```
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
% This program solves an extended version of the model allowing for partial diversification
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


global a c e     % for the R function
global ufunction alfacara alfadara f...
          kink sloperatio     % for the u function
global utilitylevel     % for ulevel function
global beta     % divesification (0=full, 1=null) for country-1 and country-2 procedures
global n11 n21 r1     % for addressing directly country-2 procedure in welfare computation
global flagz     % will recieve error warnings from country-1 procedure that solves country 2
global options  start2     % solver options for country-1 procedure (for solving country-2)
global integralpoints integralstep     % precision of integral computations for country-1 and country-2 procedures


% ***********************************
% ************ INPUT  *************
% ***********************************

% return function: R = max{a*teta-c*n+e,0}  - recall that n is the number of withdrawers
% ^^^^^^^^^^^^^^^^^^^^^
a = 2; c = 5; e = 4;

% utility function:
% ^^^^^^^^^^^^^^^^^^^^^^
% choose one of the three utility functions by making the other two 'ufunction='x' comments:

% ufunction = 'C';
% CARA: -exp(-alfa*c);
%  alfa = absolute risk aversion coefficient.
alfacara=3;

ufunction = 'D';
%  DARA: (f+c)^(1-alfa)/(1-alfa)     (this is in fact a CRRA utility function)
%  alfa = relative risk aversion coefficient.
%  f = outside riskless income of the consumer (must set above 0 to avoid -infinity)
alfadara=7; f = 0.01;

% ufunction = 'K';
%  KINKED:   kink = wealth at kink, sloperatio = slope below kink / slope above kink
kink = 3; sloperatio = 10;


% precision of the computaions: number of points in integral approximations:
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
integralstep=0.005;
integralpoints=201;     % write manually 1/integralstep + 1   (to avoid rounding errors)

% setting  values of beta to solve for:
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
% beta is the degree of diversification,
% where beta = 1 stands for no diversification, and beta = 0 stands for full diversification

firstpoint = 0;
lastpoint = 1;
% step = 0.02;
step=0.5;







% ************************************************************
% ***************** THE MAIN PROGRAM *******************
% ************************************************************

% opening a file for output, and printing the input details there
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
% prepare a name for the output files (data and figures)
% in the text strings, the "%g"'s are places for the variables that appear after the text
if ufunction == 'C'
name = sprintf ( 'contagion-cara_alfa=%g-beta_%g_to_%g-precision_%g' ,...
   alfacara, firstpoint, lastpoint, integralpoints );
end
if ufunction == 'D'
name = sprintf ( 'contagion-dara_alfa=%g-f=%g-beta_%g_to_%g-precision_%g' ,...
   alfadara, f, firstpoint, lastpoint, integralpoints );
end
if ufunction == 'K'
name = sprintf ( 'contagion-kinked_kink=%g-sloperatio=%g-beta_%g_to_%g-precision_%g' ,...
