Introductory lecture no 5 *< lecture realized in 2006, now rendered in English>*
on Basics of Data Processing

(Matlab 6.x 7.x environment's examples on visualization topics)

Within realized lectures, concerning presentation of some of transformation functions, concerning data of 2D images with use of *Image Processing Toolbox*, one decided to present some of the examples of applications of previously related group of functions, as well as, of some specific syntax of commands, applied in handling of 1D/2D/3D data. First of all, one decided to present example of code, used in generation of fractal objects, which is similar to the famous fern leaf:

```
%Artur Bernat all rights reserved
%15 April 2006, fern's fractal with stochastic transforms.
%[M,im_final]=fern(szx,szy,magnif,iter,nop);
%szy,szx <=dimensions of 2D map(output size doubled in x)
%magnif  <=magnification coeff. for the figure
%iter    <=number of iteraton,
%nop     <=number of intervals in getting of frames
%[M,img]=fern(150,300,30,95000,30); % default call params.
function [M,im_final]=fern(szx,szy,magnif,iter,nop);
A=[0.0 0.85 0.2 -0.15]; %linear transforms. coeffs.
B=[0.0 0.04 -0.26 0.28];
C=[0.0 -0.04 0.23 0.26];
D=[0.16 0.85 0.22 0.24];
E=[0.0 0.0 0.0 0.0];
F=[0.0 1.6 1.6 0.44];
step=iter/nop;
MAX_TRANSF=4;
mapsA=zeros(szy,szx); mapsB=zeros(szy,szx);
x=0;y=0;
nrp = ceil(MAX_TRANSF.*rand(1,iter));
[min(nrp(:)) max(nrp(:))]
% background;  1-8  ; 9nth control colour
paltsA=[0.1 0.25 0.30; 0 1 0;0.2 1 0; 0.5 1 0; 1 1 0;...
             1 0 0; 1 0 0; 1 0.2 0 ; 1 0.5 0 ; 1 1 0];
M=moviein(nop+2);
axis manual;
j=0;%control counter for catching of frames
for i=1:iter,
    kolor=uint8(nrp(i));
    xl=A(kolor).*x+B(kolor).*y+E(kolor);
    y=C(kolor).*x+D(kolor).*y+F(kolor);
    x=xl;
    Y=1+szy/16+magnif.*y;
    X=1+szx/2+magnif.*x;
    mapsA(uint16(1+Y.*sign(Y)),uint16(1+X.*sign(X)))=uint8(1+nrp(i));
    mapsB(uint16(1+Y.*sign(Y)),uint16(1+X.*sign(X)))=uint8(5+nrp(i));
    if mod(i,step)==0
     imshow([flipud(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
     axis([0 szx+szx 0 szy]); axis manual; j=j+1; [i j]
     M(:,j)=getframe;
    end;
end;
im_final=ind2rgb([flipud(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
movie(M);
movie2avi(M,'fern.avi','compression','Cinepak','FPS',12);
close;
```

Fig.1Exemplary given code `fern.m`, used in generation of the famous fern leaf

Some specific solution of the function presented above (fig.1) is the application of two indexed maps *`mapsA`* and *`mapsB`* in task of generation of set of points, which iteratively form the related object. These maps are to be concatenated with use of *`[]`* operand, both in animation generation process, during iteratively plot fractal object, and in final generation of bitmap *`im_final`*, which is produced outside of the global *`for`*. loop.

Such a solution gives possibility for more flexible use, definition, as well as, on-the-fly redefinition of color palette. Alternatively adopted solution would be much more difficult in implementation of for instance data handling, in their RGB or HSV representation.

The very algorithm of generation of the fractal object, in spite of the fact, that it has been called deterministic algorithm, has been provided with random set of initially imposed conditions. Wandering of point's coordinates on *XY* reference plane has been described here with especially simple, linear combinations of coefficients, as well as, of currently taken coordinates:

$$\begin{aligned} x_{i+1} &= A * x_i + B * y_i + E \\ y_{i+1} &= C * x_i + D * y_i + F \end{aligned} \qquad (1)$$

With some other, distinct values of elements of the coefficient tables `A..F`, one actually obtained another commonly known algorithm, for generation of another fractal object:

```
%Artur Bernat all rights reserved
%15 April 2006, drag's fractal with stochastic transforms.
%[M,im_final]=drag(szx,szy,magnif,iter,nop);
%szy,szx <=dimensions of 2D map(output size doubled in x)
%magnif  <=magnification coeff. for the figure
%iter    <=number of iteraton,
%nop     <=number of intervals in getting of frames
%[M,img]=drag(150,300,25,30000,30); % default call params.
function [M,im_final]=drag(szx,szy,magnif,iter,nop);
A=[ 0.824074  0.088272];%linear transform. coeffs.
B=[ 0.281482  0.520988];
C=[-0.212346 -0.463889];
D=[ 0.864198 -0.377778];
E=[-1.88229   0.78536];
F=[-0.110607  8.095795];
step=iter/nop;
MAX_TRANSF=2;
mapsA=zeros(szy,szx);mapsB=zeros(szy,szx);
x=0;y=0;
nrp = ceil(MAX_TRANSF.*rand(1,iter));
[min(nrp(:)) max(nrp(:))]
%background;  1-8 palette's color  ;
paltsA=[0.15 0.25 0.45; 0 1 0;0.2 1 0; 0.5 1 0; 1 1 0;...
              1 0 0; 1 0 0; 1 0.2 0 ; 1 0.5 0 ; 1 1 1];
M=moviein(nop+2);
axis manual;
j=0;%control counter for catching of frames
for i=1:iter,
    kolor=uint8(nrp(i));
    x1=A(kolor).*x+B(kolor).*y+E(kolor);
    y=C(kolor).*x+D(kolor).*y+F(kolor);
    x=x1;
    Y=(1+szy/16+magnif.*y);
    X=(1+szx/16+magnif.*x);
    mapsA(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(1+nrp(i));
    mapsB(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(5+nrp(i));
    if mod(i,step)==0
     imshow([flipud(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
     axis([0 szx + szx 0 szy]); axis manual; j=j+1; [i j]
     M(:,j)=getframe;
 end;
end;
im_final=ind2rgb([flipud(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
movie(M);
movie2avi(M,'drag.avi','compression','Cinepak','FPS',12);
close;
```

Fig.2Exemplary given code `drag.m`*, used in generation of the famous dragon object*

At first glance, the above implementation (on figure 2) possess two times shorter vectors `A...F`. (in numbers of their row elements). The deriving process, due to which such a famous fractal object has been produced, is rather vague and unknown. What is relevant is the fact, that in most cases, any randomly or experimentally introduced slight alternations within elements of the `A...F` matrices lead mostly to abrupt deteriorations or even divergence of quite robust and steady algorithms, based on (1) relationship. Finally, the use of 3-elements' rows in `A..F` matrices, with some specifically set coefficients (reciprocally occurred values of zeros and 0.5), gives possibility for generation of the famous Sierpinsky triangle:

```
%Artur Bernat all rights reserved
%15 April 2006, triangle's fractal with stochastic transforms.
%[M,im_final]=triangle(szx,szy,magnif,iter,nop);
%szy,szx <=dimensions of 2D map(output size doubled in x)
%magnif  <=magnification coeff. for the figure
%iter    <=number of iteraton,
%nop     <=number of intervals in getting of frames
%[M,img]=triangle(150,300,130,10000,30); %default call params.
function [M,im_final]=triangle(szx,szy,magnif,iter,nop);
A=[ 0.5 0.5 0.5];%linear transform. coeffs.
B=[ 0   0   0  ];
C=[ 0   0   0  ];
D=[ 0.5 0.5 0.5];
E=[ 0   0   0.5];
F=[ 0   1   0.5];
step=iter/nop;
MAX_TRANSF=3;
mapsA=zeros(szy,szx);mapsB=zeros(szy,szx);
x=0;y=0;
nrp = ceil(MAX_TRANSF.*rand(1,iter));
[min(nrp(:)) max(nrp(:))]
%background;  1-8 palette's color  ;
paltsA=[0.15 0.25 0.45;  0 1 0;0.2 1 0;  0.5 1 0;  1 1 0;...
              1 0 0;  1 0 0;  1 0.2 0 ;  1 0.5 0 ;  1 1 1];
M=moviein(nop+2);
axis manual;
j=0;%control counter for catching of frames
for i=1:iter,
    kolor=uint8(nrp(i));
    x1=A(kolor).*x+B(kolor).*y+E(kolor);
    y=C(kolor).*x+D(kolor).*y+F(kolor);
    x=x1;
    Y=(1+szy/16+magnif.*y);
    X=(1+szx/16+magnif.*x);
    mapsA(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(1+nrp(i));
    mapsB(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(5+nrp(i));
    if mod(i,step)==0
     imshow([fliplr(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
     axis([0 szx + szx 0 szy]); axis manual; j=j+1; [i j]
     M(:,j)=getframe;
 end;
end;
im_final=ind2rgb([fliplr(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
movie(M);
movie2avi(M,'triangle.avi','compression','Cinepak','FPS',12);
close;
```

Fig. 3Exemplary given code `triangle.m`, in generation of the famous Sierpinsky triangle

In all the algorithms, which have been so far presented above, one protects the wandering point (i.e its iteratively resolved coordinates) from passing outlines of the screen (namely: one prohibits it from passing the borders of the physically imposed size of the two maps: `mapsA` and `mapsB`). A simple scaling, realized by multiplication of the newly iteratively resolved coordinates (either negative or positive) by its sign, provides only nonnegative output in the plotting results. And this task is being realized with use of `1+sign(X).*X` and `1+sign(Y).*Y` sub-expressions, respectively.

Basically, one readily needs purely random or pseudo-random algorithm initialization. For this aim, one for instance could implement a superior calling function in sub-sessions of subsequently realized fractal object generations, with relatively small number of the produced points, as well as, with purely randomly realized initialization of point's coordinates: `x`,`y`.

