Enrico Scalas

Software for dealing with finitary models and continuous-time random walks (version 2.0)

This document contains pointers to software produced in the framework of the project *The growth of firms and countries: distributional properties and economic determinants* (grant number 2009H8WPX5). Namely, it includes software for the sub-project *Finitary and non-finitary probabilistic models in economics* (grant number 2009H8WPX5_002).

Software description

Program name: agent_market_model.nlogo

Program author(s): Marco Bosco, Pietro Terna

Program language: NetLogo

Program site: http://web.econ.unito.it/terna/tesi/agent_market_model.html

Program listing: See program site

Program description: The model describes a contemporary economic system composed by agents that are characterized by a precise economic condition and that interact on three types of market: the job market, the goods market and the financial market. The agents are divided into four different categories: Households/Families, who hold capital, provide work in exchange for wage, save and consume; Firms, that provide a wage for Families, produce goods and invest; Banks, which collecting deposits and making loans, participate in the circuit of money creation and a Central Bank which has the status of last-resort lender.

Related paper(s):

RABERTO M, RAPALLO F, SCALAS E (2011). Semi-Markov graph dynamics. PLOS ONE, vol. 6, ISSN: 1932-6203, doi: 10.1371/journal.pone.0023370

Paper site(s):
http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0023370

Program name: doubleauction.m

Program author(s): Tijana Radivojević

Program language: matlab

Program site: http://www.bcamath.org/en/people/radivojevic/simulations

Program listing: See paper sites

Program description: We introduced a stylized phenomenological model for the continuous double auction . The model reproduces the behaviour of zero-intelligence agents. For what concerns the number of orders, the model is equivalent to two independent M/M/1 queues. Our simulations present the essential double auction trading mechanism in the case of statistical equilibrium, i.e. when the load of the system is less than 1 and in the out-of-equilibrium case, i.e. the load is greater than 1.

Related paper(s):

Radivojević T., Anselmi J., Scalas E. A stylized model for the continuous double auction Managing Market Complexity, Lecture Notes in Economics and Mathematical Systems vol. 662 (2012) 115-125

Radivojević T., Anselmi J., Scalas E. Ergodic transition in a simple model of the continuous double auction

Paper site(s):
http://www.bcamath.org/documentos_public/archivos/publicaciones/RAS.pdf
http://www.bcamath.org/documentos_public/archivos/publicaciones/1_RAS-17_04_13.pdf

Program name: econ_model.m

Program author(s): Tijana Radivojević

Program language: matlab

Program site: http://www.bcamath.org/en/people/radivojevic/simulations

Program listing: see section listing below

Program description: Using the interplay of three simple exchange games, a representation for a conservative economic system is given. The first game mimics the background noise of the economy with pair-wise wealth exchange. The second game mimics taxation and redistribution as a centralized mechanism, in a simplified way, whereas, in the third game, occasional failure of an agent leads to redistribution of its wealth. These games can be used to investigate the emergence of statistical equilibrium in a simple pure-exchange environment.

Related paper(s):

Garibaldi U., Radivojević T., Scalas E. Interplay of simple stochastic games as models for the economy Proceedings of Applications of Mathematics 2013, Institute of Mathematics, Academy of Sciences of the Czech Republic, Prague, 77-87

Scalas E., Radivojević T., Garibaldi U. Wealth distribution and the Lorenz curve: A finitary approach

Paper site(s)
http://www.bcamath.org/documentos_public/archivos/publicaciones/GRS.pdf
http://www.bcamath.org/documentos_public/archivos/publicaciones/1_GRS290313.pdf

Program name: sum_rtf.m

Program author(s): Danilo Aringhieri, Michele Manzini, Andrea Mezzalira, Enrico Scalas.

Program language: matlab

Program site: in preparation

Program listing: in preparation

Related paper(s): in preparation

Program description: The program aims to build and analyse a signal obtained as a linear superposition of random telegraph signals (RTS) of Mittag-Leffler type. In the first procedure the Mittag-Leffeler-distributed sojourn times are generated and the sampling time is obtained as a fraction of the smallest sojourn time.

After the signal is created, the program computes the autocorrelation function and the power spectrum, comparing both of them with the theoretical predictions.

To reduce the noise and to increase the precision of the simulation, the code repeats the described procedure for a given number of times and mediates the results.

