Answers to Problem Set 5
Group name: Ferienspass

Sebastian Kühnl: 5642348
Alexander Dück (as: reebyte): 5504077
Patrick Blank (as: paddyblank): 6729110
Christian Wierschem: 6729288

1 Question 1

Main code for Gaussian Quadrature and Monte Carlo integration:

```
clear;
clc;
close all;
%PS5P1
%assume [-1,1]
seed=33;
rng(seed);
%Gaussian Quadrature
%use Gauss-Legendere Quadrature, since the RV might not be normal and no
%discounting
xmin=-1;
xmax=1;
n=[100;1000;10000;50000];
f={@fivefct1;@fivefct2;@fivefct3};
nodes=[2;3;4;5;7];
integral=nan(3,5);
for j=1:3
  for i=1:5
    clear Clear w;
    clear Clear b;
    %get node points and weights for new nodes
    [b,w]=qnwlege(nodes(i),xmin,xmax);
    %evaluate approximated function using chebyshev with chebyshev nodes
    [yap,p,stuff]=cheb(f{j},b,nodes(i),xmin,xmax);
    integral(j,i)=w'*p;                                   %integral value
    clear  Clear p;
  end
end

```

```

%Monte Carlo Quadrature integration
%first of all, draw random numbers on uniform [-1,1]
integralm=nan(3,4);
x=zeros(2,1); %for initial comparison
for i=1:4
  %x points for new n as uniformly distributed on
      [-1,1]
  if n(i)>length(x)
    clear Clear x;
    x=unifrnd(-1,1,n(i),1);
    %alternatively scalable to [a;b] by
    %x=a+rand(n(i),1)*b-a);
  end
  for j=1:3
    y=f{j}(x); %evaluate function at x
    integralm(j,i)=(xmax-xmin)/n(i)*sum(y);  %integral
        values
    clear  Clear y;
  end
end
disp(integral);
disp(integralm);
```

Chebyshev function for polynomial approximation in Gaussian quadrature:

```
function [yequi,ychebsli,ycheblec]=cheb(fct,x,m,xmin,
    xmax)
% In Miranda-Fackler, in fundefn, n is the degree of
    approximation, which
% is the number of nodes (m) -1. However, there is a
    problem with 2 nodes,
% so this is also set to 2 and is kept in mind.
c=max(m-1,2);

%define function space with fundefn
fspace=fundefn('cheb',c,xmin,xmax);
distance=(xmax-xmin)/(m-1);
nodesequi=zeros(m,1);
ynodesequi=zeros(m,1);
nodeschebslides=zeros(m,1);
ynodeschebsli=zeros(m,1);
ynodescheblec=zeros(m,1);
nodescheblecture=zeros(m,1);
%create nodes
%also, calculate function values at x
```

```
for j=1:m
  nodesequi(j)=xmin+(j-1)*distance;             %equidistant nodes
  ynodesequi(j)=fct(nodesequi(j));              %function values
  nodeschebslides(j)=-cos((2*j-1)*pi/(2*m));     %Chebyshev nodes according to slide set 7
  nodescheblecture(j)=-cos((2*j-1)*pi/(m));    %Chebyshev nodes according to lecture notes
  ynodeschebsli(j)=fct(nodeschebslides(j));
  ynodescheblec(j)=fct(nodescheblecture(j));
end

%calculate the matrix of basis functions
Bequi=funbas(fspace,nodesequi);  %equidistant
Bchebsli=funbas(fspace,nodeschebslides);  %Chebyshev
Bcheblec=funbas(fspace,nodescheblecture);  %Chebyshev


%get polynomial coefficients
cequi=Bequi\ynodesequi;  %equidistant
cchebsli=Bchebsli\ynodeschebsli;  %chebychev
ccheblec=Bcheblec\ynodescheblec;

%approximate the function
yequi=funeval(cequi,fspace,x);
ychebsli=funeval(cchebsli,fspace,x);
ycheblec=funeval(ccheblec,fspace,x);

end
```

Function of first expected value:

```
function y= fivefct1(x)
  y=x.^4;
end
```

Function of second expected value:

```
function y= fivefct2(x)
  y=x.^6;
end
```

Function of third expected value:

```
function y= fivefct3(x)
  y=1./(1+x.^2);
end
```

The output of the Gaussian quadrature is the 3X5 matrix in the first line, while the output of Monte Carlo is the 3X4 matrix in the second line:

0.5000	0.7500	0.4167	0.4167	0.4000
0.2500	0.5625	0.3333	0.3125	0.2875
1.3333	1.4286	1.5686	1.5772	1.5704

0.3854	0.4240	0.3962	0.3976
0.2742	0.3063	0.2839	0.2837
1.5837	1.5585	1.5739	1.5724

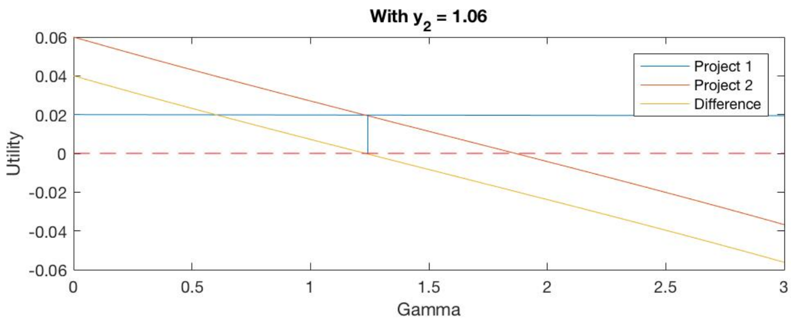
Gaussian Quadrature:
Row i uses function i, column j uses the j-th entry of the node vector. One may expect that the value converges to the true value. That might be the case for function 3. However, using function 1 and 2, there is a hump at nodes=3 and then converges to a value. The rest is as expected.

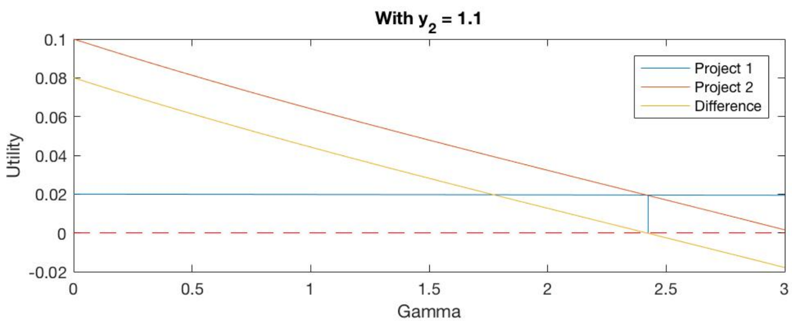
Monte Carlo:
Row i uses function i, column j uses the j-th entry of the n-vector. Again it seems to be converging to a value as n increases, but this time there is a hump in every function (for function 3 downwards) at n=1000.

Comparison:
As n increases or the number of nodes, (comparing the most right column for each method), the values are very close. However, for small n or few nodes, the methods yield different approximations.

2 Question 2

The code below can only be used after installing the CompEcon Toolbox for Matlab from this website. The approximate point where the household is indifferent between the two projects can be found via grid search. Graphically, the point where the two utility curves intersect in the point where the household becomes indifferent between the two projects. In both graphs, this point is marked by the vertical line going upwards from the horizontal line crossing through zero. Alternatively, this point can also be found where the squared difference comes close to zero. However, for a truly satisfying result, grid search is not sufficient. The root of the difference function has to be found numerically. Here, the Newton algorithm is applied. For this, the first derivative with respect to γ has to be computed. Since this is possible analytically, the derivation is





provided below.

$$u(c_t) = \frac{c_t^{1-\gamma} - 1}{1-\gamma} = \frac{c_t^{1-\gamma}}{1-\gamma} - \frac{1}{1-\gamma} \tag{1}$$

$$= \frac{e^{(1-\gamma)\ln(c_t)}}{1-\gamma} - \frac{1}{1-\gamma} \tag{2}$$

The first derivative with respect to γ can then be calculated:

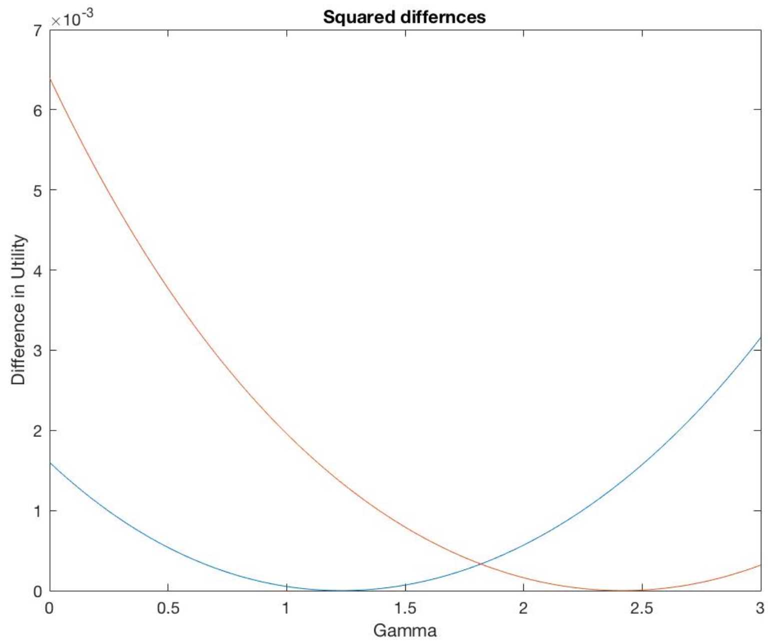
$$\frac{\partial u(c_t)}{\partial \gamma} = \frac{(-1)e^{(1-\gamma)\ln(c_t)}(1-\gamma) - (-1)e^{(1-\gamma)\ln(c_t)}}{(1-\gamma)^2} - \frac{0-(-1)*1}{(1-\gamma)^2} \tag{3}$$

$$= \frac{e^{(1-\gamma)\ln(c_t)} - e^{(1-\gamma)\ln(c_t)}(1-\gamma)}{(1-\gamma)^2} - \frac{1}{(1-\gamma)^2} \tag{4}$$

$$= \frac{e^{(1-\gamma)\ln(c_t)} - e^{(1-\gamma)\ln(c_t)}(1-\gamma) - 1}{(1-\gamma)^2} \tag{5}$$

$$= \frac{c_t^{1-\gamma} - c_t^{1-\gamma}(1-\gamma) - 1}{(1-\gamma)^2} \tag{6}$$

$$\tag{7}$$



Which can be used for the calculation of the Newton Algorithm. In the case that $\gamma = 1$

$$\frac{\partial u(c_t)}{\partial \gamma} = \frac{\partial \ln c_t}{\partial \gamma} = 0. \tag{8}$$

The output of the code provided above is then:

```

        Project 2 yields the greater expected payoff.
        Household will prefer to invest in project 1.
        As one can see clearly, changing y_2 changes
           the gamma at which both projects yield the
           same expected utility.

        In the first plot, gamma = 1.2424 produced the
            smallest difference.
        For a value close to this, the household will
           be indifferent between the two projects,
           given y_2 = 1.06

        In the second plot, gamma = 2.4242 produced
           the smallest difference.
        For a value close to this, the household will
```

```
be indifferent between the two projects, given y_2 = 1.1

        The Newton algorithm finds a root of the difference at gamma = 1.2424 , given y_2 = 1.06

        The Newton algorithm finds a root of the difference at gamma = 2.4242 , given y_2 = 1.1
```

