Answers to Problem Set 3
Group name: Ferienspass

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

1 Question 1

Newton Optimization Algorithm

```
function [root,sol]=Newton_opt(fun,x_start,eps,delta,
    maxiter)
i=0;
x=zeros(25,1);
x(1)=x_start;
%to get in while
absdiff=1;
check=0;
%check input parameters
if (eps<=0) || (delta<=0) || (maxiter<=0)
    disp('invalid input parameters')
    return;
end
%function at x_start already in optimum?
syms a
f=fun(a);
grad=eval(subs(diff(f,a,1),a,x(1)));
if (grad==0)
    root=x(1);
    return;
end
while (i<maxiter) && (absdiff>check)
  i=i+1;
  grad=eval(subs(diff(f,a,1),a,x(i)));
  he=eval(subs(diff(f,a,2),a,x(i)));
  x(i+1)=x(i)-he*grad;
  absdiff=abs(x(i)-x(i+1));
  check=eps*(1+abs(x(i+1)));
end
crit=delta*(1+abs(fun(x(i+1))));
val=abs(eval(subs(diff(f,a,1),a,x(i+1))));
if (i<maxiter) && (val<=crit)
```

```
    disp('success');
    sol=true;
    root=x(i+1);
else
    disp('failure');
    sol=false;
    root=-Inf;
end
disp(['number iterations: ',int2str(i)])
end
```

The functions:

```
function y=f1(x)
y=2.*x.^3-x.^2-3.*x+2;
end
```

```
function y=f2(x)
y=-x.*exp(-x);
end
```

2 Question 2

2.1 Subquestion 1

From the slides, we get the following conditions for the variables on the balanced growth path

$$s_k f(k^*, h^*) - (\delta_k + g + n + ng)\, k^* = 0 \tag{1}$$

$$s_h f(k^*, h^*) - (\delta_h + g + n + ng)\, h^* = 0. \tag{2}$$

solving for k^* and h^*, respectively, gives

$$k^* = \frac{s_k f(k^*, h^*)}{(\delta_k + g + n + ng)} \tag{3}$$

$$h^* = \frac{s_h f(k^*, h^*)}{(\delta_h + g + n + ng)}. \tag{4}$$

However, since $f(k^*, h^*)$ depends on the variables that we try to solve for, this is not the final form yet.
Since we are assuming $F(K_t, H_t, A_t L_t) = K_t^\alpha H_t^\beta (A_t L_t)^{1-\alpha-\beta}$, we can further solve the expression

$$f(k^*, h^*) = \frac{F(K_t, H_t, A_t L_t)}{A_t L_t} = \frac{K_t^\alpha H_t^\beta (A_t L_t)^{1-\alpha-\beta}}{A_t L_t} = \left(\frac{K_t}{A_t L_t}\right)^\alpha \left(\frac{H_t}{A_t L_t}\right)^\beta. \tag{5}$$

By definition of k^* and h^*, this is gives

$$f\left(k^*, h^*\right) = k_t^\alpha h_t^\beta. \tag{6}$$

Now, inserting this into 3 and 4

$$k^* = \frac{s_k k_t^\alpha h_t^\beta}{\left(\delta_k + g + n + ng\right)} \Leftrightarrow k^* = \left(\frac{s_k h_t^\beta}{\left(\delta_k + g + n + ng\right)}\right)^{\frac{1}{1-\alpha}} \tag{7}$$

$$h^* = \frac{s_h k_t^\alpha h_t^\beta}{\left(\delta_h + g + n + ng\right)} \Leftrightarrow h^* = \left(\frac{s_h k_t^\alpha}{\left(\delta_h + g + n + ng\right)}\right)^{\frac{1}{1-\beta}}. \tag{8}$$

It becomes evident, that k^* and h^* are a function of each other. However, we can simply substitute and then solve for the expression depending on parameters only

$$k^* = \left(\frac{s_k \left(\left(\frac{s_h k_t^\alpha}{\left(\delta_h + g + n + ng\right)}\right)^{\frac{1}{1-\beta}}\right)^\beta}{\left(\delta_k + g + n + ng\right)}\right)^{\frac{1}{1-\alpha}} \tag{9}$$

$$h^* = \left(\frac{s_h \left(\left(\frac{s_k h_t^\beta}{\left(\delta_k + g + n + ng\right)}\right)^{\frac{1}{1-\alpha}}\right)^\alpha}{\left(\delta_h + g + n + ng\right)}\right)^{\frac{1}{1-\beta}}. \tag{10}$$

which can now be solved for the respective variable. For k^*:

$$\begin{aligned}
k^* &= \left(\frac{s_k \left(s_h k_t^\alpha\right)^{\frac{\beta}{1-\beta}}}{\left(\delta_h + g + n + ng\right)^{\frac{\beta}{1-\beta}} \left(\delta_k + g + n + ng\right)}\right)^{\frac{1}{1-\alpha}} \\
&= \left(\frac{s_k s_h^{\frac{\beta}{1-\beta}}}{\left(\delta_h + g + n + ng\right)^{\frac{\beta}{1-\beta}} \left(\delta_k + g + n + ng\right)}\right)^{\frac{1}{1-\alpha}} k_t^{\frac{\alpha\beta}{1-\beta-\alpha+\alpha\beta}} \\
&= \left(\frac{s_k s_h^{\frac{\beta}{1-\beta}}}{\left(\delta_h + g + n + ng\right)^{\frac{\beta}{1-\beta}} \left(\delta_k + g + n + ng\right)}\right)^{\frac{1-\beta-\alpha+\alpha\beta}{(1-\alpha)^2+\alpha\beta-\beta}}
\end{aligned}$$