   kink, sloperatio, firstpoint, lastpoint, integralpoints );
end
filename = [name,'.txt'];    % mane of the data file
figurename = [name,'.jpg'];    % name of the figure file

fid = fopen(filename, 'w');    % open data file for writing
fprintf ( fid, 'contagion model simulation,  ' );
fprintf ( fid, datestr (now) );    % the date
fprintf ( fid, ' \nreturn function: R = max { 0 , %g*teta-%g*n+%g }  \n' ,a ,c ,e );
if ufunction == 'C'
   fprintf ( fid, 'CARA utility function: u=-exp(-%g*c) \n\n', alfacara );
end
if ufunction == 'D'
   fprintf ( fid, 'DRRA utility function: u=(%g+c)^(1-%g)/(1-%g) \n\n', f, alfadara, alfadara );
end
if ufunction =='K'
   fprintf ( fid, 'KINKED utility function. kink at %g, sloperatio=%g \n\n', kink, sloperatio );
end

% setting  solver properties and starting points
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
start1 = [0 0.85];     % starting value for country-1 solver
start2 = [0 0 0 0.85];     % starting value for country-2 solver
options = optimset ('Display' , 'off' ,...    % turns off solver display
                              'TolFun',0.00000001);

% preparing the vector of values of beta to solve for; points is the number of elements
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
betaz = (firstpoint: step: lastpoint);
points = round( (lastpoint-firstpoint)/step+1);    %  to avoid rounding errors which lose the last element
% to avoid computational diffeiculties, avoid the extreem case of beta=0,1:
if firstpoint <= 0
  betaz(1) = 0.01;
end
if lastpoint >= 1
  betaz(points) = 0.99;
end

% initializing vectors
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
w1 = zeros(1,integralpoints);    % country-1 agents' welfare at any teta2 for a specific teta1
w2 = zeros(1,integralpoints);    % country-2 agents' welfare at any teta2 for a specific teta1
v1 = zeros(1,integralpoints);    % country-1 agents' average welfare (over teta2) for any teta1
v2 = zeros(1,integralpoints);    % country-2 agents' average welfare (over teta2) for any teta1
c1 = zeros(integralpoints,integralpoints);    % country-1 return any teta1, teta2
c2 = zeros(integralpoints,integralpoints);    % country-2 return any teta1, teta2
teta2starlist = zeros(1,integralpoints);    % stores teta2star for each teta1
flagzz = zeros(1,integralpoints);    % retains error information from country-2 solver for each teta1 at the welfare computation
flagz = zeros(2,integralpoints);    % retains error information from country-2 solver for each teta1 at the last teta1star
                                              %  first row for teta2star at each n11 (country 1 agents)
                                              % second row for teta2star at each n21 (country 2 agents)
teta1starz = zeros(1,points);    % collects teta1star at each beta for final graphs
avteta2z = zeros(1,points);    % collects the average teta2star at each beta for final graphs
moneytotalwelfarez = zeros(1,points);    % collects average welfare (translated to level of safe consumption) at each beta for graphs
correlationz = zeros(1,points);    % collects correlation of country-1 and country-2 returns at each beta for final graphs
```

```
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
% main loop (for each level of beta)
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

for i = 1:points

    beta = betaz(i)      % take the cuurent level of diversification from the list betaz

    % solving for teta1star
    % ^^^^^^^^^^^^^^^^^^^^^^^^^^^^

   [solution, funcvalue, flag1] = fsolve(@country1,start1, options);
   teta1star = solution(2);
   % explanation:
   % fslove tries to find a zero of the function/procedure "country1"
   % country1 accepts an input vector with 2 elements: [d teta1star] and outputs a vector with 2 elements
   % (the utility differencials for the 2 types of agents at the points at which they switch actions)
   % when fsolve terminates, it returns a list of results:
   % solution: a 2-elements vector [d teta1star] at which the output of country-1 is [0 0]
   % (we are interested only in solution(2) which is teta1star)
   % funcvalue: value of country1 at solution (not used)
   % flag1: >0 = ok; 0 = limit of iterations ; <0 = didn't find a solution

   teta1starindex = floor(teta1star/integralstep);
   % translates teta1star (between 0 and 1) to teta1starindex (integer between 0 and integralpoints)
   % we do this because we will run the loop over teta1 (0 to 1) by running an integer from 0 to integralpoints
   % and teta1starindex will be the point below which there is a run in country 1.

   integercorrection1 = teta1star/integralstep-teta1starindex;
   % since we approximate integral calculations by sum over integralpoints points,
   % there is an inaccuracy, especially at the discontinuity point teta1star
   % integercorrection helps reduce this inaccuracy by registering the rounding error

   % computation of welfare and correlation
   % ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

   % The external integral over teta1

     % run 1 area - note that because teta2star is constant here we run country 2 only once

       n11 = 1; n21 = 1;    % all agents from country 1 (n11) and country 2 (n21) run
       r1 = 1;    % the return in country 1 is 1

       % solve for teta2star
       [solution, funcvalue, reportcode] = fsolve(@country2, start2, options);
       % explanation: see solution for teta1star above, except for one difference: country-2 procedure accepts
       % a 4-elements vector [d12 d13 d14 teta2star] and outputs 4 utility differencials at the 4 switching points.
