
function Kprime = K_Policy(K,alpha,beta,delta)
%This function computes the capital policy rule.


%% Initialization
eps          = 1e-6;
del          = 1e-4;
max_it       = 1e6;
ini_Jac      = eye(length(K));
ini_val      = K;
func         = @(Kprime)((beta*(1+alpha*Kprime.^(alpha-1)-delta)).^(1/alpha)
    .*K-Kprime);
stop_crit    = [eps,del,max_it];

%% Compute C0
Kprime       = Inverse_Broyden_Method(func,ini_Jac,ini_val,stop_crit);

end
```