For h^*:

$$
\begin{aligned}
h^* &= \left(\frac{s_h \left(s_k h_t^{\beta} \right)^{\frac{\alpha}{1-\alpha}}}{\left(\delta_k + g + n + ng\right)^{\frac{\alpha}{1-\alpha}} \left(\delta_h + g + n + ng\right)} \right)^{\frac{1}{1-\beta}} \\
&= \left(\frac{s_h s_k^{\frac{\alpha}{1-\alpha}}}{\left(\delta_k + g + n + ng\right)^{\frac{\alpha}{1-\alpha}} \left(\delta_h + g + n + ng\right)} \right)^{\frac{1}{1-\beta}} h_t^{\frac{\alpha\beta}{1-\beta-\alpha+\alpha\beta}} \\
&= \left(\frac{s_h s_k^{\frac{\alpha}{1-\alpha}}}{\left(\delta_k + g + n + ng\right)^{\frac{\alpha}{1-\alpha}} \left(\delta_h + g + n + ng\right)} \right)^{\frac{1-\beta-\alpha+\alpha\beta}{(1-\beta)^2+\alpha\beta-\alpha}} .
\end{aligned}
$$

Thus, the solution vector becomes

$$
\begin{bmatrix} k^* \\ h^* \end{bmatrix}' = \begin{bmatrix} \left(\frac{s_k s_h^{\frac{\beta}{1-\beta}}}{(\delta_h+g+n+ng)^{\frac{\beta}{1-\beta}} (\delta_k+g+n+ng)} \right)^{\frac{1-\beta-\alpha+\alpha\beta}{(1-\alpha)^2+\alpha\beta-\beta}} \\ \left(\frac{s_h s_k^{\frac{\alpha}{1-\alpha}}}{(\delta_k+g+n+ng)^{\frac{\alpha}{1-\alpha}} (\delta_h+g+n+ng)} \right)^{\frac{1-\beta-\alpha+\alpha\beta}{(1-\beta)^2+\alpha\beta-\alpha}} \end{bmatrix}'
$$

2.6

All questions that lie between the first and this one are excluded from this document since they only involved programming exercises.

The path to be interpreted:

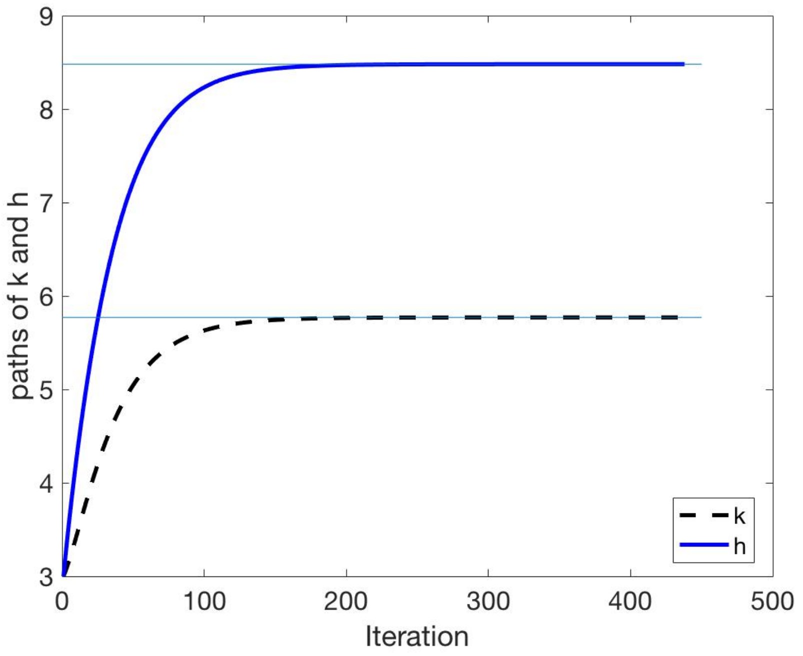


Figure 1: Paths of k^* and h^*

Both variables converge to the solution at the same time/speed. Optimal allocation human capital exceeds the optimal level of capital in all iterations.
A possible interpretation: That the balanced growth path level of human capital is greater than the balanced growth path level of physical capital could have two (simultaneous) reasons:

1. Human capital has a greater marginal product compared to physical capital

2. Human capital has a lower depreciation rate than physical capital

The reasoning behind this it that if an additional unit of human capital yields more than an additional unit of physical capital, and lasts longer than it as well, and investment in human capital is preferred over an investment in physical capital. This goes on this way until a point is reached, where an investment in human capital has the same marginal-product-to-depreciation-rate as physical capital. The investment in the two goods will increase such that their levels always give the same marginal-product-to-depreciation-rate (which implies a constant ratio of human to physical capital (which can be seen in the graph as well)) until the entire income is spent. At this point, the balanced growth path level is reached.

