# ДОДАТОК А
## Тексти програм (функцій)

```
function varargout = Hon(varargin)
% HON M-file for Hon.fig
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Hon_OpeningFcn, ...
                   'gui_OutputFcn',  @Hon_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Hon is made visible.
function Hon_OpeningFcn(hObject, eventdata, h, varargin)
% This function has no output args, see OutputFcn.
% hObject   handle to figure
% eventdata reserved - to be defined in a future version of
MATLAB
% h         structure with handles and user data (see GUIDATA)
% varargin  command line arguments to Hon (see VARARGIN)
% Choose default command line output for Hon
%h.output = hObject;
h.NumSer = 0;
h.N = 512;
h.LW = 2;
DisableButtons(h);
set(h.axes1, 'XLim',[0 h.N], 'NextPlot','replacechildren',...
    'XTickLabel',[], 'XGrid','on','YGrid','on');
set(h.axes2, 'XLim',[0 h.N], 'NextPlot','add',
'XGrid','on','YGrid','on');
h.LSO =  ['- ';'--';': ';'-.'];
h.curstyle = [0 0 0 0];
h.NumSer = 0;
guidata(hObject, h);                % Update handles structure

% --- Outputs from this function are returned to the command
line.
function varargout = Hon_OutputFcn(hObject, eventdata, h)
%varargout{1} = h.output;
```

```
% --- Executes on button press in btn_auto.
function btn_auto_Callback(hObject, eventdata, h)
set(h.axes1,'Visible','off');
set(h.axes2,'Visible','off');
set(h.axes3,'Visible','on', 'Position',[40 25 600 480]);
set(h.btn_uBm,'Enable','off');
set(h.btn_fBm,'Enable','off');
set(h.btn_sLp,'Enable','off');
set(h.btn_fLm,'Enable','off');
for k = 1:h.NumSer
    [T, tmp] = IncAutocorr(h.Motion(:,k));
    plot(h.axes3, T, tmp, 'LineWidth',2,...
        'Color',h.Color(k,:), 'LineStyle',h.LStyle(k,:),
'Marker','o');
    hold on;
    h.Corr(:,k) = tmp;
end
set(h.axes3, 'XLim', [0 T(end)+1]);
grid on; hold off;
guidata(gcf, h);

% --- Executes on button press in btn_frac.
function btn_frac_Callback(hObject, eventdata, h)
set(h.axes1,'Visible','off');
set(h.axes2,'Visible','off');
set(h.axes3,'Visible','on', 'Position',[40 25 600 480]);
set(h.btn_uBm,'Enable','off');
set(h.btn_fBm,'Enable','off');
set(h.btn_sLp,'Enable','off');
set(h.btn_fLm,'Enable','off');
[T, h.Len] = FracLen(h.Motion);
for k = 1:h.NumSer
    loglog(h.axes3, T, h.Len(:,k), 'LineWidth',2,
'Marker','o',...
        'Color',h.Color(k,:), 'LineStyle',h.LStyle(k,:));
    hold on;
end
%loglog(h.axes3, T, 2*h.N/gamma(1/h.Alpha(1))*T.^(h.Hurst(1)-
1),...
%    '--*k', 'LineWidth',2);
y1 = floor(log2(min(min(h.Len))));
y2 = ceil(log2(max(max(h.Len))));
y = pow2(y1 : y2);
set(h.axes3, 'XLim',[T(1) T(end)], 'YLim',pow2([y1 y2]),
'XTick',T, 'YTick',y);
grid on, hold off;
set(h.axes3, 'XGrid','on', 'XMinorGrid','off',
'YMinorGrid','off', 'YGrid','on');
guidata(gcf, h);

% --- Executes on button press in btn_exit.
function btn_exit_Callback(hObject, eventdata, h)
close;
```

```
% --- Executes on button press in btn_save.