       teta2star = solution(4);
       teta2starindex = floor(teta2star/integralstep);
       integercorrection2 = teta2star/integralstep-teta2starindex;

       % The internal integral over teta2 (at the given teta1)

       % compute for all teta2 as if there are no runs (saves run time by using vector operations)
       teta2z = 0:integralstep:1;
       r2z = max(0,R(teta2z,0));
       w1 = u( (1+beta)*1+(1-beta)*r2z );
       w2 = u( (1-beta)*1+(1+beta)*r2z );
       % remember values at discontinuity point (jump of returns between the run and stay areas) for correction later
       w1mid = w1(teta2starindex);
       w2mid = w2(teta2starindex);
       % run 2 area (overwrite)
       w1(1:teta2starindex) = u( (1+beta)*1+(1-beta)*1 );
       w2(1:teta2starindex) = u( (1-beta)*1+(1+beta)*1 );
       % correct for the approximation in the integral at the discontinuity
       w1(teta2starindex) = integercorrection2*w1(teta2starindex)+(1-integercorrection2)*w1mid;
       w2(teta2starindex) = integercorrection2*w2(teta2starindex)+(1-integercorrection2)*w2mid;
       % integrate
       m1 = mean(w1);
       m2 = mean(w2);
```

```
    % store results by duplicating (for each teta1 below teta1star)
    teta2starlist(1:teta1starindex) = teta2star;
    flagzz(1:teta1starindex) = reportcode;
    v1(1:teta1starindex) = m1;
    v2(1:teta1starindex) = m2;
    c1(1:teta1starindex,:) = 1;    % country-1 return for each teta1 below teta1star and each teta2 (agents run)
    for teta1index=1:teta1starindex     %  country-2 return for each teta1 below teta1star
      c2(teta1index,:) = max(0,R(teta2z,0));    %  at each teta2 - as if agents never run (save time by using vector operations)
      c2(teta1index,1:teta2starindex) = 1;    %  overwrite below teta2star - here agents run
    end

  % stay 1 area - note that now we need to compute teta2star for each teta1

  for teta1index = teta1starindex+1:integralpoints;
    teta1 = (teta1index-1)*integralstep;
    n11 = 0; n21 = 0;    % no agents from country 1 (n11) and country 2 (n21) run
    r1 = max(0,R(teta1,0));    % country-1 return when there is no run there

    [solution, funcvalue, reportcode] = fsolve(@country2, start2, options);    % solve for teta2star
    teta2star = solution(4);
    teta2starindex = floor(teta2star/integralstep);
    integercorrection2 = teta2star/integralstep-teta2starindex;

    % The internal integral over teta2 (at the given teta1)

    % compute for all teta2 as if there are no runs (saves run time by using vector operations)
    teta2z = 0:integralstep:1;
    r2z = max(0,R(teta2z,0));
    w1 = u( (1+beta)*r1+(1-beta)*r2z);
    w2 = u( (1-beta)*r1+(1+beta)*r2z);
    % remember values at discontinuity point for correction later
    w1mid = w1(teta2starindex+1);
    w2mid = w2(teta2starindex+1);
    % run 2 area (overwrite)
    w1(1:teta2starindex) = u( (1+beta)*r1+(1-beta)*1 );
    w2(1:teta2starindex) = u( (1-beta)*r1+(1+beta)*1 );
    % correct for the approximation in the integral at the discontinuity
    w1(teta2starindex) = integercorrection2*w1(teta2starindex)+(1-integercorrection2)*w1mid;
    w2(teta2starindex) = integercorrection2*w2(teta2starindex)+(1-integercorrection2)*w2mid;
    % integrate and store results
    teta2starlist(teta1index) = teta2star;
    flagzz(teta1index) = reportcode;
    v1(teta1index) = mean(w1);
    v2(teta1index) = mean(w2);
    c1(teta1index,:) = max(0,R(teta1,0));    % country-1 return (agents stay)
    c2(teta1index,:) = max(0,R(teta2z,0));    %  country-2 return: (as if agents never run)