3 Question 3

3.1

To begin with, recall two properties of the logistic function characterising y_n:

$$y(x) := \Lambda(x) = \frac{1}{1+exp(-x)} = \frac{exp(x)}{1+exp(x)} \tag{*}$$

Applying the quotient rule, yields the partial derivative w.r.t. x

$$\begin{aligned}
\lambda(x) &= \frac{exp(x)}{(1+exp(x))^2} \\
&= \frac{exp(x)}{1+exp(x)} \frac{1}{1+exp(x)} \\
&= \Lambda(x)\Lambda(-x) && \text{(using (*))} \\
&= \Lambda(x)(1-\Lambda(x)) && \text{(by symmetry around zero)} \\
&= y(x)(1-y(x)) && \text{(**)}
\end{aligned}$$

The log-likelihood function is given by

$$\mathbb{E}(\beta) = -\sum_{n=1}^{N} t_n log(y_n) + (1-t_n)log(1-y_n)$$

Accordingly, the first order condition w.r.t. coefficient β_i is given by

$$\begin{aligned}
\frac{\partial \mathbb{E}(\beta)}{\partial \beta_i} &= -\sum_{n=1}^{N} \frac{(t_n - \Lambda(X_n'\beta))\lambda(X_n'\beta)X_{ni}}{\Lambda(X_n'\beta)(1-\Lambda(X_n'\beta))} && \text{(for } i = 1, ..., K) \\
&= -\sum_{n=1}^{N} (t_n - \Lambda(X_n'\beta))X_{ni} && \text{(using (**)} \\
&= \sum_{n=1}^{N} (y_n - t_n)X_{ni} && \text{(by definition of } y_n) \\
&= 0 && \text{(for i = 1,...,K)}
\end{aligned}$$

Putting these K likelihood equations together yields the result

$$\begin{aligned}
\nabla\mathbb{E}(\beta) &= \sum_{n=1}^{N} (y_n - t_n)X_n' \\
&= (y-t)'X
\end{aligned}$$

In order to characterise the Hessian, take derivative w.r.t. β_j

$$\begin{aligned}
\frac{\partial \mathbb{E}(\beta)}{\partial \beta_i \partial \beta_j} &= \sum_{n=1}^{N} \lambda(X_n'\beta) X_{ni} X_{nj} && \text{(for any } i,j = 1,...,K\text{)} \\
&= \sum_{n=1}^{N} \Lambda(X_n'\beta)(1 - \Lambda(X_n'\beta)) X_{ni} X_{nj} \\
&= \sum_{n=1}^{N} y_n (1 - y_n) X_{ni} X_{nj} && \text{(by definition of } y_n\text{)}
\end{aligned}$$

which represents the (i,j)th element of the Hessian. In summary, we have

$$\begin{aligned}
H &= \sum_{n=1}^{N} y_n (1 - y_n) X_n X_n' \\
&= X'RX
\end{aligned}$$

3.2

It is well known that the log-likelihood function is globally concave in β because any contribution n is globally concave in the parameters and the sum of concave functions is again concave. Multiplying by (-1) yields a globally convex function in β. Moreover, any real number is feasible for β_i and the set of real numbers is convex. To sum up, a globally convex function is minimised over a convex set, i.e., we face a convex minimisation problem. If a local minimum exists, then it is also a global minimum point. It furthermore implies that the likelihood equations uniquely define β (except for a very special case).