function btn_save_Callback(hObject, eventdata, h)
h.f_out = uiputfile('*.xls','Save time series');
if isequal(h.f_out,0) return, end;
if isfield(h,'Noise')
    xlswrite(h.f_out, {'N'; 'Method'; 'Hurst'; 'Alpha'},
'Noise', 'A1');
    xlswrite(h.f_out, h.N, 'Noise', 'B1');
    xlswrite(h.f_out, [h.Method; h.Hurst; h.Alpha], 'Noise',
'B2');
    xlswrite(h.f_out, [0:h.N]','Noise', 'A5');
    xlswrite(h.f_out, h.Noise, 'Noise', 'B5');
end
%
if isfield(h, 'Motion')
    xlswrite(h.f_out, {'N'; 'Method'; 'Hurst'; 'Alpha'},
'Motion', 'A1');
    xlswrite(h.f_out, h.N, 'Motion', 'B1');
    xlswrite(h.f_out, [h.Method; h.Hurst; h.Alpha], 'Motion',
'B2');
    xlswrite(h.f_out, [0:h.N]', 'Motion', 'A5');
    xlswrite(h.f_out, h.Motion, 'Motion', 'B5');
end
%
if isfield(h, 'Corr') && ~isempty(h.Corr)
    xlswrite(h.f_out, {'N'; 'Method'; 'Hurst'; 'Alpha'},
'Corr', 'A1');
    xlswrite(h.f_out, h.N, 'Corr', 'B1');
    xlswrite(h.f_out, [h.Method; h.Hurst; h.Alpha], 'Corr',
'B2');
    xlswrite(h.f_out, [0:h.N]', 'Corr', 'A5');
    xlswrite(h.f_out, h.Corr, 'Corr', 'B5');
end
%
if isfield(h, 's') && ~isempty(h.s)
    xlswrite(h.f_out, {'N'; 'Method'; 'Hurst'; 'Alpha'},
'FracLen', 'A1');
    xlswrite(h.f_out, h.N, 'FracLen', 'B1');
    xlswrite(h.f_out, [h.Method; h.Hurst; h.Alpha], 'FracLen',
'B2');
    xlswrite(h.f_out, pow2(0:log2(h.N)-1)', 'FracLen', 'A5');
    xlswrite(h.f_out, h.Len, 'FracLen', 'B5');
end

% --- Executes on button press in btn_clear.
function btn_clear_Callback(hObject, eventdata, h)
set(h.slider1, 'Enable','on');  set(h.ed_N,'String','512');
set(h.btn_uBm,'Enable','on');   set(h.btn_fBm,'Enable','on');
set(h.btn_sLp,'Enable','on');   set(h.btn_fLm,'Enable','on');
set(h.ed_Alpha,'String','2.0');
set(h.ed_Hurst,'String','0.5');
set(h.txt_alpha, 'String', '');
```

```
set(h.txt_hurst, 'String', '');
set(h.txt_gencc, 'String', '');
DisableButtons(h);
h.NumSer = 0; h.N = 512; h.curstyle = [0 0 0 0]; h.LW = 2;
h.Hurst = []; h.Alpha = []; h.Corr = []; h.Method = [];
h.Color = []; h.LStyle = ''; h.Noise = []; h.Motion = []; h.s
= [];
set(h.axes1,'NextPlot','replace');
newplot(h.axes1);
set(h.axes1,'Visible','on','XLim',[0
h.N],'NextPlot','replacechildren',...
    'XTickLabel',[],'XGrid','on','YGrid','on');
set(h.axes2,'NextPlot','replace');
newplot(h.axes2);
set(h.axes2,'Visible','on','XLim',[0 h.N],'NextPlot','add',...
    'XGrid','on','YGrid','on');
set(h.axes3,'NextPlot','replace');
newplot(h.axes3);
set(h.axes3,'Visible','off','NextPlot','replace');
guidata(gcf, h);

function ed_Alpha_Callback(hObject, eventdata, h)
cur = str2double(get(hObject,'String'));
if isnan(cur) | cur <= 0 | cur > 2
    errordlg('Enter  0 < Alpha <= 2','Input Error','modal')
end

function ed_Hurst_Callback(hObject, eventdata, h)
cur = str2double(get(hObject,'String'));
if isnan(cur) | cur <= 0 | cur >= 1
    errordlg('Enter  0 < Hurst < 1','Input Error','modal')
end

function [newstyle, number] = StyleUpdate(LSO, curstyle)
number = curstyle + 1;
if number > length(LSO), number = 1; end
newstyle = LSO(number,:);

% --- Executes on button press in btn_uBm.