Listings

Function econ_model.m

```
% B D Y GAME (Bennati-Dragulescu-Yakovenko)

function Y = bdy(g,Ybdy)

    % Random selection of the loser

    indexl = randi(g);

    % Verify if the loser has objects

    while Ybdy(indexl) == 0

        indexl = randi(g);
```

```
    end
    % Random selection of the winner
    indexw = randi(g);
    % Dynamic step
    Ybdy(indexl) = Ybdy(indexl) - 1;
    Ybdy(indexw) = Ybdy(indexw) + 1;
    Y = Ybdy;
end

% B D Y GAME (Bennati-Dragulescu-Yakovenko) - one step
function [Y,freq,normal] = bdy_yf(n,g,Ybdy,freqold)
    % Random selection of the loser
    indexl = randi(g);
    % Verify if the loser has objects
    while Ybdy(indexl) == 0
        indexl = randi(g);
    end
    % Random selection of the winner
    indexw = randi(g);
    % Dynamic step
    Ybdy(indexl) = Ybdy(indexl) - 1;
    Ybdy(indexw) = Ybdy(indexw) + 1;
    Y = Ybdy;
    % frequencies (from 0)
    normal = sum(freqold);
    f = zeros(1,n+1);
    for j=0:n
```

```
        f(j+1) = length(find(Y==j));
        freqold(j+1) = freqold(j+1) + f(j+1);
    end
    freq = freqold;
    normal = normal + g;
end

% Z I P F - Y U L E - S I M O N GAME
function Y = zsy(n,Yzsy)
    % destruction
        indexp = find(Yzsy>0); % agents with at least one coin
        Kp = length(indexp); % number of such agents
        % an agent is selected and removed
        R = randi(Kp);
        irem = indexp(R);
    m = Yzsy(irem); % wealth of the removed agent
    Yzsy(irem) = 0; % agent is removed
    N = n-m; % number of coins in the system
    % creation
    for i=1:m-1
        % when all the sites are empty, a new site is filled with uniform probability.
        % It never coincides with the previously destroyed site
        indexp = find(Yzsy>0); % number of active clusters
        prob = Yzsy(indexp)/N; % frequency vector
        cumprob = cumsum(prob); % cumulative probability
        indexsite = find((cumprob-rand(1))>0, 1 );
        indexsite = indexp(indexsite);
```

```
% pointer to selected site randomly out of active sites
        Yzsy(indexsite) = Yzsy(indexsite) + 1;


    N = N + 1;
    end
    Yzsy(irem) = 1; % compassionate capitalism
    Y = Yzsy;
end


% T A X A T I O N - R E D I S T R I B U T I O N GAME - one step
function [Y,freq,normal] = tr_yf(n,g,Ytr,freqold,alpha)
sumalpha = sum(alpha); % Polya parameter
% taxation - a coin is randomly taken out of n coins from an agent with
% at least one coin
m = floor(n/2); % number of destructions
N = n; % number of coins in the system
for i=1:m
    probt = Ytr/N; % prob. of selecting a coin from agent, proportional to the wealth
    cumprobt = cumsum(probt); % cumulative probability
    indext = find((cumprobt-rand(1))>0, 1 ); % pointer to select agent randomly
    Ytr(indext) = Ytr(indext) - 1;
    N = N - 1;
end
% redistribution - m coins are redsitributed among g agents
for i=1:m
    probr = (alpha+Ytr)/(sumalpha+N);
    cumprobr = cumsum(probr);
```

```
    indexr = find((cumprobr-rand(1))>0, 1 ); % pointer to select agent randomly
    Ytr(indexr) = Ytr(indexr) + 1;
    N = N + 1;
end
Y = Ytr;
    % frequencies (from 0)
    normal = sum(freqold);
    f = zeros(1,n+1);
    for j=0:n
        f(j+1) = length(find(Y==j));
        freqold(j+1) = freqold(j+1) + f(j+1);
    end
    freq = freqold;
    normal = normal + g;
end
```

Function econ_model.m

```
randn('state', sum(100*clock));
rand('twister', sum(100*clock));
tic;
n = 10000; % number of coins
g = 100; % number of agents
c = n/g; % initial number of coins per agent
nruns = 10^2; % number of Monte Carlo steps
Tmax = 5000; % number of years
% parameters
alpha = 10^0*ones(1,g); % redistribution parameters (leading to symetric Polya if all equal) (TR)
```

```
sumalpha = sum(alpha); % Polya parameter

k = 5; % number of ZSY steps (in a year)

l = 50; % number of BDY steps, following one ZSY step

f = 0:n;

% initalization

Y0 = c*ones(1,g); % occupation vector

freq = zeros(1,n+1); freq(c+1) = g;

% variables for collecting samples (detection of the wealth) before and after taxation

mFn_b = zeros(nruns,n+1);

mFn_a = zeros(nruns,n+1);

for i=1:nruns

    Y = Y0;

    freq = zeros(1,n+1); freq(c+1) = g;


     freq_b=freq;

     freq_a=freq;

    for T=1:Tmax


            for j=1:k

                Y = zsy(n,Y);

                for d=1:l

                      if d==l & j==k

                       [Y,freq_b,normal] = bdy_yf(n,g,Y,freq_b);

                      else

                         Y = bdy(g,Y);

                      end
```

```
            end
          end;
        [Y,freq_a,normal] = tr_yf(n,g,Y,freq_a,alpha);;

   end

   mFn_b(i,:) = freq_b/normal;
   mFn_a(i,:) = freq_a/normal;

end

 mF_b = mean(mFn_b);
 mF_a = mean(mFn_a);

figure(1)
subplot(2,1,1)
plot(f,mF_a,'*k','Markersize',5)
hold on
title('wealth distribution','FontSize',16)
xlabel('number of coins','FontSize',16);
set(gca,'XTickLabel',[0:500],'FontSize',16);
ylabel('time mean of rel. frequences','FontSize',16);

subplot(2,1,2)
loglog(f,mF_a,'*k','Markersize',5)
hold on
xlabel('number of coins','FontSize',16);
```

```
set(gca,'XTickLabel',[0:500],'FontSize',16);
ylabel('time mean of rel. frequences','FontSize',16);
```