3.3

3.4

Masterfile for the following subquestions:

```
%% Problem set 3; problem 3: logit estimation
%% 3.3
% Load data set
filename = 'MRW92QJE-data.xlsx';
sheet = 1;
%[num, txt, raw] = xlsread(filename, sheet);
OECDdummy = xlsread(filename, sheet, 'E2:E122');
gdp85pad = xlsread(filename, sheet, 'G2:G122');
school = xlsread(filename, sheet, 'K2:K122');
X = [OECDdummy, gdp85pad, school];
nrow0 = size(X,1);
% Delete rows with NaN-values
X(any(isnan(X), 2), :) = [];
nrow1 = size(X,1);
```

```
diffrow = nrow0 - nrow1;
fprintf( 'N.b.: %g observations deleted due to missing
    values! \n', diffrow);

%% 3.4
% Dependent variable
t = X(:,1);
% Independent variables
const = ones(nrow1,1);
lgdp85pad = log(X(:,2));
lschool = log(X(:,3)/100);
X = [const, lgdp85pad, lschool];

%% 3.5
beta0 = zeros(3,1);
maxiter = 200;
tol = 0.00001;
% (a) Newton
[beta, iter, t] = mynewton(t, X, beta0, maxiter, tol);
beta
iter
t

% (b) Broyden's mthod
[beta, iter, t] = broyden(t, X, beta0, maxiter, tol);
beta
iter
t

% (c) Fixed point iteration
[beta, iter, t] = fixedpoint(t, X, beta0, maxiter, tol
    );
beta
iter
t

% (d) fsolve

%% 3.6 see 3.5 (b)

%% 3.7 Standardization
M = max(X);
m = min(X);
maxmin = M - m;

for j = 1:nrow1
```

```
    % Standardize lgdp85adult
    X(j,2) = ( X(j,2) - m(2) )/maxmin(2);
    % Standardize lschool
    X(j,3) = ( X(j,3) - m(3) )/maxmin(3);
end

beta0 = [-10; 10; 1];
[beta, iter, t] = broyden(t, X, beta0, maxiter, tol);
beta
iter
t

%% 3.8
% (a) Newton's method
[beta, iter, t] = broyden(t, X, beta0, maxiter, tol);
beta
iter
t

%% 3.9 tic-toc is already implemented above
```

3.5

1. Newton's method converges to $\beta = (-19.9159 \quad 2.7321 \quad 2.0589)'$ after 9 iterations and 0.0142 seconds.

2. Broyden's method does not converge.

3. For any dampening factor $\lambda > 0$, the fixed point iteration did not converge.

Matlab Code

Newton's Algorithm:

```
function [beta, iter, t] = mynewton( t, X, beta0,
    maxiter, tol )
% Newton's method: algorithm using exact Hessian
% function [beta, iter] = newton(t, X, beta0, maxiter,
     tol)

%% (0) Initialise
tic
iter = 0;
beta = beta0;
exit_flag = 0;

%% (1) Compute gradient and Hessian, then iterate
```

```
for j = 1:maxiter
    iter = j;
    y = sigmoid(X*beta);
    grad = X'*(y - t);
    h1 = y.*(1-y);
    R = diag(h1);
    Hess = X'*R*X;
    iHess = inv(Hess);
    p = - iHess*grad;
    % For clarity, display iteration steps:
    fprintf( 'Step made in iteration %g: \n', iter )
    p'
    beta_new = beta + p;
    %% (2) Check stopping criterion
    check = ( norm( p ) <= tol*( 1+norm(beta_new) ) &&
        norm(grad) <= tol );
    if check == 1
        exit_flag = 1; break;
    else
        beta = beta_new;
    end
end
%% (3) Report success or failure
if exit_flag == 0
    fprintf( 'Algorithm has not converged after %g
       iterations\n That is bad luck, my friend!\n',
       maxiter)
end
toc
t = toc;
```

Broyden's Algorithm:

```
function [beta, iter, t] = broyden(t, X, beta0,
   maxiter, tol)
% Broyden's algorithm: Quasi-Newton method using
   approximated Hessian
% function [beta, iter] = broyden(t, X, beta0, maxiter
   , tol)

%% (0) Initialise
tic
iter = 0;
beta = beta0;
% Evaluate likelihood equations for first guess
y = sigmoid(X*beta);
```

```
grad = X'*(y - t);
% Trivial initial guess for Hessian: I(K)
ncol = size(X,2);
Hess = eye(ncol);
Hess = inv(Hess); % Calculate inverse for first
    iteration

exit_flag = 0;
%% (1) Compute next iterate
for j = 1:maxiter
    iter = j;
    p = - Hess*grad;
    beta_new = beta + p;

    %% (2) Update approximate Hessian
    dbeta = beta_new - beta;
    y = sigmoid(X*beta_new);
    grad_new = X'*(y - t);
    dgrad = grad_new - grad;
    a = dgrad - Hess*dbeta;
    b = dbeta'*dbeta;
    c = a*dbeta';
    Hess = Hess + c/b;
    condition = cond(Hess);
    fprintf( 'The condition number of the Jacobian in
        iteration %g \n is %.4f \n', iter, condition)
    %% (3) Check stopping criterion
    check = ( norm( p ) <= tol*( 1+norm(beta_new) ) &&
         norm(grad_new) <= tol );
    beta = beta_new;
    if check == 1
        exit_flag = 1; break;
    else
        grad = grad_new;
    end
end
%% (4) Report success or failure
if exit_flag == 0
    fprintf( 'Algorithm has not converged after %g
        iterations\n That is bad luck, my friend! \n',
        maxiter)
end
toc
t = toc;
```

