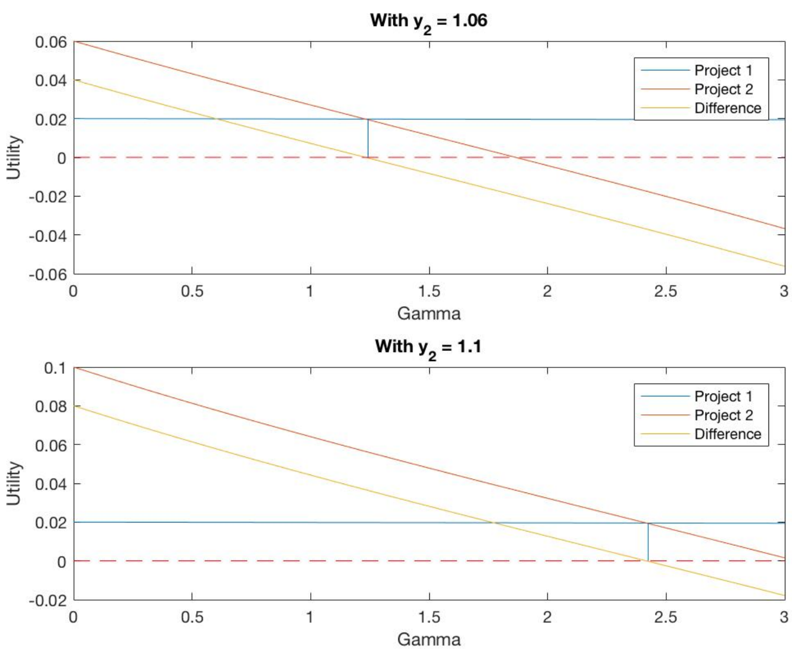
Answers to Problem Set 5
Group name: Ferienspass