function btn_uBm_Callback(hObject, eventdata, h)
EnableButtons(h);
set(h.ed_Alpha,'String','2.0');
set(h.ed_Hurst,'String','0.5');
k = h.NumSer + 1;
h.NumSer = k;
h.Method(k) = 1;
h.Hurst(k) = 0.5;
h.Alpha(k) = 2;
h.Color(k,:) = [0 0 1];
h.Noise(:,k) = [0; sqrt(2)*randn(h.N, 1)];
h.Motion(:,k) = cumsum(h.Noise(:,k));
guidata(gcf, h);
PlotSeries(h, k, 1);
```

```
% --- Executes on button press in btn_fBm.
function btn_fBm_Callback(hObject, eventdata, h)
set(h.ed_Alpha,'String','2.0');
EnableButtons(h);
k = h.NumSer + 1;
h.NumSer = k;
h.Method(k) = 2;
h.Hurst(k) = str2double(get(h.ed_Hurst,'String'));
h.Alpha(k) = 2;
h.Color(k,:) = [1 0 1];
[h.Noise(:,k), h.Motion(:,k)] = fBm(h.N, h.Hurst(k));
guidata(gcf, h);
PlotSeries(h, k, 2);

% --- Executes on button press in btn_sLp.
function btn_sLp_Callback(hObject, eventdata, h)
EnableButtons(h);
k = h.NumSer + 1;
h.NumSer = k;
h.Method(k) = 3;
h.Alpha(k) = str2double(get(h.ed_Alpha,'String'));
if h.Alpha(k) < 1
    errordlg('1 <= Alpha <= 2 for Stable Levy Process','Input
Error','modal')
else
    h.Hurst(k) = 1 / h.Alpha(k);
    set(h.ed_Hurst,'String',h.Hurst(k));
    h.Color(k,:) = [0 0.5 0];
    h.Noise(:,k) = [0; alpha_stable_rand(h.N, h.Alpha(k))];
    h.Motion(:,k) = cumsum(h.Noise(:,k));
    guidata(gcf, h);
    PlotSeries(h, k, 3);
end

% --- Executes on button press in btn_fLm.
function btn_fLm_Callback(hObject, eventdata, h)
EnableButtons(h);
k = h.NumSer + 1;
h.NumSer = k;
h.Method(k) = 4;
h.Hurst(k) = str2double(get(h.ed_Hurst,'String'));
h.Alpha(k) = str2double(get(h.ed_Alpha,'String'));
h.Color(k,:) = [1 0 0];
[h.Noise(:,k), h.Motion(:,k)] = fLm(h.N, h.Hurst(k),
h.Alpha(k));
guidata(gcf, h);
PlotSeries(h, k, 4);

function PlotSeries(h, k, m)
[h.LStyle(k,:), h.curstyle(m)] = StyleUpdate(h.LSO,
h.curstyle(m));
```

```
plot(h.axes1, [0:h.N], h.Noise(:,k),
'Color',h.Color(k,:),'LineWidth',1);
plot(h.axes2, [0:h.N], h.Motion(:,k), 'Color',h.Color(k,:),...
    'LineStyle',h.LStyle(k,:), 'LineWidth',h.LW);
guidata(gcf, h);

function EnableButtons(h)
set(h.btn_alpha,'Enable','on');
set(h.btn_hurst,'Enable','on');
set(h.btn_gencc,'Enable','on');
set(h.btn_save,'Enable','on');
set(h.btn_frac,'Enable','on');
set(h.btn_auto,'Enable','on');
set(h.btn_clear,'Enable','on');
set(h.slider1, 'Enable','off');
guidata(gcf, h);

function DisableButtons(h)
set(h.btn_alpha,'Enable','off');
set(h.btn_hurst,'Enable','off');
set(h.btn_gencc,'Enable','off');
set(h.btn_save,'Enable','off');
set(h.btn_frac,'Enable','off');
set(h.btn_auto,'Enable','off');
set(h.btn_clear,'Enable','off');
set(h.slider1, 'Enable','on');
guidata(gcf, h);

% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, h)
h.N = pow2(floor(get(hObject,'Value')));
set(h.ed_N, 'String', h.N);
set(h.axes1, 'XLim',[0 h.N]);
set(h.axes2, 'XLim',[0 h.N]);
if h.N <= 512, h.LW = 2, else h.LW = 1, end
guidata(gcf, h);

% --- Executes on button press in btn_alpha.