Fixed Point Algorithm:

```matlab
function [beta, iter, t] = fixedpoint( t, X, beta0,
    maxiter, tol)
% Fixed point iteration <=>: gradient method/ steepest
     descent
% lambda is not obtained via linesearch but fixed at
    the beginning
% function [beta, iter] = fixedpoint( t, X, beta0,
    maxiter, tol, lambda0)

%% (0) Initialise
tic
iter = 0;
beta = beta0;
exit_flag = 0;
lambda = input( 'Please insert a dampening factor
    lambda: ' );

%% (1) Compute gradient and Hessian, then iterate
for j = 1:maxiter
    iter = j;
    y = sigmoid(X*beta);
    grad = X'*(y - t);
    p = - lambda*grad;
    % For clarity, display iteration steps:
    fprintf( 'Step made in iteration %g: \n', iter )
    p'
    beta_new = beta + p;

    %% (2) Check stopping criterion
    check = ( norm( p ) <= tol*( 1+norm(beta_new) ) &&
        norm(grad) <= tol );
    if check == 1
        exit_flag = 1; break;
    else
        beta = beta_new;
    end
end
%% (3) Report success or failure
if exit_flag == 0
    fprintf( 'Algorithm has not converged after %g
        iterations\n That is bad luck, my friend!\n',
        maxiter)
end
toc
t = toc;
```

Sigmoid:

```
function y = sigmoid( z )
y = 1 ./ ( 1 + exp(-z) );
```

3.6

The function broyden.m used in the previous question returns the condition number of the Jacobian at every iteration. The condition number is consistently extremely large. Large values indicate that the problem is ill-conditioned. This is why the iteration did not converge.

3.7

Standardisation significantly reduces the condition of the Jacobian. Unfortunately, the condition is still far from being optimal and the Broyden's method does not converge again.

3.8

Unfortunately, the Newton's method did not converge again.

3.9

Newton's method is the most time-consuming algorithm since it requires to evaluate and invert the Hessian at every iteration. Unfortunately, other algorithms often failed to converge. Theoretically, Broyden's method is faster because only one inversion is necessary. Afterwards, it is possible to approximate the Hessian using information already available. However, Broyden's method is more imprecise and, therefore, requires more iterations for convergence than Quasi-Newton methods that directly update the inverse of the Hessian such as BFGS and DGP.

4 Question 4

4.1 Analytical Solution

From the Euler equation and the budget constraint we get that

$$c_t = (\beta(1+r))^{-\frac{1}{\theta}} c_{t+1} \tag{11}$$

$$\sum c_t \left(\frac{1}{1+r}\right)^t = \sum w_t \left(\frac{1}{1+r}\right)^t + a_0 \tag{12}$$

Since we can express all c_t as a function of c_0, we can write

$$c_0 \sum \left(\beta^{-\frac{1}{\theta}} (1+r)^{\frac{\theta-1}{\theta}}\right)^t = \sum w_t \left(\frac{1}{1+r}\right)^t + a_0 \tag{13}$$

which allows to compute all c_t after solving for c_0

$$c_0 = \frac{\sum w_t \left(\frac{1}{1+r}\right)^t + a_0}{\sum \left(\beta^{-\frac{1}{\theta}} (1+r)^{\frac{\theta-1}{\theta}}\right)^t} \tag{14}$$

$$c_{t+1} = c_t \left(\beta(1+r)\right)^{\frac{1}{\theta}} . \tag{15}$$

$$\tag{16}$$

This can now be implemented into an algorithm by using a for-loop.

4.2 Algorithm

```
%PS3 Problem 4
%No T is given, so declare it at the beginning of the
   program, and for any
%theta
clear;
close all;
clc;
T=20;
theta=1;
%variable initialization (fixed)
time=linspace(0,T,T+1); %just for plot
w=zeros(T+1,1);
w(1)=10;
beta=0.99;
r=0.05;
%these variables will be determined
a=zeros(T+1,1);
c=zeros(T+1,1);
%computation (could also be written as a function)
num=zeros(T+1,1);
denom=zeros(T+1,1);
if theta==1
    factor=beta*(1+r);
else
    factor=(1+r)^((1-theta)/theta)*beta^(1/theta);
end
for i=0:T
  num(i+1)=w(i+1)*(1+r)^(-i);
  denom(i+1)=(factor)^i;
end
c(1)=sum(num)/sum(denom);
for i=1:T
  a(i+1)=a(i)*(1+r)+w(i)-c(i);
  c(i+1)=c(i)*(beta*(1+r))^(1/theta);
end
figure
plot(time,c,time,w,time,a);
legend('C','W','A')
acheck=a(end);
ccheck=c(end);
if (round(ccheck*1000)/1000==round(acheck*(1+r)*1000)
    /1000)
    display('True --> a_{T+1}= 0')
elseif (round(ccheck*1000)/1000>round(acheck*(1+r)
    *1000)/1000)
  display('a_{T+1}<0')
  disc=a(end);
  subst=zeros(T+1,1);
  for i=0:T
    subst(i+1)=(1+r)^i*((beta*(1+r))^(i/theta))^T-i;
  end
  d=sum(subst);
  c(1)=c(1)-d;
  for i=1:T
    c(i+1)=c(i)*(beta*(1+r))^(i/theta);
    a(i+1)=a(i)*(1+r)+w(i)-c(i);
  end
else
    display('error, a_{T+1}>0')
end
```