Therefore, for this aim, one implemented a script `ferns.m`, which reciprocally calls two functions both from script `fern_rand.m`, and from script `fern_rand_jet.m`:

```
%Artur Bernat, auxilliary script for fern gens.
%function [imout]=ferns(size_x,size_y,magnif,iter,movieN,iterN)
%function [imout]=fernxn_xor(size_x,size_y,magnif,iter,movieN,iterN)
%size_x,size_y   <=input x,y size of bitmap to be generated
%magnif          <=input magnification coeff. for fern leaf
%iter            <=input iters. for each of subcall in fern_rand function
%movieN          <=input movieN frames' number per one avi film
%iterN           <=input global iteration in superior loop of generating
%[img_s_out]=ferns(1600,900,89,157300,3,4);%default call's parameters
%[img_s_out]=ferns(400,225,25,57300,3,4);%2nd set of call's parameters(fast)
function [imout]=ferns(size_x,size_y,magnif,iter,movieN,iterN)
[M,img]=fern_rand(size_x/2,size_y,magnif,iter,movieN,1);
imout=img;
state=0;
for i=1:iterN-1,
    state=bitxor(state,1);
    if state==1
     [M,img]=fern_rand(size_x/2,size_y,magnif,iter,movieN,1+i);
    else
     [M,img]=fern_rand_jet(size_x/2,size_y,magnif,iter,movieN,1+i);
    end;
%     imout=imadd(imout,img);
     for j=1:size_x,
         for k=1:size_y,
          if (img(k,j,1)>imout(k,j,1))
           imout(k,j,1)=img(k,j,1);
          end;
          if (img(k,j,2)>imout(k,j,2))
           imout(k,j,2)=img(k,j,2);
          end;
          if (img(k,j,3)>imout(k,j,3))
           imout(k,j,3)=img(k,j,3);
          end;
         end;
     end;
end;
%-------------------------------------------------------------%
```

Fig.4Exemplary given superior script `ferns.m`*, in generation of the famous fern leaf*

Implementation code embedded within `fern_rand.m` script, basically does not differs from the former one, presented on figure 1 above. The starting values are tailored randomly from within range [1,17] and [1,23] respectively. Due to random choose, in each of the iterations, accordingly to the sub-elements of the contents of vectors, derived from `A..F` matrices, it is possible to '*fulfill*' some peripheral elements-regions of the famous fern leaf, which could not be '*acquired*', even with use of extensively enlarged number of iterations, performed in one global generation session, with use of `fern.m` script. Next to this, calling of function realized in the second script `fern_rand_jet.m`, presents itself, as some kind of algorithmic colorization of the surface of the famous fern leaf with use of predefined/imbedded color palette `jet`. (Matlab – 6.5 and 7.1 environmental versions).

Implicitly assumed, cold and hot colors in color palette applied, are usually used in indications of height points locations on any 3D plots, which are lower and higher respectively, accordingly to `JET` palette scheme.

In such a case (figure 6B) one decided to divide the 64-graded `JET` palette into two parts (each of the 31-elements) and to adopt them to impose colored pattern on two generated fern leafs, accordingly to *Y* coordinate of the wandering point (which is iteratively resolved).

In the result, in the following superior script one inlets some mixed contributions of the RGB components from calling of the two inferior scripts.

```
%Artur Bernat all rights reserved
%15 April 2006, fern's fractal with stochastic transforms.
%[M,im_final]=fern_rand(szx,szy,magnif,iter,nop);
%szy,szx <=dimensions of 2D map(output size doubled in x)
%magnif  <=magnification coeff. for the figure
%iter    <=number of iteraton,
%nop     <=number of intervals in getting of frames
%[M,img]=fern_rand(150,300,30,95000,30); % default call params.
function [M,im_final]=fern_rand(szx,szy,magnif,iter,nop,Niter);
A=[0.0 0.85 0.2 -0.15]; %linear transforms. coeffs.
B=[0.0 0.04 -0.26 0.28];
C=[0.0 -0.04 0.23 0.26];
D=[0.16 0.85 0.22 0.24];
E=[0.0 0.0 0.0 0.0];
F=[0.0 1.6 1.6 0.44];
step=iter/nop;
MAX_TRANSF=4;
mapsA=zeros(szy,szx); mapsB=zeros(szy,szx);
x=23.*rand(1,1)
y=17.*rand(1,1)
nrp = ceil(MAX_TRANSF.*rand(1,iter));
[min(nrp(:)) max(nrp(:))]
% background;  1-8  ; 9nth control colour
paltsA=[0.1 0.25 0.30; 0 1 0;0.2 1 0; 0.5 1 0; 1 1 0;...
             1 0 0; 1 0 0; 1 0.2 0 ; 1 0.5 0 ; 1 1 0];
M=moviein(nop+2);
axis manual;
j=0;%control counter for catching of frames
for i=1:iter,
    kolor=uint8(nrp(i));
    x1=A(kolor).*x+B(kolor).*y+E(kolor);
    y=C(kolor).*x+D(kolor).*y+F(kolor);
    x=x1;
    Y=1+szy/16+magnif.*y;
    X=1+szx/2+magnif.*x;
    mapsA(uint16(1+Y.*sign(Y)),uint16(1+X.*sign(X)))=uint8(1+nrp(i));
    mapsB(uint16(1+Y.*sign(Y)),uint16(1+X.*sign(X)))=uint8(5+nrp(i));
    if mod(i,step)==0
     imshow([flipud(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);hold on;
     title(['Global iter: ' int2str(Niter) ', local inter no. ' int2str(j)]);
     axis([0 szx+szx 0 szy]); axis manual; j=j+1; [i j]
     M(:,j)=getframe;
    end;
end;
im_final=ind2rgb([flipud(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
movie(M);
RandName=100+899*rand(1,1);
movie2avi(M,['fernxn_rand' int2str(RandName) '.avi'],'compression','Cinepak','FPS',3);
disp(['Number of superior calling iteration: ' int2str(Niter)]);
close;
```

Fig.5 Exemplary given code of interior calling `ferns_rand.m`*, in generation of famous fern leaf*

Exemplary code presented on figure 6 is slightly extended with extra calculations of currently resolved color, originated from `JET` palette (within one of the two intervals, accordingly to *Y* coordinates and either left or right screen panels, respectively):

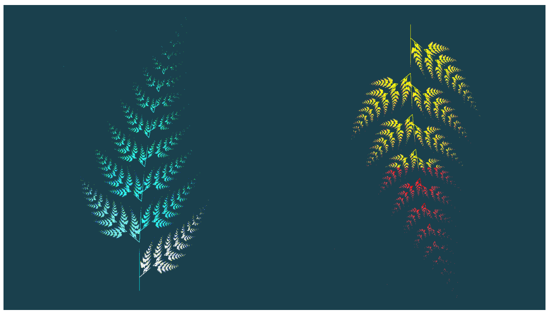


Fig 6A `JET` *palette with contributions derived from another bounded color palette, inadvertently results in disruptive color changing on right sided panel (right fern leaf colored with hot colors)*

```
%Artur Bernat all rights reserved
%15 April 2006, fern's fractal with stochastic transforms.
%AND randomized starting x0,y0 point of calculations
%[M,im_final]=fern_rand_jet(szx,szy,magnif,iter,nop,Niter);
%szy,szx <=dimensions of 2D map(output size doubled in x)
%magnif  <=magnification coeff. for the figure
%iter    <=number of iteraton,
%nop     <=number of intervals in getting of frames
%Niter   <=number of superior call iteration(just info)
%[M,img]=fern_rand_jet(150,300,30,95000,30,1); % default call params.
function [M,im_final]=fern_rand_jet(szx,szy,magnif,iter,nop,Niter);
A=[0.0 0.85 0.2 -0.15]; %linear transforms. coeffs.
B=[0.0 0.04 -0.26 0.28];
C=[0.0 -0.04 0.23 0.26];
D=[0.16 0.85 0.22 0.24];
E=[0.0 0.0 0.0 0.0];
F=[0.0 1.6 1.6 0.44];
step=iter/nop;
MAX_TRANSF=4;
mapsA=zeros(szy,szx); mapsB=zeros(szy,szx);
x=23.*rand(1,1);%randomized starting x0
y=17.*rand(1,1);%randomized starting y0
nrp = ceil(MAX_TRANSF.*rand(1,iter));
[min(nrp(:)) max(nrp(:))]
% background;  1-8  ; 9nth control colour
%paltsA=[0.1 0.25 0.30; 0 1 0;0.2 1 0; 0.5 1 0; 1 1 0;...
%              1 0 0; 1 0 0; 1 0.2 0 ; 1 0.5 0 ; 1 1 0];
paltsA=JET;     %64-graded JET predefined palette;
paltsA(1,:)=0; % first from jet palette is equal to zero
palstepY=szy/31;
M=moviein(nop+2);
axis manual;
j=0;%control counter for catching of frames
for i=1:iter,
    kolor=uint8(nrp(i));
    x1=A(kolor).*x+B(kolor).*y+E(kolor);
    y=C(kolor).*x+D(kolor).*y+F(kolor);
    x=x1;
    Y=1+szy/16+magnif.*y;
    X=1+szx/2+magnif.*x;
    mapsA(uint16(1+Y.*sign(Y)),uint16(1+X.*sign(X)))=uint8(1 +mod( ( (1+Y.*sign(Y)) /palstepY ),31));
    mapsB(uint16(1+Y.*sign(Y)),uint16(1+X.*sign(X)))=uint8(32+mod( ( (1+Y.*sign(Y)) /palstepY ),31));
    if mod(i,step)==0
     imshow([flipud(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);hold on;
     title(['Global iter: ' int2str(Niter) ', local inter no. ' int2str(j)]);
     axis([0 szx+szx 0 szy]); axis manual; j=j+1; [i j]
     M(:,j)=getframe;
    end;
end;
im_final=uint8(ind2rgb([flipud(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA));
movie(M);
RandName=100+899*rand(1,1);
movie2avi(M,['fernxn_rand_jet' int2str(RandName) '.avi'],'compression','Cinepak','FPS',3);
disp(['Number of superior calling iteration: ' int2str(Niter)]);
close;
```

Rys6B Exemplary given code for interior callling of `ferns_rand_jet.m` *script, in the generation of the famous fern leaf.*

Resuming this part of the currently provided lecture (lecture number 5), it could be said, that there were no simple recipe for migrating process from C/C++ languages, accordingly to the already existed implementations or even programmers' thinking, into Matlab programming environmental specifics, in programming practice, variable handling etc.