    c2(teta1index,1:teta2starindex) = 1;    %  country-2 return: below teta2star agents run - overwrite)

  end

% correct for the approximation in the integral over teta1 at the discontinuity
v1(teta1starindex) = integercorrection1*m1+(1-integercorrection1)*v1(teta1starindex);
v2(teta1starindex) = integercorrection1*m2+(1-integercorrection1)*v2(teta1starindex);
% integrate
welfare1 = mean(v1);
welfare2 = mean(v2);

% compute correlation
cc1 = reshape(c1,[integralpoints^2,1]);    % transform matrices to vectors
cc2 = reshape(c2,[integralpoints^2,1]);    % transform matrices to vectors
corrmatrix = corrcoef ( cc1 , cc2 );    % compute the 2by2 correlation matrix
correlation = corrmatrix(1,2);    % the correlation coeeficient is an off diagonal element

% collect data
totalwelfare = (welfare1+welfare2)/2;
avteta2 = mean(teta2starlist);
% translate welfare levels to the safe consumption level that yields that welfare
utilitylevel = welfare1;
moneywelfare1 = fsolve(@ulevel, 1, options);
```

```
    utilitylevel = welfare2;
    moneywelfare2 = fsolve(@ulevel, 1, options);
    utilitylevel = totalwelfare;
    moneytotalwelfare = fsolve(@ulevel, 1, options);


    % output results for this beta
    % ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

    fprintf ( fid, 'beta = %g \n',beta);
    fprintf ( fid, 'teta1star = %g     ',teta1star);
    fprintf ( fid, 'average teta2star = %g \n',avteta2);
    fprintf ( fid, 'correlation of returns = %g \n',correlation);
    fprintf ( fid, 'country-1 agents'' welfare = %g , equivalent to consumption of %g \n',welfare1, moneywelfare1);
    fprintf ( fid, 'country-2 agents'' welfare = %g , equivalent to consumption of %g \n',welfare2, moneywelfare2);
    fprintf ( fid, 'average welfare = %g , equivalent to consumption of %g \n',totalwelfare, moneytotalwelfare);
    % report if the solver run into problems at soem stage, in which case results might be wrong
    if flag1 <= 0
       fprintf ( fid, 'problems in computation of teta1star. the report code is: \n');
       fprintf ( fid, '%g ',flag1);
       fprintf ( fid, '\n');
    end
    if any (any (flagz <= 0) )
       fprintf ( fid, 'problems in computation of some of teta2star for teta1star. the vector of flags is: \n');
       fprintf ( fid, '%g ',flagz');
       fprintf ( fid, '\n');
    end
    if any (flagzz <= 0)
       fprintf ( fid, 'problems in computation of some of teta2star for welfare. the vector of flags is: \n');
       fprintf ( fid, '%g ',flagzz');
       fprintf ( fid, '\n');
    end
    fprintf ( fid, '\n**********************************************************************\n\n');

    % collect data of this beta for graphs
    % ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    teta1starz(i) = teta1star;
    avteta2z(i) = avteta2;
    moneytotalwelfarez(i) = moneytotalwelfare;
    correlationz(i) = correlation;

 end

% close the output file
% ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
status = fclose ( fid );


% plot graphs
% ^^^^^^^^^^^^^^^^

subplot (3,1,1);    % top plot
plot ( betaz, teta1starz, '-', betaz, avteta2z, ':' );
title ( 'teta1* (solid) and average teta2* (dotted)' );

subplot (3,1,2);    % middle plot
plot ( betaz, moneytotalwelfarez );
% axis ([0 1 2.25 2.4])    % use this command to manually set the ranges
title ( 'average welfare (money units equivalent)');

subplot (3,1,3);    % bottom plot
plot ( betaz, correlationz );
title ( 'correlation between returns' );

print ( '-djpeg' , figurename )          % output figure to file
```