function btn_alpha_Callback(hObject, eventdata, h)
% hObject    handle to btn_alpha (see GCBO)
% h    structure with handles and user data (see GUIDATA)
h.EA = EstAlpha(h.Noise,sqrt(1-4.^(h.Hurst-1)));
set(h.txt_alpha, 'String', mat2str(h.EA,3));
guidata(gcf, h);

% --- Executes on button press in btn_hurst.
function btn_hurst_Callback(hObject, eventdata, h)
% hObject    handle to btn_hurst (see GCBO)
% h    structure with handles and user data (see GUIDATA)
h.EH = EstHurst(h.Noise);
set(h.txt_hurst, 'String', mat2str(h.EH,3));
guidata(gcf, h);
```

```
% --- Executes on button press in btn_gencc.
function btn_gencc_Callback(hObject, eventdata, h)
% hObject    handle to btn_gencc (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% h    structure with handles and user data (see GUIDATA)
h.EC = EstCoef(h.EA, h.EH);
set(h.txt_gencc, 'String', mat2str(h.EC,3));
guidata(gcf, h);


function [T, Len] = FracLen(Ser)

% Computation of fractal length of time series
%
% INPUT ARGUMENTS---------------------------------
%
% Ser:     Sample series (vector or matrix).
%          If Ser is vector, its size should be 1 + power of
two
%          If Ser is matrix, it is treated as number of
vectors
%
% OUTPUT VARIABLES--------------------------------
%
% T:       vector of time steps (power of two from 0 to
log2(N-1)-1)
% Len:     vector or matrix of fractal length
%
[N M] = size(Ser);
if N==1  Ser = Ser'; N = M; end;
if N==1  return; end;
n = floor(log2(N-1));
step = (N-1) / 2;
for t = 1:n
    ind = 0 : step : N-1;
    x = diff(Ser(ind+1,:));
    Len(n-t+1,:) = sum(abs(x));
    step = step / 2;
end
T = pow2(0:n-1)';


function [noise, motion] = fBm(N, H)

% Generation of Brownian motion and its increments ( Gaussian
noise)
%
% INPUT ARGUMENTS---------------------------------
%
% N:     Sample size ( is a power of 2 )
% H:     Hurst parameter ( 0 < H < 1 )
%
```

```
% OUTPUT VARIABLES-------------------------------
%
% motion:   Brownian motion
% noise:    Gaussian noise
%
sigma = N^H;
y = zeros(N+1, 1);
y(N+1) = sigma * randn(1);
k = log2(N);
step = N / 2;
sigma = sigma * sqrt(4^(-H)-1/4);
rho = pow2(-H);
%rho = sqrt(4^(-H)-1/4);
for t = 1:k
%    n = 2^t + 1;
    x = step:2*step:N-step;
    y(x+1) = (y(x+1-step)+y(x+1+step))/2 + sigma *
randn(numel(x),1);
    sigma = sigma * rho;
    step = step/2;
end
motion = sqrt(2) * y;
noise = [0; diff(motion)];

function [noise, motion] = fLm(N, H, Alpha)

% Generation of fractal Levy motion and its increments (
Gaussian noise)
%
% INPUT ARGUMENTS---------------------------------
%
% N:     Sample size     ( is a power of 2 )
% H:     Hurst parameter ( 0 < H < 1 )
% Alpha: Stability index ( 0 < Alpha <= 2 )
%
% OUTPUT VARIABLES-------------------------------
%
% motion:   fractal Levy motion
% noise:    Alpha-stable noise
%
sigma = N^H;
y = zeros(N+1, 1);
y(N+1) = sigma * alpha_stable_rand(1, Alpha);
k = log2(N);
step = N / 2;
sigma = sigma * sqrt(4^(-H)-1/4);
rho = pow2(-H);
for t = 1:k
%    n = 2^t + 1;
    x = step:2*step:N-step;
    y(x+1) = (y(x+1-step) + y(x+1+step)) / 2 + ...