Likewise, firstly made superstitions, mostly invalid, made *ad hoc*, at first glance, accordingly to this or that function, in performing some specifically aimed task, usually requires later some practice and exhaustive and watchful verification process, as it was in case of creation of some particular mixture of data colors, derived from two distinct functions, both used in generation process of the very same fractal object. It seemed, that calling of the function *imadd* would suffice, and moreover it seemed that the contents of the two images would be to be simply summed up. However in practice, it occurred that this tiny and apparently unimportant, inferior task were to be solved as watchfully, as the main superior task.
In relation to contents for the former lectures (number 4A and 4B) one decided to present below some script, which demonstrates progressive image quality deteriorations, as the functions of narrowed set of information, extracted based on processing with use of Discrete Cosine Transform (DCT)
Usually, there occurs this or that demo-script, performed on quite interactive level, which could serve for educational aim. However this time, contents of the script dctmovie, which is presented below, creates two animations, written on hard disk. One of them visualizes variable quality of the 2D image after performing simple transformation based on a) DCT (spectral data representation) b) reduction of set of DCT coefficients, c) realization of the inverse DCT (IDCT):

```
%all rights rewerved Artur Bernat
%a script for DCT transf. deterioration of image quality
%function [Min,Mcoeff,img]=dctmovie(im,nsteps,limit);
%im_in=imread('rice.png');  %example of image
% [M1,M2,ims]=dctmovie(im_in,20,300); %default call params
function [Mim,Mcoeff,img]=dctmovie(im,nsteps,limit);
%im=rgb2gray(im);
szx=size(im,2);
szy=size(im,1);
dct_im=dct2(im);
Mim=moviein(nsteps);
Mcoeff=moviein(nsteps);
interv=limit/nsteps;
for i=1:nsteps,
 dct_im(abs(dct_im)<i*interv)=0;
 dct_bw=zeros(szy,szx);
 dct_bw(abs(dct_im)>0)=1;
 idct_im=idct2(dct_im)/255;
 subplot(2,1,1);imshow(idct_im);hold on;
 title(['Frame nr. ' int2str(i) ' ,thresh: ' int2str(i*interv)]);
 Mim(:,i)=getframe;
 subplot(2,1,2);imshow(dct_bw);
 Mcoeff(:,i)=getframe;
end;
img=idct_im;
RandName=100+899*rand(1,1);
movie2avi(Mim,   ['dctmovie' int2str(RandName) '.avi'],'compression','Cinepak','FPS',9);
close;
movie2avi(Mcoeff,['dctcoeffs' int2str(RandName) '.avi'],'compression','Cinepak','FPS',9);
close;
```

Fig. 7 Exemplary given code dctmovie.m, for animation, which visualizes the progressive deterioration in quality of the 2D image contents

Second animation presents itself a binarized map of DCT coefficients, and it aims to visualize contributions of the DCT coefficients in loose DCT 2D image recreation (i.e. with use of both DCT and IDCT transforms, and set of coefficients contractions, realized between these two stage of data processing).

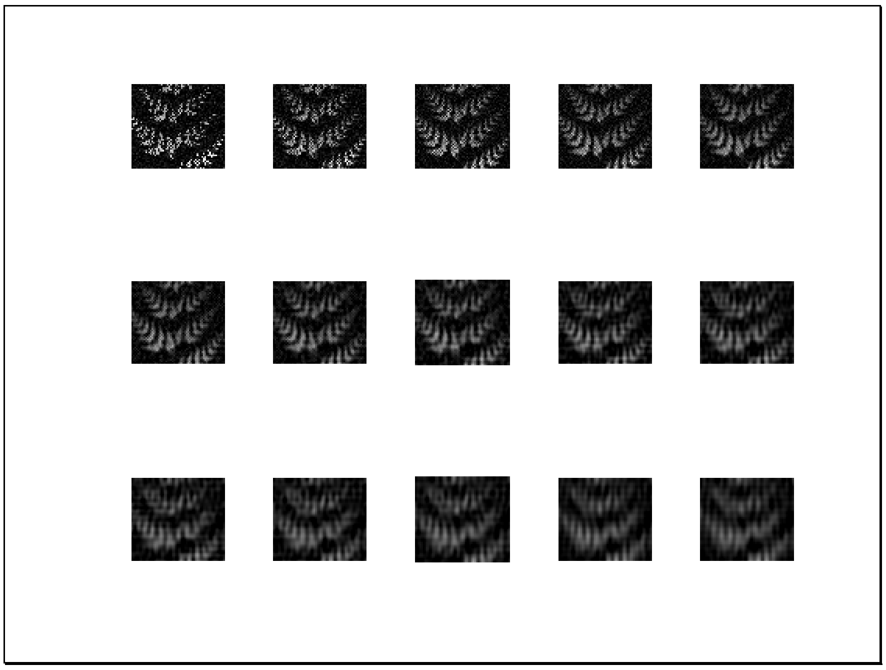


Fig. 8 Fifteen frames of animation in form of sub-plots, as the results of performance of loose DCT compression, due to script's calling, which is presented above, on figure 7

On figure 8 above, the progressive deterioration of the quality of 2D intesity image has been presented, with upper limit for the value of the DCT coefficients set at 600, with 300 by 300 pixels of data of 2D image being processed.

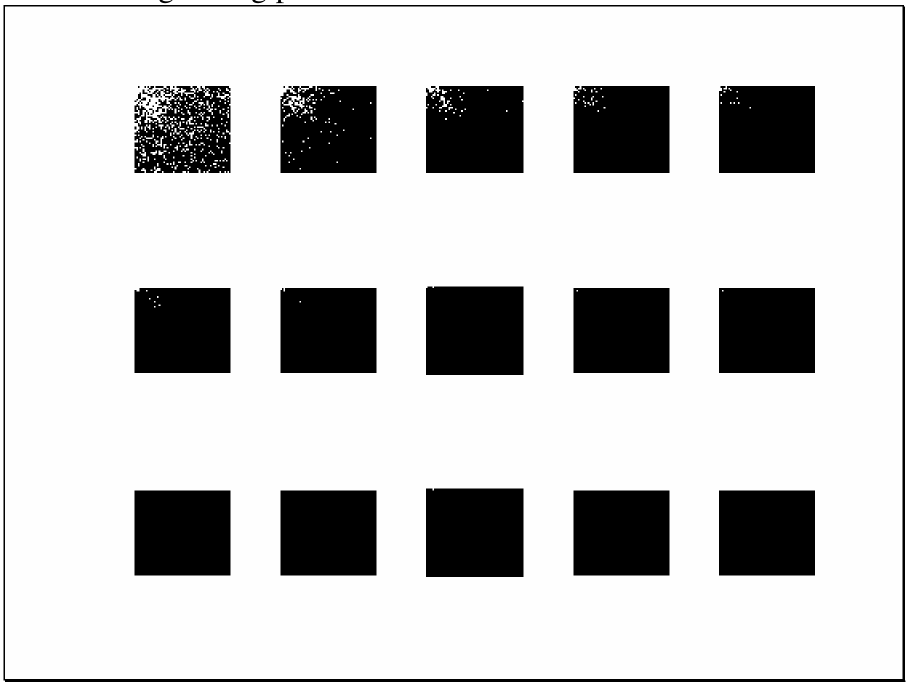


Fig. 9 Fifteen frames of animation, analogously generated, accordingly to the results on figure 8 above,. which present themselves the very DCT coefficients, in form of white pixels, and which are selected for further data processing, in function of the progressive data set contraction (i.e. with increasing level of loose compression with use of DCT/IDCT transforms).

Based on comparison of the results, both presented on figure 8 and 9, respectively, including both quality of the 2D image recreated, and range of the set of the DCT coefficients selected for further stage of data processing one could say, that the whole performance of the loose DCT compression is quite efficient.

```
C:\MATLAB6p1\work\fftmovie.m
26 kwiecień 2006
%rotate white rectangle with FFT transform visualisation
%function [Mfft,Mfttb,ims]=fftmovie(sz)
%[Morg,Mfftt,Mfftb,imss]=fftmovie(100);%default call parameters
function [M,M1,M2,ims]=fftmovie(sz)
M=moviein(4);
M1=moviein(4);
M2=moviein(4);
ims=zeros(sz);
ims((sz/8:7*sz/8),(7*sz/16:9*sz/16))=1;
j=1;
hnd=subplot(1,3,1);
for i=1:4,
    imrot=rot90(ims,i);
 subplot(1,3,1);imshow(imrot);hold on;title('original');hold off;
 F1=fft2(imrot);
 F2=log(abs(F1));
 subplot(1,3,2);imshow(F2);colormap(jet);colorbar;hold on;
 title('FFT image');hold off; M1(:,i)=getframe;
 F3=fftshift(F1);
 subplot(1,3,3);imshow(log(abs(F3)));colormap(jet);colorbar;hold on;
 title('shifted FFT image');hold off;M2(:,i)=getframe;
 M(:,i)=getframe(hnd);
end;
RandName=100+899*rand(1,1);
movie2avi(M ,['FFT2c' int2str(RandName) '.avi'],'compression','Cinepak','FPS',3);
movie2avi(M1,['FFT2a' int2str(RandName) '.avi'],'compression','Cinepak','FPS',3);
movie2avi(M2,['FFT2b' int2str(RandName) '.avi'],'compression','Cinepak','FPS',3);
close;
```

Fig.10 Visualization of some results of calling of some script, based in data processing on Fast Fourier Transform

On one hand, the analyzed series of the recreated 2D images, presented on figure 8, do not indicate for dramatic deterioration in the output image quality. Simultaneously, on another hand, the range of the DCT coefficients selected can be seriously contracted. Analogously, with use of visualization in form of the generated animation, one decided to present as well some potential possibilities of application Fast Fourier Transform (however without presentation here of any output results).

Introductory lectures no 6A&B<lectures realized in 2006, now rendered in English> on Basics of Data Processing

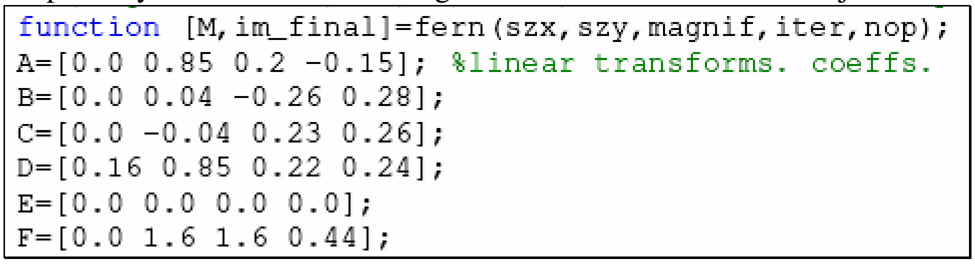
(Matlab 6.x, 7.x environment's examples on visualization topics, continued…)

Within frame of lectures, concluding some introductory notions and talks about possible applications of functions, derived from *Image Processing Toolbox* in Matlab environment, one decided to slightly extend the analysis of the formulas and rules concerning generation process of fractal motifs. The notion: *object*, as well as, notion: fractal *motif*, both these will be used here, as the visual aspect of some output result (results iteratively resolved on screen for fractal's creator or programmer), in opposition to strict *formula*, that is plainly rendered, and interpreted, as mere mathematical relationship.