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

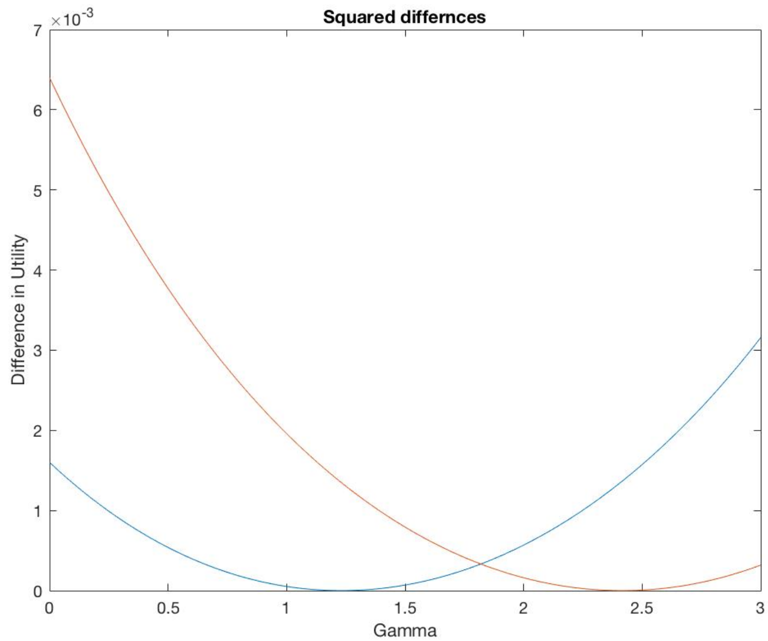
1 Question 1

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 1

Main code:

```
clear;
clc;
close all;
%PS5P1

%Monte Carlo Quadrature integration
```

```
%assume [-1,1]

%first of all, draw random numbers on uniform (-1,1)
a=-1;
b=1;
n1=100;
n2=1000;
n3=10000;
n4=50000;
x1=unifrnd(-1,1,n1,1);
x2=unifrnd(-1,1,n2,1);
x3=unifrnd(-1,1,n3,1);
x4=unifrnd(-1,1,n4,1);

%Polynomial approximation of the functions
%SPLINES for every draw of random numbers do
m1=3;
m2=4;
m3=5;
m4=6;
m5=7;
m=[m1;m2;m3;m4;m5];
ypsilon1=zeros(n1,5,3);
ypsilon2=zeros(n2,5,3);
ypsilon3=zeros(n3,5,3);
ypsilon4=zeros(n4,5,3);
ef1=zeros(5,3);
ef2=zeros(5,3);
ef3=zeros(5,3);
ef4=zeros(5,3);
func={@fivefct1;@fivefct2;@fivefct3};
%for every function and number of nodes do splines
%n=100
for i=1:5
  x1=unifrnd(-1,1,n1,1);
  for j=1:3
    ypsilon1(:,i,j)=spl(func{j},x1,m(i),a,b);
    ef1(i,j)=(b-a)/n1*sum(ypsilon1(:,i,j));
  end
end
%n=1000
for i=1:5
  x2=unifrnd(-1,1,n2,1);
  for j=1:3
    ypsilon2(:,i,j)=spl(func{j},x2,m(i),a,b);
    ef2(i,j)=(b-a)/n2*sum(ypsilon2(:,i,j));
```

```
  end
end
%n=10000
for i=1:5
  x3=unifrnd(-1,1,n3,1);
  for j=1:3
    ypsilon3(:,i,j)=spl(func{j},x3,m(i),a,b);
    ef3(i,j)=(b-a)/n3*sum(ypsilon3(:,i,j));
  end
end
%n=50000
for i=1:5
  x4=unifrnd(-1,1,n4,1);
  for j=1:3
    ypsilon4(:,i,j)=spl(func{j},x4,m(i),a,b);
    ef4(i,j)=(b-a)/n4*sum(ypsilon4(:,i,j));
  end
end

%Gaussian Quadrature
[xg1,w1]=lgwt(n1,a,b);
[xg2,w2]=lgwt(n2,a,b);
[xg3,w3]=lgwt(n3,a,b);
%[xg4,w4]=lgwt(n4,a,b);
ypsilong1=zeros(n1,5,3);
ypsilong2=zeros(n2,5,3);
ypsilong3=zeros(n3,5,3);
ypsilong4=zeros(n4,5,3);
efg1=zeros(5,3);
efg2=zeros(5,3);
efg3=zeros(5,3);
efg4=zeros(5,3);
%n=100
for i=1:5
  for j=1:3
    ypsilong1(:,i,j)=spl(func{j},xg1,m(i),a,b);
    efg1(i,j)=(b-a)/n1*sum(ypsilong1(:,i,j).*w1);
  end
end
%n=1000
for i=1:5
  for j=1:3
    ypsilong2(:,i,j)=spl(func{j},xg2,m(i),a,b);
    efg2(i,j)=(b-a)/n2*sum(ypsilong2(:,i,j).*w2);
  end
end
```

```
%n=10000
for i=1:5
  for j=1:3
    ypsilong3(:,i,j)=spl(func{j},xg3,m(i),a,b);
    efg3(i,j)=(b-a)/n3*sum(ypsilong3(:,i,j).*w3);
  end
end
%n=50000
%for i=1:5
%  for j=1:3
%    ypsilong4(:,i,j)=spl(func{j},xg4,m(i),a,b);
%    efg4(i,j)=(b-a)/n4*sum(ypsilong4(:,i,j).*w4);
%  end
%end
```

Spline function:

```
function yapspl=spl(fct,x,m,a,b)
  c=max(m-1,2);
  fspacespl=fundefn('spli',c,-1,1,m);
  distance=(b-a)/(m-1);
  xspl=zeros(m,1);
  yspl=zeros(m,1);
  for i=1:m
    xspl(i)=a+(i-1)*distance;
    yspl(i)=fct(xspl(i));
  end

  %calculate the matrix of basis functions
  Bspl=funbas(fspacespl,xspl);

  %get polynomial coefficients
  cspl=Bspl\yspl;

  %approximate the function
  yapspl=funeval(cspl,fspacespl,x);
end
```

Function of first expected value:

```
function y= fivefct1(x)
  y=x.^5;
end
```

Function of second expected value:

```
function y= fivefct2(x)
  y=x.^7;
```

```
end
```

Function of third expected value:

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

Monte Carlo integration:

```
%backup monte carlo
function [integral_dx] = integral_dx( f,a,b )
%integrate function (f) using monte carlo method

x2=linspace(a,b,1000);
syms z % zero vector holder to find max y value
z = zeros(size(x2));

z = f(x2);

y = f(b).*rand(1,1000);

x = rand(1,1000);

h=0; % counters
n= 0;
%if you want to see visual representation just un-
    commnet plot lines

plot(x2,z); hold on;
plot(x,y,'x')
count = 0;
for k=1:numel(x);

      if y(k) <= exp(x(k))+1;
          count= count +1;
      end

end

count
integral_dx = count/numel(x) * max(z) * (b-a);
end
```

Gaussian Quadrature:

```
function [x,w]=lgwt(N,a,b)
N=N-1;
```

```
N1=N+1; N2=N+2;

xu=linspace(-1,1,N1)';

% Initial guess
y=cos((2*(0:N)'+1)*pi/(2*N+2))+(0.27/N1)*sin(pi*xu*N/N2);

% Legendre-Gauss Vandermonde Matrix
L=zeros(N1,N2);

% Derivative of LGVM
Lp=zeros(N1,N2);

% Compute the zeros of the N+1 Legendre Polynomial
% using the recursion relation and the Newton-Raphson method

y0=2;

% Iterate until new points are uniformly within epsilon of old points
while max(abs(y-y0))>eps


    L(:,1)=1;
    Lp(:,1)=0;

    L(:,2)=y;
    Lp(:,2)=1;

    for k=2:N1
        L(:,k+1)=( (2*k-1)*y.*L(:,k)-(k-1)*L(:,k-1) )/k;
    end

    Lp=(N2)*( L(:,N1)-y.*L(:,N2) )./(1-y.^2);

    y0=y;
    y=y0-L(:,N2)./Lp;

end

% Linear map from[-1,1] to [a,b]
x=(a*(1-y)+b*(1+y))/2;
x=flipud(x);
```

```
% Compute the weights
w=(b-a)./((1-y.^2).*Lp.^2)*(N2/N1)^2;
w=flipud(w);
end
```

B 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
```