             sigma * alpha_stable_rand(numel(x), Alpha);
    sigma = sigma * rho;
```

```
    step = step/2;
end
motion = y;
noise = [0; diff(motion)];


function [T, Corr] = IncAutocorr(Ser)

% Computation of autocorrelation with rising time step
%
% INPUT ARGUMENTS---------------------------------
%
% Ser:     Vector of sample series with size 1 + power of two
%
% OUTPUT VARIABLES--------------------------------
%
% T:       Vector of time steps
% Corr:    Vector of correlation coefficients
%
[N M] = size(Ser);
if N==1  Ser = Ser'; N = M; end;
if N==1  return; end;
n = floor(log2(N-1));
N = pow2(n);
T = unique(fix(pow2(0:0.25:n-4)));
for s = 1:numel(T)
    step = T(s);
    ind = 1:N+1-2*step;
    dx(ind,1) = Ser(ind+step)   - Ser(ind);
    dx(ind,2) = Ser(ind+2*step) - Ser(ind+step);
    w = corrcoef(dx);
    Corr(s) = w(1,2);
    dx = [];
end

function EA = EstAlpha(Ser,c)
% Computation of stability index (Alpha)
%
% INPUT ARGUMENTS---------------------------------
%
% Ser:     Sample series (vector or matrix).
%          If Ser is matrix, it is treated as number of
vectors
%
% OUTPUT VARIABLES--------------------------------
%
% EA:      estimated value of stability index (Alpha)
%)
[N M] = size(Ser);
if N==1  Ser = Ser'; N = M; end;
if N==1  return; end;
a = 1.19236;
b = 0.64072;
```

```
s = 0.5;
chi = cos(pi/2*s)*gamma(1-s);
EA = zeros(1,M);
for k = 1:M
    v = abs(Ser(:,k));
%    m = c(k);
    m = (quantile(v,0.72)-quantile(v,0.28))/1.654;
    z = mean((v/m).^s);
    e = s*(a+b./(chi*z-1));
    EA(k) = max(min(e,2),1);
    EA(k) = stabcull(Ser(:,k));
end
%EA = z;
%z = mean(v.^s);

function EH = EstHurst(Ser)
% Computation of Hurst index (H)
%
% INPUT ARGUMENTS---------------------------------
%
% Ser:     Sample series (vector or matrix).
%          If Ser is matrix, it is treated as number of
vectors
%
% OUTPUT VARIABLES--------------------------------
%
% EH:      estimated value of Hurst index (H)
%)
[N M] = size(Ser);
if N==1  Ser = Ser'; N = M; end;
if N==1  return; end;
s = 0.5;
v = mean(abs(Ser).^s);
w = Ser(1:end-1,:) + Ser(2:end,:);
w = mean(abs(w).^s);
EH = (log2(w)-log2(v)) / s;

function EC = EstCoef(Alpha, Hurst)
n = numel(Alpha);
if n~=numel(Hurst)   EC = NaN; return; end;
EC = zeros(n,1);
for k=1:n
    a = Alpha(k);
    h = Hurst(k);
    func = @(z) (1+z)^a-2^(a*h-1)*(1+abs(z)^a);
    z = fzero(func, [-1 1]);
    EC(k) = 2*z/(1+z^2);
end
```

# ДОДАТОК Б
## Відомість атестаційної роботи магістра

| Позначення | Найменування | Дод. відомості |
|---|---|---|
| | Текстові документи | |
| **1.** | Пояснювальна записка | 90 с. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | Інші документи | |
| | | |
| **2.** | Презентаційні матеріали | 20 плакатів |
| | | |
| | | |
| | | |
| | | |
| | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| . | . | Прізвище та ініціали. | Під п. | Дата | |
| Розробив | | Бєлоусова К.В. | | | Оцінювання фрактальних характеристик руху Леві |
| Перевірив | | Шергін В.Л. | | | |
| Н.контр. | | Малєєва І.А. | | | |
| Затв. | | Філатов В.О. | | | |

| Шифр групи | Код напр./спец. |
|---|---|
| СШІм-19-1 | 122 |
| ХНУРЕ кафедра ШІ | |