In the formerly given lecture (lecture no 5, presented on a verge of the two spring's months: April/May 2006, on topics: *visualization examples*) one gave simple formula for generation of the famous *Sierpinsky triangle*, with use of script: `triangle.m` (another, less commonly adopted term to name it, is for instance: *Sierpinsky gasket*), *fern leaf* with use of script: `fern.m` and silhouette of the famous *dragon*, with use of script: `drag.m`.

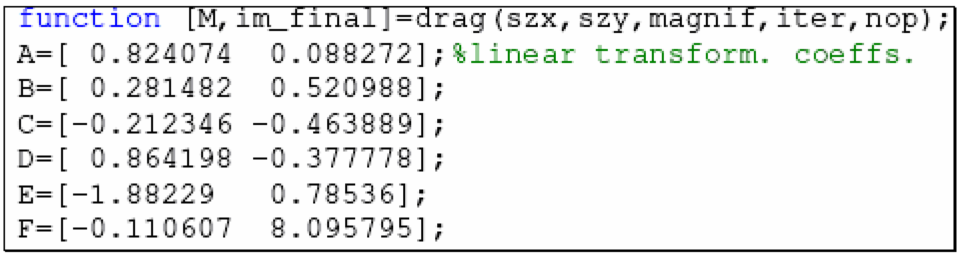
$$\begin{aligned} x_{i+1} &= A * x_i + B * y_i + E \\ y_{i+1} &= C * x_i + D * y_i + F \end{aligned} \qquad (1)$$

If one had started the analysis of the more embedded common sense or logics in values and arrangements of the coefficients in tables `A..F`, for the last two scripts' examples, this would probably have led the programmers to nothing relevant. But our aim of the following considerations in this matter, is to obtain the most generic clues about how in consciously and at least partially in controlled manner generate or form new fractal objects:



```
function [M,im_final]=fern(szx,szy,magnif,iter,nop);
A=[0.0 0.85 0.2 -0.15]; %linear transforms. coeffs.
B=[0.0 0.04 -0.26 0.28];
C=[0.0 -0.04 0.23 0.26];
D=[0.16 0.85 0.22 0.24];
E=[0.0 0.0 0.0 0.0];
F=[0.0 1.6 1.6 0.44];
```

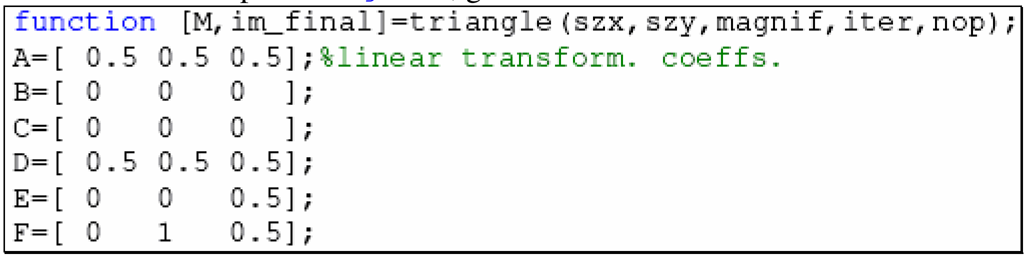
Fig. 1 Header of the script `fern.m` *given with values of one-dimensional tables: A, B, C, D, E, F*



```
function [M,im_final]=drag(szx,szy,magnif,iter,nop);
A=[ 0.824074  0.088272];%linear transform. coeffs.
B=[ 0.281482  0.520988];
C=[-0.212346 -0.463889];
D=[ 0.864198 -0.377778];
E=[-1.88229   0.78536];
F=[-0.110607  8.095795];
```

Fig. 2 Header of the script `drag.m` *given with values of one-dimensional tables: A, B, C, D, E, F*

On the other hand, few moments bestowed to pondering on, as to how to tailor coefficients in tables `A..F` for first script `triangle.m`, give instant illumination and conclusion:



```
function [M,im_final]=triangle(szx,szy,magnif,iter,nop);
A=[ 0.5 0.5 0.5];%linear transform. coeffs.
B=[ 0   0   0  ];
C=[ 0   0   0  ];
D=[ 0.5 0.5 0.5];
E=[ 0   0   0.5];
F=[ 0   1   0.5];
```

Fig. 3 Header of the script `triangle.m` *given with the values of one-dimensional tables: A, B, C, D, E, F*

The coefficients from tables `E` i `F` , now have become some inherently embedded set of coordinates, randomly chosen, in <u>*spanning*</u> operation of the sub-regions on *XY* plane. Within this sub-region, the point, of iteratively resolved coordinates, moves while drawing fractal's *motif*. One can notice, that shape of the famous Sierpinsky gasket, that means - its outline in from of regular triangle, spanned between points (0.0, 0), (1.0, 0.0) and (0.5, 0.5), directly originates from values of the *x* and *y* coordinates, taken column-wisely from matrices: `E` i `F`.

What is more, one could say, that there were no *skew correlations* in calculations of the currently iteratively resolved *x* coordinate, accordingly to the currently resolved *y* coordinate, originated from the former iteration (coefficients of `B` table are actually equal to zero). Likewise, one could say, that there were no skew correlations, between currently iteratively resolved *y* coordinate, accordingly to *x* coordinate, originated from the former iterations. That means the coefficients of `C` table are also equal to zero.

In case of the `A` and `D` tables, their coefficients constitute themselves some set of randomly chosen (from iteration to another iteration) <u>*contraction*</u> coefficients of the currently drawn point. These contraction operations mean, that the iteratively resolved coordinates, from one iteration to another, are constantly subdued to contraction operation, between some (*x*, *y*) coordinates, determined in the former iteration, and the origin of the coordinates set, that means point (0, 0). They could be called also the coefficients of the *linear correlation*, because they take into account solely a relationship between *x* coordinates from one to another iterations, as well as, they take also into account a relationship between *y* coordinates from one to another iterations. In case of the script `triangle.m,` these values (of *x* and *y* coordinates) are idempotent, and are equal to 0.5. That means, that from iteration to another iteration, each of the former coordinate values (both *x* and *y*) are multiplicatively shortened by half (obviously, not taking into account the influences of another coefficients, here playing only a additive character, see relationship (1)).

On the other hand, from time to time, the coefficients' values from tables `E` i `F`, renders the whole drawing process, of the iteratively resolved point's coordinates, similar to situation, where the algorithm starts from the beginning, except this, that the point is being drawn on the very outskirts of the whole fractal *object*. For the object drawn with use of script `triangle.m`, these are the following 3 points: (1.0, 0.5) or (0.5, 0.5) or (0.0, 0.0).

Therefore, concluding the considered case with the involved relationship (1), in drawing fractal objects, one deals with two distinct and mutually opposite phenomena, which should be preserved in balance from one iteration to another, giving in the result some specific aspect the output fractal object. Firstly, there is a phenomenon of <u>*expansion*</u> or <u>*spanning*</u> of the iteratively resolved (*x*, *y*) coordinates, based on spanning of coordinates, which iteratively lead to expansion of values of coordinates, for the point, which is wandering on *XY* plane. And for this, the coefficients of the `E` and `F` tables are responsible (of the additive character).

Secondly, another phenomenon, opposite to the former, is the <u>*contraction*</u> of the iteratively resolved (*x*, *y*) coordinates, which are drawn between one of the spanning point somewhere on the *XY* plane and origin of the set of coordinates, that means point (0.0, 0.0). And for this the coefficients of the the `A` and `D` tables are responsible.

Based on the concluding remarks and notes presented here above, one can actually derive, with use of the evolution trial-and-error methods, or more conveniently spoken: one can generate a totally new brand of fractal *motifs*.

While proving this hypothesis, actually here below some results of the trials and experiments have been presented, mostly concerning alternations in contents of the `A..F` tables, all of them originally derived from the values, which are in conformity of the famous Sierpinsky triangle generation algorithm.

Presented in this lecture, aspects of the generated fractal objects, have been created, based on implicit arguments calling, which are available with context-help invocations for the script named: *`fract.m`* `(please use this context-help, and highlights-by-clicks and F9 function keys for command invoking)`:
`>> help fract`

```
Artur Bernat all rights reserved
12 May 2006, triangle's fractal with stochastic transforms.
[M,im_final]=fract(szx,szy,magnif,iter,nop,opts,DoF,x0,y0);
szy,szx <=dimensions of 2D map(output size doubled in x)
magnif  <=magnification coeff. for the figure
iter    <=number of iteraton,
nop     <=number of intervals in getting of frames
opts    <=number of matrix of linear transformation coefficients
DoF     <=DoF in choosing of coefficients in random walks
x0,y0   <=coords. starting points,default: (0,0) middle of the screen
[M,img]=fract(300,450,180,90000,16,0,3,-0.5,-1); %DEFAULT TRIANGLE NO.0
[M,img]=fract(300,450,175,90000,16,1,5,-0.4,-1);%default params.FRACT.no.1
[M,img]=fract(300,450,175,90000,16,2,3,-0.5,-1);%default params.FRACT.no.2
[M,img]=fract(300,450,160,90000,16,3,5,-0.6,-1);%default params.FRACT.no.3
[M,img]=fract(300,450,160,90000,16,4,4,-0.6,-1);%def. call prms.FRACT.no.4
[M,img]=fract(300,450,175,90000,16,5,5,-0.6,-1);%def. call prms.FRACT.no.5
[M,img]=fract(300,450,75,90000,16,6,5); %default call params. fract.no.6
[M,img]=fract(300,450,125,90000,16,6,4); %default call params. fract.no.6B
[M,img]=fract(300,450,125,60000,16,6,6); %default call params. fract.no.6C
[M,img]=fract(300,450,125,60000,16,6,5); %default call params. fract.no.6D
```