4.3 What to do if $\theta = 1$

in the case that $\theta = 1$, we get

$$\frac{c_t^{1-1}-1}{1-1} = \frac{1-1}{0} = \frac{0}{0} \ .$$

In this case, an application of L'Hôpital's rule has to be applied

$$\frac{f(x)}{g(x)} = \frac{\left(\frac{\partial f(x)}{\partial x}\right)}{\left(\frac{\partial g(x)}{\partial x}\right)} \ .$$

We now have to use

$$f(\theta) = c_t^{1-\theta} - 1 \tag{17}$$
$$g(\theta) = 1 - \theta \tag{18}$$

giving

$$\frac{\partial f(\theta)}{\partial\theta} = (-1)\ln(c_t)\exp((1-\theta)\ln(c_t)) \tag{19}$$

$$\frac{\partial g(\theta)}{\partial\theta} = -1 \tag{20}$$

$$\frac{\left(\frac{\partial f(x)}{\partial x}\right)}{\left(\frac{\partial g(x)}{\partial x}\right)} = \ln(c_t)\exp((1-\theta)\ln(c_t)) \tag{21}$$

in the limit

$$\lim_{\theta\to 1}\frac{\left(\frac{\partial f(x)}{\partial x}\right)}{\left(\frac{\partial g(x)}{\partial x}\right)} = \ln(c_t) \tag{22}$$

which can then be used in the maximization problem to obtain a new Euler equation.

5 Question 5

1. First of all, let us short notation in terms of $r_L := r_{low}$ and $r_H := r_{high}$. In order to check concavity of the objective function, consider its SOC:

$$-\gamma p\,(w_0(1+r^f+\alpha(r_L-r^f)))^{-(1+\gamma)}\,(r_L-r^f)^2+-\gamma(1-p)\,(w_0(1+r^f+\alpha(r_H-r^f)))^{-(1+\gamma)}\,(r_H-r^f)^2$$

Obviously its sign is ambiguous. Therefore, even though the (two) inequality constraints are (weakly) concave and no equality constraints arise, the first "convexity-condition" is not satisfied generally. That is, KTK-conditions are just necessary (as usual) but not sufficient for an optimum.

2. a)

FOC:

$$p\,(w_0(1+r^f+\alpha(r_L-r^f)))^{-\gamma}\,(r_L-r^f)+(1-p)\,(w_0(1+r^f+\alpha(r_H-r^f)))^{-\gamma}\,(r_H-r^f)=0 \tag{1}$$

Assume the opposite, i.e., that the optimal portfolio share α^* (the value of α for which the equation above holds) depends on initial wealth w_0. Denote this maximizer contingent on w_0 by $\alpha(w_0)$. As the optimal portfolio share responds on w_0, the first order derivative of the LHS of (1) (setting $\alpha = \alpha(w_0)$) with respect to wealth should be equal zero. Namely, the total differentiation of the FOC w.r.t. α and w_0:

$$\begin{aligned}\frac{\mathrm{d}\mathbb{E}u(w_1)}{\mathrm{d}\alpha\,\mathrm{d}w_0}= & -\gamma w_0^{-(1+\gamma)}\,p\,(1+r^f+a(r_L-r^f))^{-\gamma}\,(r_L-r^f)+w_0^{-\gamma}\,p\,\alpha'(w_0)\,(r_L-r^f)^2\\ & -\gamma w_0^{-(1+\gamma)}\,(1-p)\,(1+r^f+a(r_H-r^f))^{-\gamma}\,(r_H-r^f)+w_0^{-\gamma}\,p\,\alpha'(w_0)\,(r_H-r^f)^2=0.\end{aligned}$$

Clearly, this equality can be rearranged as follows:

$$w_0^{-\gamma}\,p\,\alpha'(w_0)\,(r_L-r^f)^2+w_0^{-\gamma}\,p\,\alpha'(w_0)\,(r_H-r^f)^2$$
$$=\tfrac{1}{\gamma}w_0^{-(1+\gamma)}\,p\,(1+r^f+\alpha(r_L-r^f))^{-\gamma}\,(r_L-r^f)+\tfrac{1}{\gamma}w_0^{-(1+\gamma)}\,(1-p)\,(1+r^f+\alpha(r_H-r^f))^{-\gamma}\,(r_H-r^f)$$

and now, the RHS of this equality corresponds to the FOC multiplied by γw_0^{-1} which has to be still equal zero. Thus, dividing the equality by the LHS (except $\alpha'(w_0)$) yields

$$\alpha'(w_0) = \frac{0}{w_0^{-\gamma}\,p\,(r_L-r^f)^2+w_0^{-\gamma}\,p\,(r_H-r^f)^2} = 0$$

which contradicts that the optimal portfolio share depends on w_0.