Appendices

A Code to Question 2

All in one code (additional functions defined on bottom of script file):

```
close all;
clear;
clc;

y_1 = 1.02;
Var_ln_eta = (0.25)^2;
Mu_ln_eta = -Var_ln_eta/2;
y_2 = 1.06;
n = 11;                                              %11 nodes
[ln_eta,w]=qnwnorm(n,Mu_ln_eta,Var_ln_eta);          % Distribution of log(eta)
eta=exp(w'*ln_eta);                                  % Expectation of eta

%%%%%%%%%%%%%% Question 1 %%%%%%%%%%%%%%%

p_1=y_1;                                             % Expected Payoff of Project 1
p_2=y_2*eta;                                         % Expected Payoff of Project 2

if p_1 < p_2
    disp('Project 2 yields the greater expected payoff.');
elseif p_1 == p_2
```

```
    disp('Project 1 and project 2 yield the same expected payoff.');
else
    disp('Project 1 yields the greater expected payoff.');
end

%%%%%%%%%%%%%% Question 2 %%%%%%%%%%%%%%

gamma = 1.5;

u_1 = utility(y_1,gamma);
u_2 = w'*utility(exp(ln_eta)*y_2,gamma);

if u_1 < u_2
    disp('Household will prefer to invest in project 2.');
elseif u_1 == u_2
    disp('Household will be indifferent betwee project 1 and project 2.');
else
    disp('Household will prefer to invest in project 1.');
end

%%%%%%%%%%%%%% Question 3 %%%%%%%%%%%%%%

gamma = linspace(0,3,100);                              %Gamma is now a vector of different values (for plotting only)
y_2= [1.06 1.1];
u_1=nan(1,100);
u_2=nan(1,100);
difference = nan(2,100);
%Plot intersection point
figure('Name','PS5Q2Sub3_Utility')
for j=1:2
for i=1:100
u_1(1,i) = utility(y_1,gamma(1,i));
u_2(1,i) = w'*utility(exp(ln_eta)*y_2(1,j),gamma(1,i));
difference(j,i) = u_2(1,i)-u_1(1,i);
end
[Min,Index] = min(abs(difference(j,:)));

subplot(2,1,j)
```

```
plot(gamma,u_1,gamma,u_2,gamma,difference(j,:))
line([min(gamma),max(gamma)],[0,0],'Color','red','LineStyle','--')
line([gamma(1,Index),gamma(1,Index)],[0,u_2(1,Index)])
title(['With y_2 = ', num2str(y_2(1,j))])
legend('Project 1','Project 2','Difference')
xlabel('Gamma')
ylabel('Utility')
end

figure('Name','PS5Q2Sub3_Quad_Diff')
plot (gamma, (difference(1,:)).^2,gamma, (difference(2,:)).^2)
title('Squared differnces')
xlabel('Gamma')
ylabel('Difference in Utility')

%Find intersection via grid search
[Min1,Index1] = min(abs(difference(1,:)));
[Min2,Index2] = min(abs(difference(2,:)));

disp('As one can see clearly, changing y_2 changes the gamma at which both projects yield the same expected utility.');
fprintf(['\n In the first plot, gamma = ', num2str(gamma(1,Index1)),' produced the smallest difference. \n For a value close to this, the household will be indifferent between the two projects, given y_2 = 1.06 \n '] );
fprintf(['\n In the second plot, gamma = ', num2str(gamma(1,Index2)),' produced the smallest difference. \n For a value close to this, the household will be indifferent between the two projects, given y_2 = 1.1 \n'] );

%Find intersection point numerically using Newton. This is equal to finding
%the gamma for which the difference is equal to zero --> Root finding Problem

params = [ln_eta;w;y_1;y_2(1,1)];
f = @(x) Utility_Difference(x,params);
y = [gamma(1,Index1) gamma(1,Index2)];          % educated guess
cc =[0.1;0.1;1000];                             % criteria
```

```
fprintf(['\n The Newton algorithm finds a root of the difference at gamma = ', num2str(newton(f,y(1,1),cc)),' , given y_2 = 1.06 \n '] );

params = [ln_eta;w;y_1;y_2(1,2)];
f = @(x) Utility_Difference(x,params);
fprintf(['\n The Newton algorithm finds a root of the difference at gamma = ', num2str(newton(f,y(1,2),cc)),' , given y_2 = 1.1 \n '] );

%Newton
function [x,fx,ef,iter] = newton(f,x,cc)

% convergence criteria
tole = cc(1,1); told = cc(2,1); maxiter = cc(3,1);

% newton algorithm
for j = 1:maxiter
    [fx,dfx] = f(x);

    xp = x - dfx\fx;
    D = (norm(x-xp) <= tole*(1+norm(xp)) && norm(fx) <= told);
    if D == 1
        break;
    else
        x = xp;
    end
    break
end
ef = 0; if D == 1; ef = 1; end
iter = j;
end

%Function whose root if to be found
function [fx,dfx] = Utility_Difference(x,y)

weight=[y(1,1);y(2,1);y(3,1);y(4,1);y(5,1);y(6,1);y(7,1);y(8,1);y(9,1);y(10,1);y(11,1)];
rv=[y(12,1);y(13,1);y(14,1);y(15,1);y(16,1);y(17,1);y(18,1);y(19,1);y(20,1);y(21,1);y(22,1)];

y_1=y(23,1);
y_2=y(24,1);

```

```
fx = utility(y_1,x) - weight'*utility(exp(rv)*y_2,x);

dfx = derivative(y_1,x)-weight'*derivative(exp(rv)*y_2,x);

end

% Declare CRRA utility function
function u= utility(x,gamma)
    if gamma == 1
        u=log(x);
    else
        u=(x.^(1-gamma)-1)./(1-gamma);

    end
end

function dgu = derivative(x,gamma)
    if gamma == 1
          dgu = -0.00001*ones(length(x),1); %Should be zero but putting dgu = 0; yields an error
    else
         dgu=((x.^(1-gamma)-1)-(x.^(1-gamma)-1).*(1-gamma)-1)./((1-gamma).^2);
    end

end
```