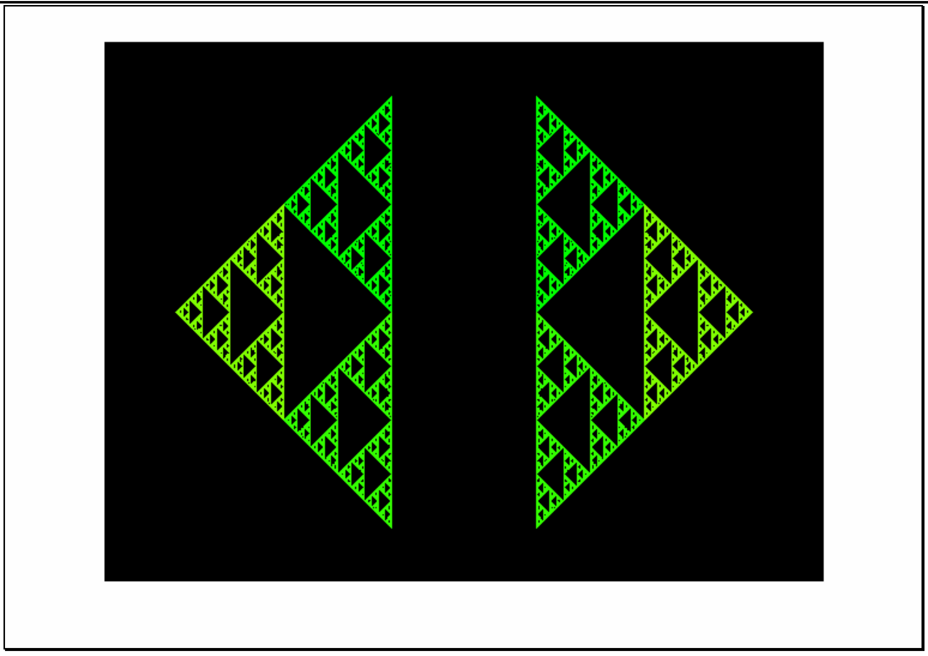


Fig.1 Initially obtained, doubled image of the famous Sierpinksy triangle,
option no 0 of implicitly given argument list in script `fract.m`

With slightly alternated values for the coefficients in rows 5th and 6th in *`MAT1`* matrix (and this corresponds to *`E`* i *`F`* tables in the formerly considered script *`triangle.m`*), one can actually obtain extra fluctuations in color of the merged famous Sierpinsky triangle.

ATTENTION: for convenience and improved proficiency of the all doings here related in this lecture, in handling some specific calling argument lists, please use:
- context-help calling with implicitly given headers of the scripts here considered in this lecture,
- then, please make multiply double-clicking with mouse pointer on selected line (among many others) for the printouts of context-help calling, for the aim of highlighting the desired arguments' list (of desired single line of the chosen argument list among many other lines of exemplary given command lists)
- call them (single or many lines highlighted) with special function key F9.

Motifs, in spite of the previous definitions placed somewhere above in this lectures, would be rather some distinctive and salient occurrences of their self-similarities, in relationship to whole part of the ***fractal object***., thus here motifs merged only as some self-similarities in merged patterns.

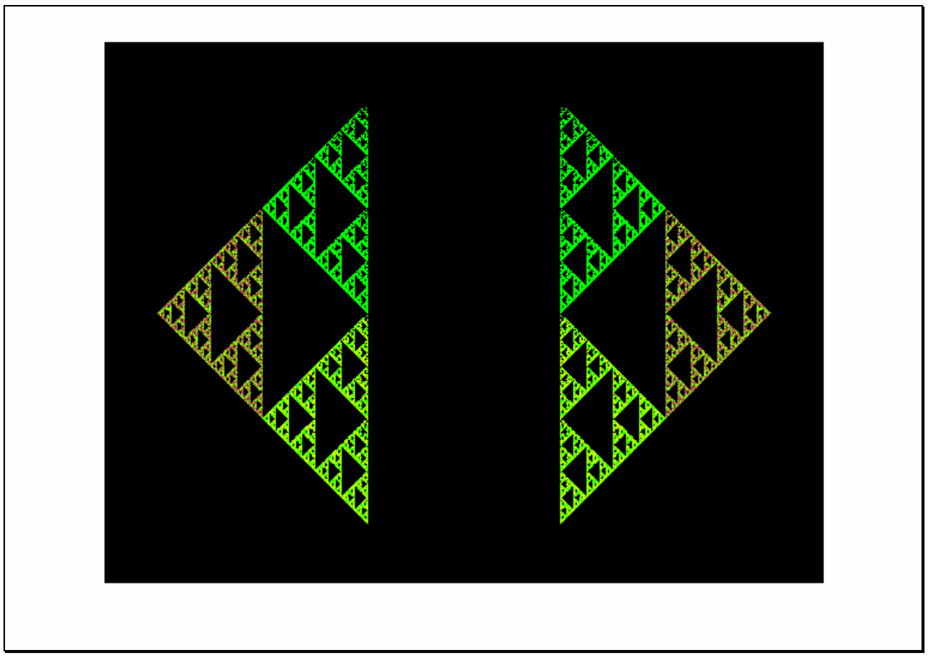


Fig.2 The generation of the object, accordingly to argument calling list with use of script `fract.m` *for option no 1 (first implicitly given set of arguments)*

Yet, slightly other and distinct coefficients' values for the matrix `MAT2` in script `fract.m`, in relation to matrix `MAT1` in 5th and 6th row, create possibilities for generation of the discussed object in skew direction, accordingly to *XY* plane. What's left, is the question, whether it is really perspective shortening (in 3D space), or merely skew distortions on *XY* reference plane.

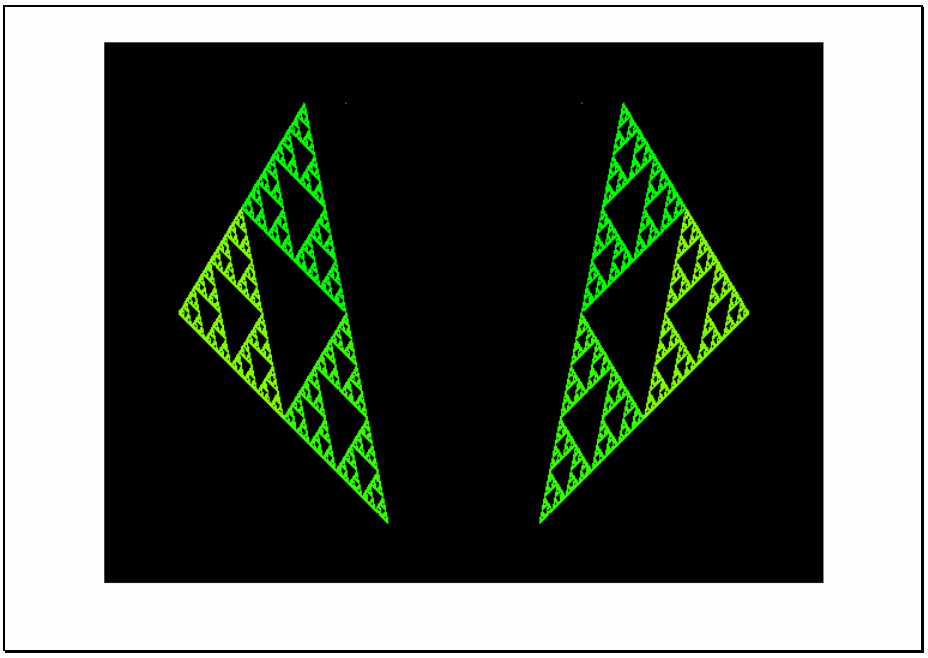
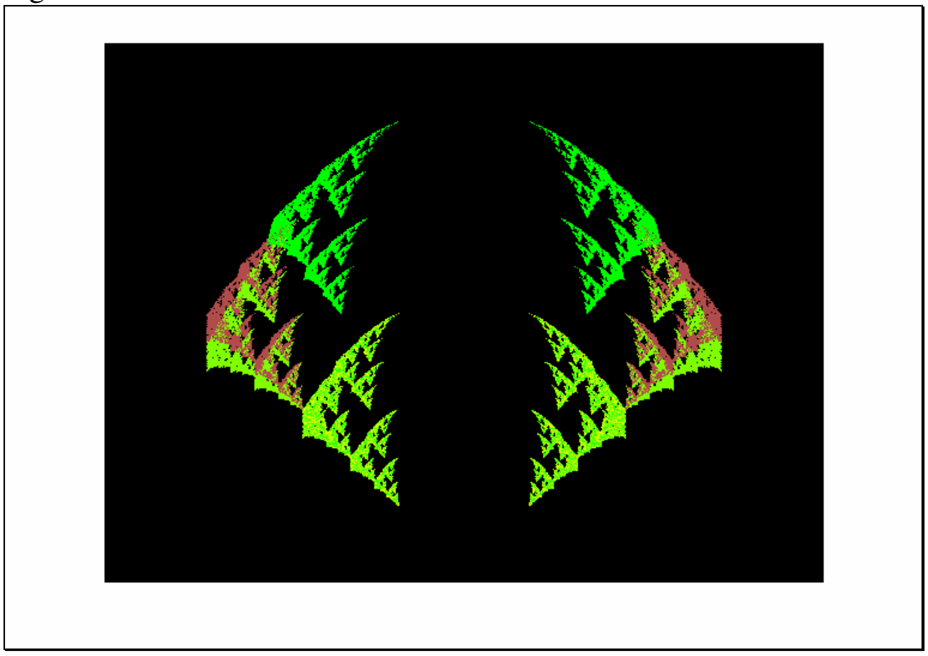


Fig.3 The generation of the object, accordingly to script `fract.m` *for option no 2 with implicitly defined calling argument list*

So far, the coefficients responsible for skew correlations, (which should determine multiplicative influences of *x* on *y* and *y* on *x* coordinates) were zeroed. However, if one introduced non-zeroed values into one of the columns in the second row of the matrix `MAT3` and moreover in one of the columns in the third row of the matrix `MAT3,` then for the first time, the whole generation scheme in algorithms of the famous Sierpinsky triangle would be seriously changed:



Rys.4 The generation of the object, accordingly to the script `fract.m` *for option no 3 in implicitly given calling argument list*

Moreover, a little much more boldly put trials and experiments with alternation of the rest of the coefficients, responsible for skew correlations phenomena, lead to another, distinctly innovative relationship. Though, firstly it should be said, that it is a solution with decreased number of degree of freedom, in randomly realized selection of all of the coefficients from within elements of *`MAT4`* matrix. So far in the considerations, related here in extensively above in this lecture, the degree of freedom was equal to 3, 5, 3 and 5, from one example of implementation to another, respectively. But, currently it is below 5, and it is actually set to be equal to 4.