Alternatively, notice that the initial objective (which we aim to maximize) can be divided by w_0^{ϕ} and thus, wealth cancels out. Clearly, the optimal portfolio shares for the old resp. the new problem, will coincide.

b) The associated FOC becomes:

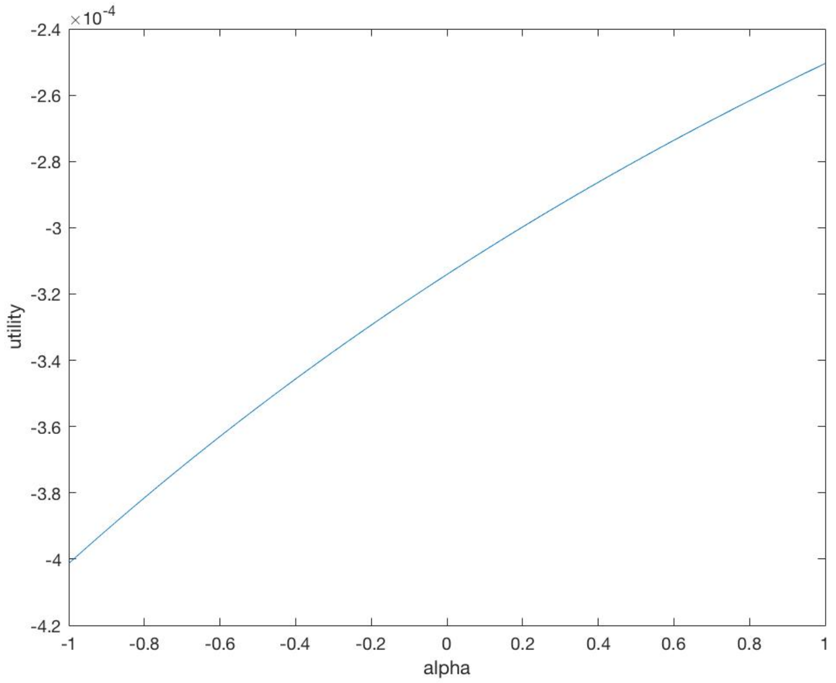
$$p\,(1+r^f+\alpha(r_L-r^f))^{\phi-1}\,(r_L-r^f)+(1-p)\,(1+r^f+\alpha(r_H-r^f))^{\phi-1}\,(r_H-r^f)=0 \tag{2}$$

using $(r_L-r^f) = -(r_H-r^f) = -0.1$ and rearranging we obtain

$$p^{\frac{1}{\phi-1}}\,(1+r^f+\alpha\,(-0.1)) = (1-p)^{\frac{1}{\phi-1}}\,(1+r^f+\alpha 0.1) \tag{3}$$

$$\Leftrightarrow \alpha = 10\,(1+r^f)\,\frac{p^{\frac{1}{\phi-1}}-(1-p)^{\frac{1}{\phi-1}}}{p^{\frac{1}{\phi-1}}+(1-p)^{\frac{1}{\phi-1}}} = 2.73308... \tag{4}$$

evaluated at $p=0.1$, $r^f=0.02$ and $\phi=-3$.



3. a) Intuitively , $\alpha \geq 0$ prevents the case where the household supplies the portfolio and $\alpha \leq 1$ rules out that the household borrows money in order to invest more in the portfolio.

b)

```
clear;
close all;

% Variables

phi=-3;
prob_low=0.1;
prob_high=1-prob_low;
alpha_upper=1;
alpha_lower=0;
w_init=10;
r_f=0.02;
r_high=0.12;
r_low=-0.08;

% Apply fminbnd
fun= @(alpha) (-utility(alpha,phi,prob_low,prob_high,
    w_init,r_f,r_high,r_low));
alpha_fminbnd = fminbnd(fun,alpha_lower,alpha_upper);

% Apply fmincon
A=[];
b=[];
Aeq=[];
beq=[];
lb=alpha_lower;
ub=alpha_upper;
alpha_fmincon = fmincon(fun,0,A,b,Aeq,beq,lb,ub);

% Plot Utility and Marginal Utility
alpha_v=nan(100,1);
a=nan(100,1);
a_marg=nan(100,1);

for i=1:100
   a(i,1)=utility(i/100,phi,prob_low,prob_high,w_init,
       r_f,r_high,r_low);
   a_marg(i,1)=marginal_utility(i/100,phi,prob_low,
       prob_high,w_init,r_f,r_high,r_low);
   alpha_v(i,1)=i/100;
end

figure
```

```
subplot(2,1,1); plot(alpha_v,a);
hline=refline((a(100,1)-a(1,1)),a(1,1));
hline.Color = 'r';
xlabel('alpha');
ylabel('utility');
legend('utility','Location','northwest');
subplot(2,1,2); plot(alpha_v,a_marg);
hline=refline((a_marg(100,1)-a_marg(1,1)),a_marg(1,1))
    ;
hline.Color = 'r';
xlabel('alpha');
ylabel('marginal utility');
legend('marginal utility','Location','northeast');

disp(['fminbnd solution at alpha=', num2str(
    alpha_fminbnd), '. fmincon solution at alpha=',
    num2str(alpha_fmincon)]);

% Define functions for utility and marginal utility
function util=utility(alpha,phi,prob_low,prob_high,
    w_init,r_f,r_high,r_low)
r=prob_high*r_high+prob_low*r_low;
util=(1/phi)*(w_init*(1+r_f+alpha*(r-r_f)))^(phi);
end

function mutil=marginal_utility(alpha,phi,prob_low,
    prob_high,w_init,r_f,r_high,r_low)
r=prob_high*r_high+prob_low*r_low;
mutil=(w_init*(r-r_f))/((w_init*(1+r_f+alpha*(r-r_f)))
    ^(1-phi));
end
```