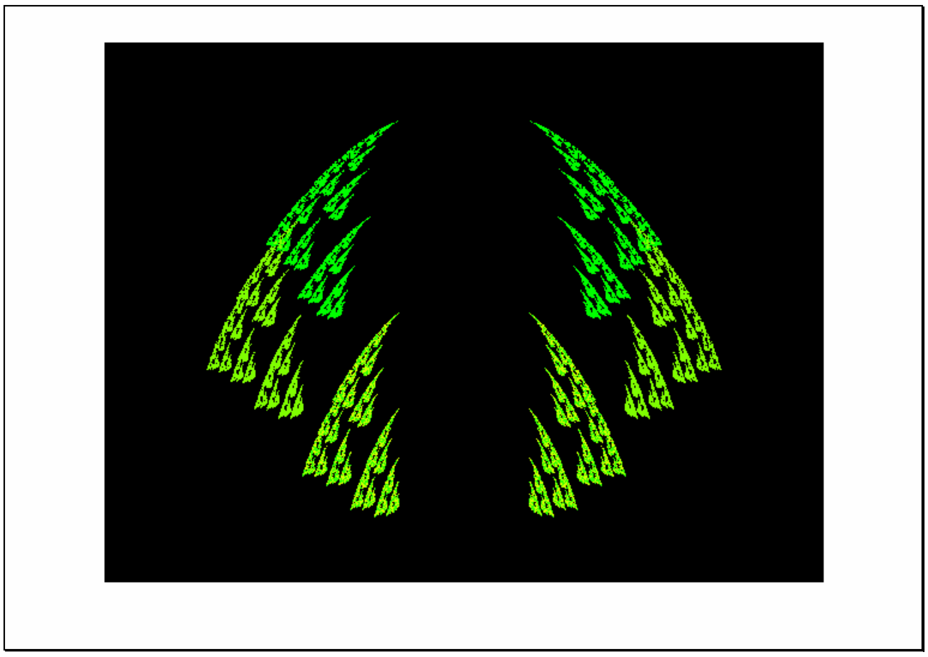


Fig.5 The generation of the object accordingly to the script `fract.m` *for the option no 4 of implicitly defined calling argument list*

Now, currently, there will be introduced clearly distinct values of the coefficients, which are responsible for occurring of linear correlations. In another words, the values of coefficients, which play a role in contracting of the iteratively resolved coordinates, down to point (0, 0).

Let it be of course values 0.25 and 0.5 in various columns of the 1st and 4th rows of *MAT5* matrix, with obviously preserved, so far made alternations of some of elements, already existed in *MAT4* matrix, which had been gradually introduced in former examples:

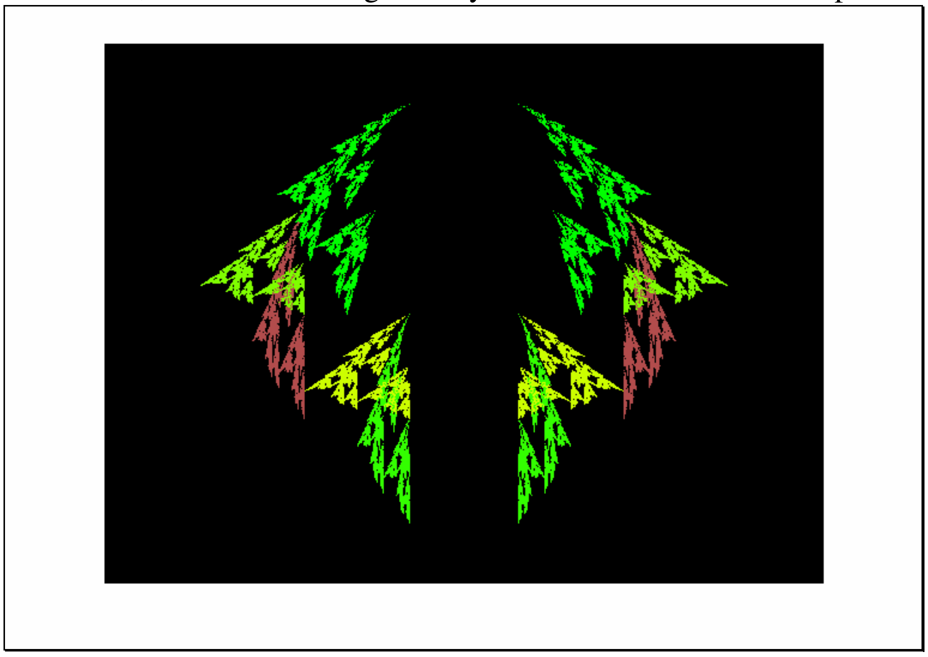


Fig.6 The generation of the object, accordingly to the script fract.m with option no 5 of implicitly defined calling argument list

Finally, in option no 6 of the implicitly defined and exemplary given argument calling list, for the scripts *fract.m*, one actually adopts rule of arithmetic average, which is roughly equal or nearly equal to zero, for the 2nd and 3rd row of the *MAT6* matrix (and these are coefficients of the skew correlations) and moreover: for the 5th and 6th rows of the *MAT6* matrix (and these are the coefficients, which determine size of the sub-region, within which there is spanned trajectory of the iteratively resolved coordinates of the currently drawn point).

ATTENTION: as to the last remarks, this must be possibly some convex polygon, or sum of convex sub-polygons, for the sub-regions, within which the trajectory of the drawn point is being contained, spanned by some of the extremely located points, which in turn are produced due to so-called (in this lecture) *spanning* function.

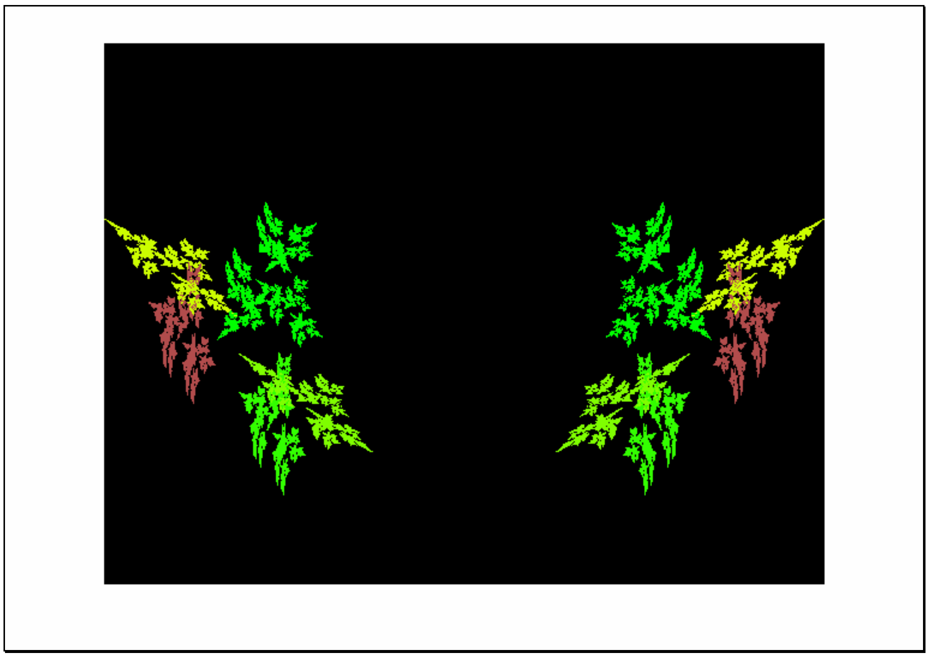


Fig.7 The generation of the object, accordingly to the script `fract.m` *for the option no 6, with implicitly defined calling argument list*

Introducing of both positive and negative coefficients' values, and alternations in both rows of *linear correlations*, and in rows of *skew correlations*, and moreover in rows of the *spanning functions* for the regions, occupied by the drawn object, give altogether in the result the outputs totally dissimilar to the famous Sierpinsky triangle. The above resulted object (presented on figure 7) has been obtained for the matrix *`MAT6`*, of coefficients, at number of degree of freedom equal to 5. However, there are also some predefined calling argument lists, implicitly tagged, as options no *6B*, *6C* and *6D*. In invocations of the option no 6B, the degree of freedom has been decreased from 5 to 4.

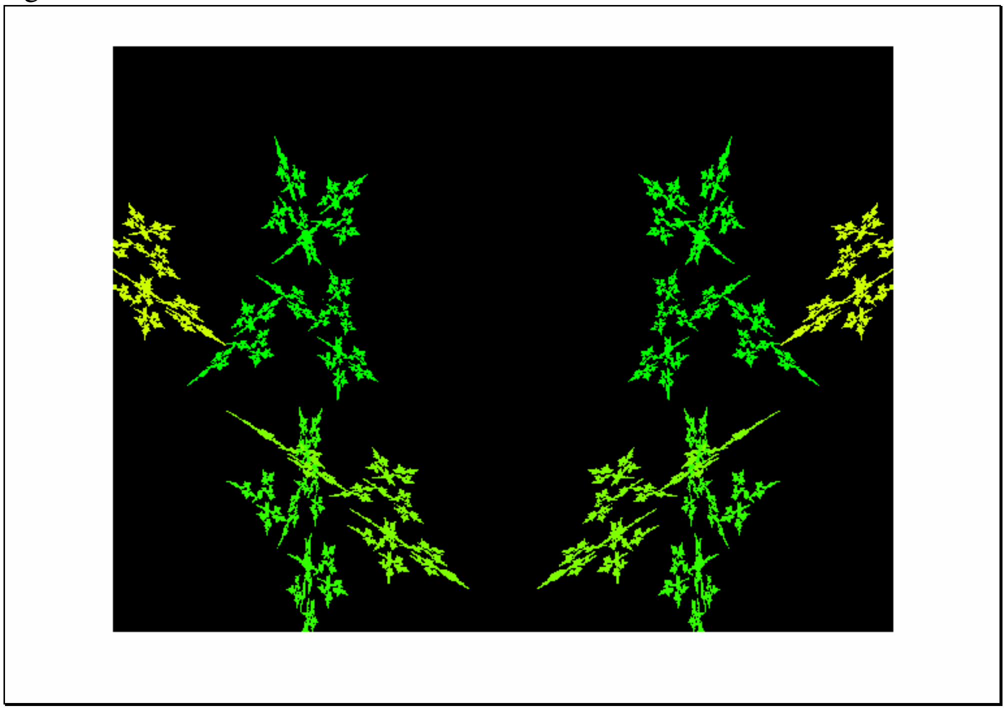


Fig.8 The generation of the object, accordingly to the script `fract.m` *with the option no 6B, of implicitly defined calling argument list*

Lastly, in next calling argument list, implicitly tagged as option no *6C*, the degree of freedom has been set on value equal to 6, at the very same magnification coefficient, as defined previously, in the case with output results presented on figure 8:

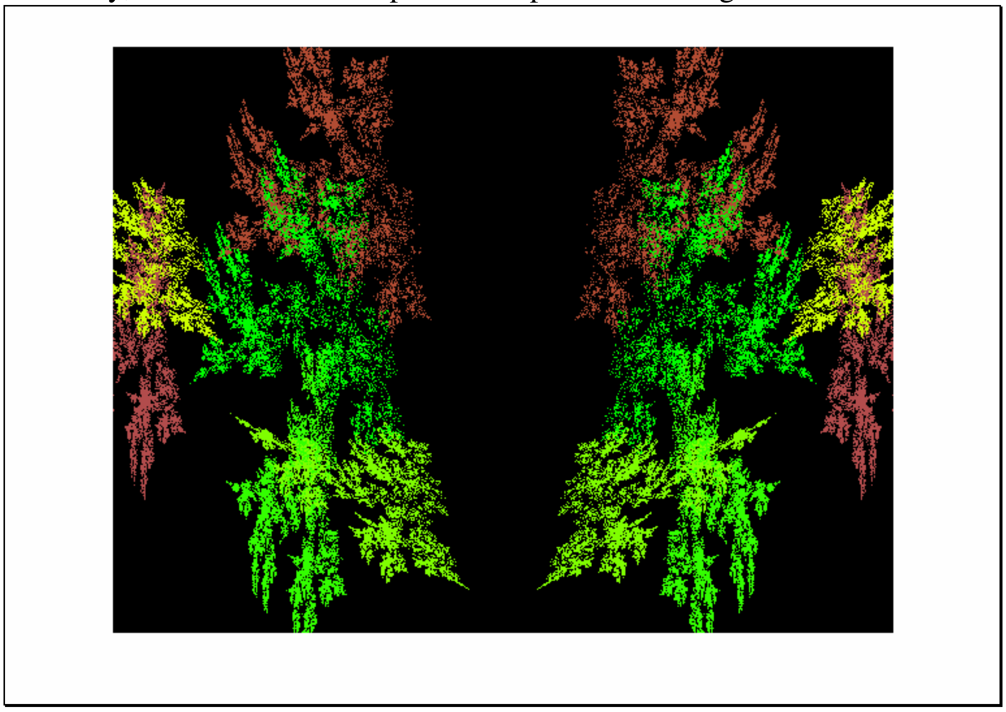


Fig.9 The generation of the object, with use of the script `fract.m` *with option no 6C, of implicitly defined calling argument list*

Finally, one more approach of implicitly defined calling argument list, tagged as 6D:

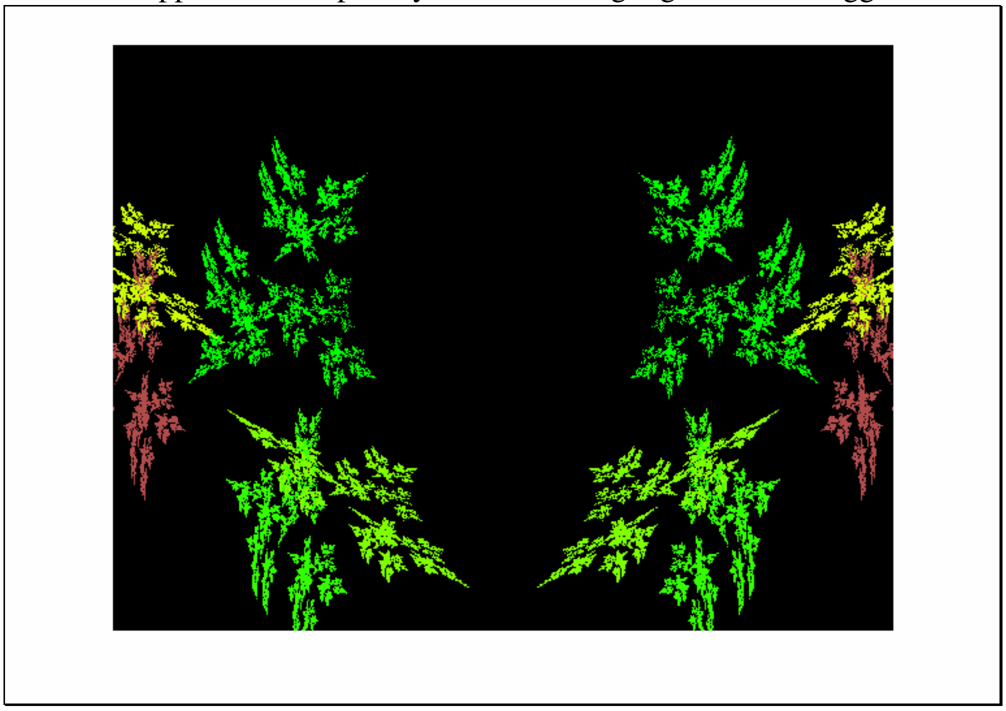
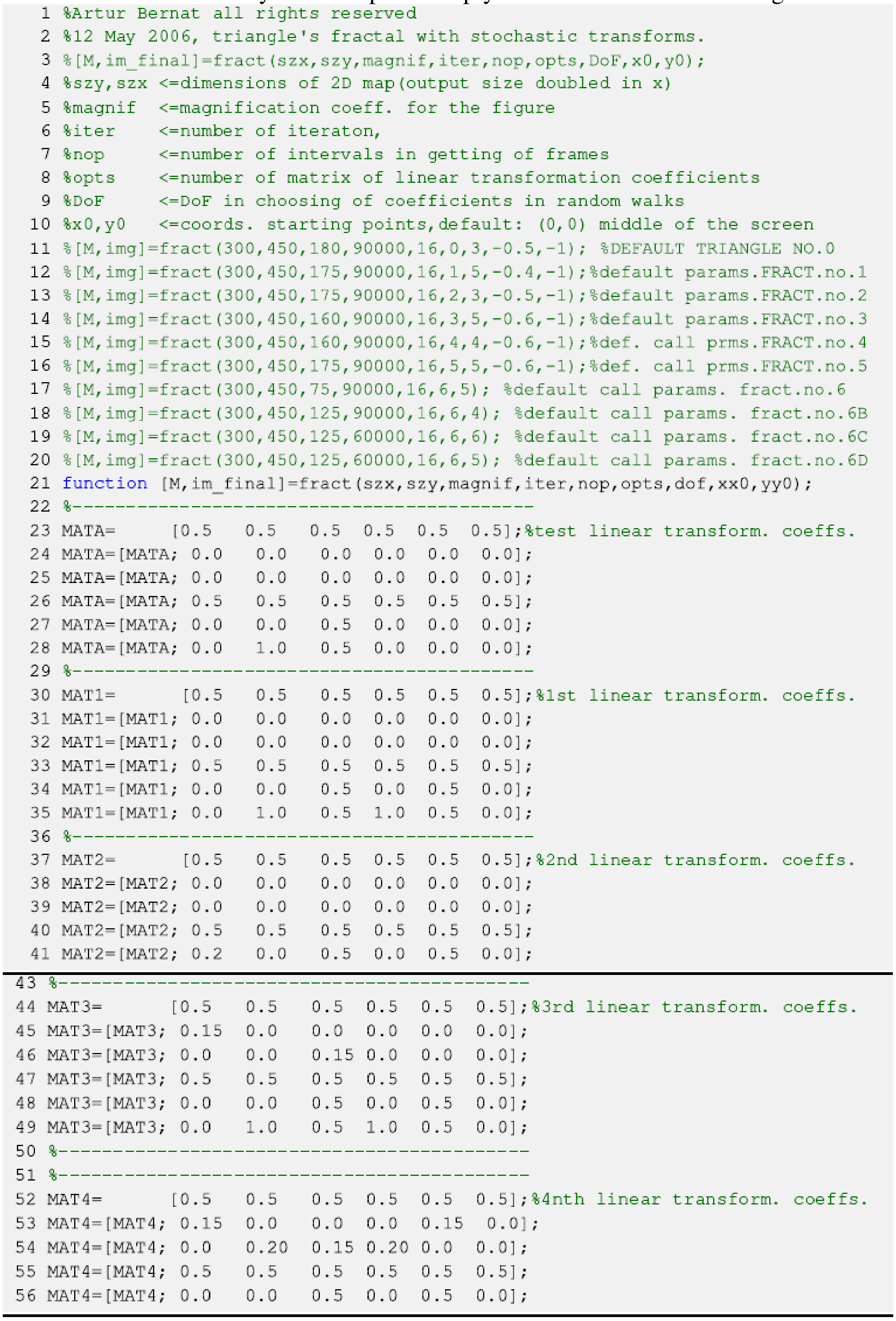


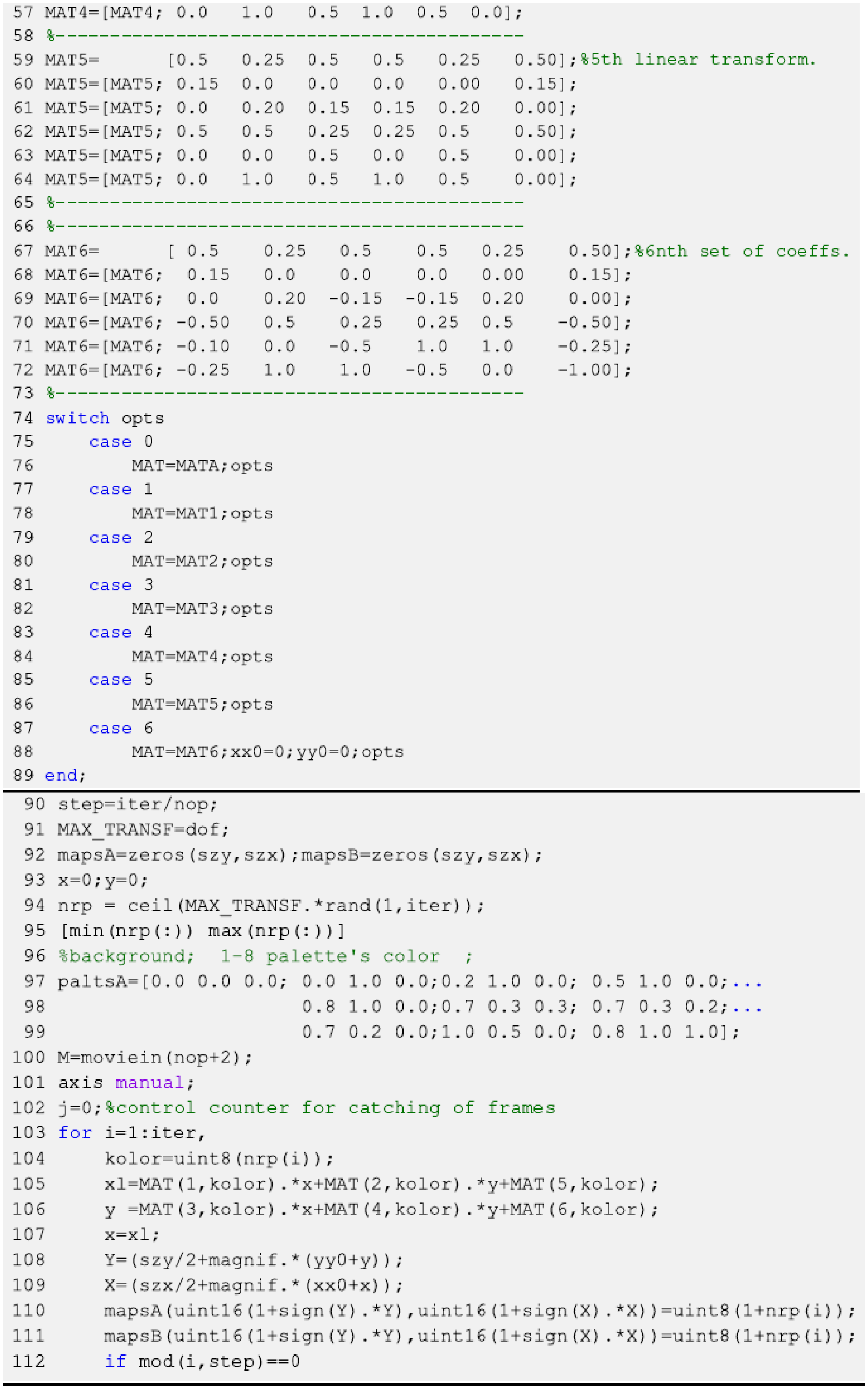
Fig.10 The generation of the object, with use of the script `fract.m` *with option no 6D, of implicitly defined calling argument list*