The fminbnd solution is at alpha=0.99993. The first fmincon solution, starting from the upper boundary is at alpha=0.99828. The second fmincon solution, starting from the lower boundary is at alpha=0.99767.

The two methods can yield different results because

1. The way the constraints are defined differ. In fmincon, the can hold with equality, in fminbnd they have to hold with strict inequality.

2. The way the algorithm searches for a minimum. fmincon starts at some point x_0 and moves along the x-axis while evaluating the function. fminbnd searches for a local minimum on the entire interval.

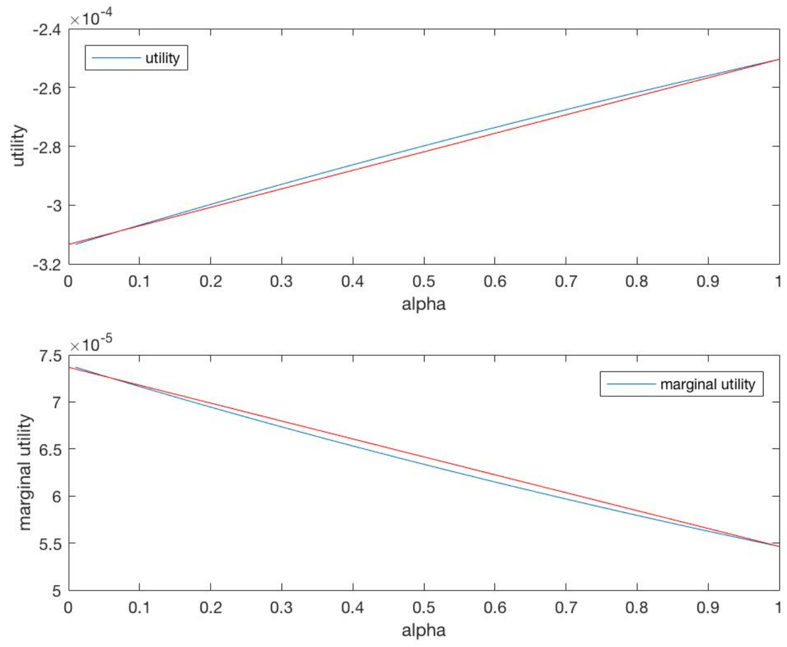


Figure 2: Illustration of the utility and marginal utility, depending on α.
(The red line is a reference line with linear slope)

Comment: After running the algorithm for the first two times, the solutions produced by fmincon were = 1. The difference between the results from the two algorithms was then easily explainable by the difference in the interval constrains mentioned above ($lb \leq \alpha \leq ub$ (fmincon) vs $lb < \alpha < ub$ (fminbnd)).

c)

Part two of this question resulted in $\alpha = 2.73308$. However, the solution in part three was bounded on the upside at $\overline{\alpha} = 1$. The solution, derived numerically, resulted in α close to one (and one the first tries with fmincon resulted in $\alpha = 1$, exactly the boundary). If one runs the maximization without the upper constrain, the argmax increases with the starting point (for fmincon (that the solution does not converge to infinity can be explained as the differences in function value lie below machine precision which results in a maximum being reported)). This implies that the household would like to borrow at the risk free rate (negative α^f) to invest more in stocks to increase his expected portfolio return.
Since $\phi = 1 - \gamma$ and $\phi = -3$, it can be implied that $\gamma = 4$. Thus, the agent is risk averse. However, one would expect a risk averse agent not to borrow to invest into the risky asset.

Looking at the utility function with the parameters specified in the question, $u(w_1) = \frac{\left(\frac{1}{w_1^3}\right)}{-3}$, it becomes obvious that with increasing wealth, the utility moves closer to zero from below as wealth increases. Since the probability to receive a high interest rate is nine times as large as the probability to receive a low interest rate, the expected return from the risky asset is much larger than the return of the safe asset.
Since the maximization problem can be rewritten as

$$\max_{\alpha} \underbrace{\frac{1}{\phi}}_{<0} \left(\underbrace{w_0 \left(1 + r^f + \alpha \left(E[r] - r^f\right)\right)}_{>0} \right)^{\underbrace{\phi}_{<0}} \tag{23}$$

which explains that α exceeds 1 since there is no more uncertainty in the model.