Distinctly set, another from the previously used, magnification coefficient, too large accordingly to the displayed outlines of the objects on the computer screen, would provoke occurrence of extra phenomenon: the reflecting of parts of the drawn object, while coordinates of the iteratively resolved point simply clash with screen’s four edges.



```
%Artur Bernat all rights reserved
%12 May 2006, triangle's fractal with stochastic transforms.
%[M,im_final]=fract(szx,szy,magnif,iter,nop,opts,DoF,x0,y0);
%szy,szx <=dimensions of 2D map(output size doubled in x)
%magnif  <=magnification coeff. for the figure
%iter    <=number of iteraton,
%nop     <=number of intervals in getting of frames
%opts    <=number of matrix of linear transformation coefficients
%DoF     <=DoF in choosing of coefficients in random walks
%x0,y0   <=coords. starting points,default: (0,0) middle of the screen
%[M,img]=fract(300,450,180,90000,16,0,3,-0.5,-1); %DEFAULT TRIANGLE NO.0
%[M,img]=fract(300,450,175,90000,16,1,5,-0.4,-1);%default params.FRACT.no.1
%[M,img]=fract(300,450,175,90000,16,2,3,-0.5,-1);%default params.FRACT.no.2
%[M,img]=fract(300,450,160,90000,16,3,5,-0.6,-1);%default params.FRACT.no.3
%[M,img]=fract(300,450,160,90000,16,4,4,-0.6,-1);%def. call prms.FRACT.no.4
%[M,img]=fract(300,450,175,90000,16,5,5,-0.6,-1);%def. call prms.FRACT.no.5
%[M,img]=fract(300,450,75,90000,16,6,5); %default call params. fract.no.6
%[M,img]=fract(300,450,125,90000,16,6,4); %default call params. fract.no.6B
%[M,img]=fract(300,450,125,60000,16,6,6); %default call params. fract.no.6C
%[M,img]=fract(300,450,125,60000,16,6,5); %default call params. fract.no.6D
function [M,im_final]=fract(szx,szy,magnif,iter,nop,opts,dof,xx0,yy0);
%-------------------------------------------
MATA=      [0.5   0.5   0.5  0.5   0.5   0.5];%test linear transform. coeffs.
MATA=[MATA; 0.0   0.0   0.0  0.0   0.0   0.0];
MATA=[MATA; 0.0   0.0   0.0  0.0   0.0   0.0];
MATA=[MATA; 0.5   0.5   0.5  0.5   0.5   0.5];
MATA=[MATA; 0.0   0.0   0.5  0.0   0.0   0.0];
MATA=[MATA; 0.0   1.0   0.5  0.0   0.0   0.0];
%-------------------------------------------
MAT1=      [0.5   0.5   0.5  0.5   0.5   0.5];%1st linear transform. coeffs.
MAT1=[MAT1; 0.0   0.0   0.0  0.0   0.0   0.0];
MAT1=[MAT1; 0.0   0.0   0.0  0.0   0.0   0.0];
MAT1=[MAT1; 0.5   0.5   0.5  0.5   0.5   0.5];
MAT1=[MAT1; 0.0   0.0   0.5  0.0   0.5   0.0];
MAT1=[MAT1; 0.0   1.0   0.5  1.0   0.5   0.0];
%-------------------------------------------
MAT2=      [0.5   0.5   0.5  0.5   0.5   0.5];%2nd linear transform. coeffs.
MAT2=[MAT2; 0.0   0.0   0.0  0.0   0.0   0.0];
MAT2=[MAT2; 0.0   0.0   0.0  0.0   0.0   0.0];
MAT2=[MAT2; 0.5   0.5   0.5  0.5   0.5   0.5];
MAT2=[MAT2; 0.2   0.0   0.5  0.0   0.5   0.0];
%-------------------------------------------
MAT3=      [0.5   0.5   0.5  0.5   0.5   0.5];%3rd linear transform. coeffs.
MAT3=[MAT3; 0.15  0.0   0.0  0.0   0.0   0.0];
MAT3=[MAT3; 0.0   0.0   0.15 0.0   0.0   0.0];
MAT3=[MAT3; 0.5   0.5   0.5  0.5   0.5   0.5];
MAT3=[MAT3; 0.0   0.0   0.5  0.0   0.5   0.0];
MAT3=[MAT3; 0.0   1.0   0.5  1.0   0.5   0.0];
%-------------------------------------------
%-------------------------------------------
MAT4=      [0.5   0.5   0.5  0.5   0.5   0.5];%4nth linear transform. coeffs.
MAT4=[MAT4; 0.15  0.0   0.0  0.0   0.15   0.0];
MAT4=[MAT4; 0.0   0.20  0.15 0.20  0.0   0.0];
MAT4=[MAT4; 0.5   0.5   0.5  0.5   0.5   0.5];
MAT4=[MAT4; 0.0   0.0   0.5  0.0   0.5   0.0];
```

Fig. 11 First page of the printout, for the script `fract.m`

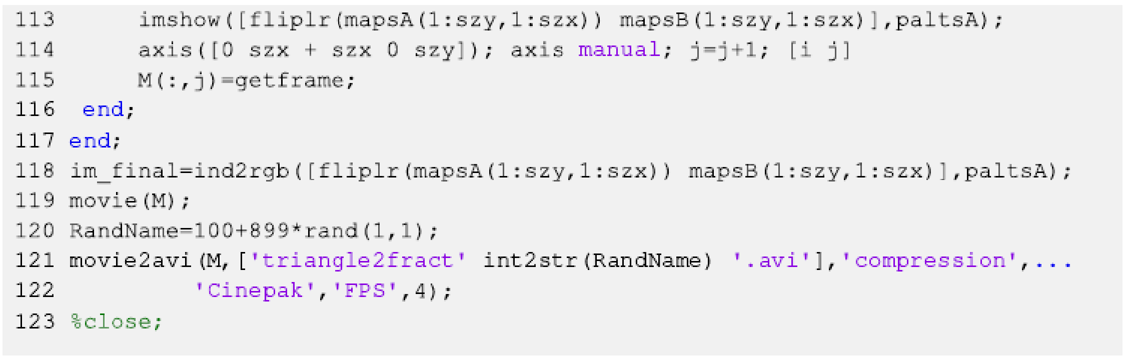


```
MAT4=[MAT4; 0.0   1.0   0.5  1.0  0.5  0.0];
%----------------------------------------------
MAT5=       [0.5   0.25  0.5   0.5   0.25   0.50];%5th linear transform.
MAT5=[MAT5; 0.15  0.0   0.0   0.0   0.00   0.15];
MAT5=[MAT5; 0.0   0.20  0.15  0.15  0.20   0.00];
MAT5=[MAT5; 0.5   0.5   0.25  0.25  0.5    0.50];
MAT5=[MAT5; 0.0   0.0   0.5   0.0   0.5    0.00];
MAT5=[MAT5; 0.0   1.0   0.5   1.0   0.5    0.00];
%----------------------------------------------
%----------------------------------------------
MAT6=       [ 0.5    0.25   0.5    0.5   0.25    0.50];%6nth set of coeffs.
MAT6=[MAT6;  0.15   0.0    0.0    0.0   0.00    0.15];
MAT6=[MAT6;  0.0    0.20  -0.15  -0.15  0.20    0.00];
MAT6=[MAT6; -0.50   0.5    0.25   0.25  0.5    -0.50];
MAT6=[MAT6; -0.10   0.0   -0.5    1.0   1.0    -0.25];
MAT6=[MAT6; -0.25   1.0    1.0   -0.5   0.0    -1.00];
%----------------------------------------------
switch opts
    case 0
        MAT=MATA;opts
    case 1
        MAT=MAT1;opts
    case 2
        MAT=MAT2;opts
    case 3
        MAT=MAT3;opts
    case 4
        MAT=MAT4;opts
    case 5
        MAT=MAT5;opts
    case 6
        MAT=MAT6;xx0=0;yy0=0;opts
end;
```

```
step=iter/nop;
MAX_TRANSF=dof;
mapsA=zeros(szy,szx);mapsB=zeros(szy,szx);
x=0;y=0;
nrp = ceil(MAX_TRANSF.*rand(1,iter));
[min(nrp(:)) max(nrp(:))]
%background;  1-8 palette's color  ;
paltsA=[0.0 0.0 0.0; 0.0 1.0 0.0;0.2 1.0 0.0; 0.5 1.0 0.0;...
                     0.8 1.0 0.0;0.7 0.3 0.3; 0.7 0.3 0.2;...
                     0.7 0.2 0.0;1.0 0.5 0.0; 0.8 1.0 1.0];
M=moviein(nop+2);
axis manual;
j=0;%control counter for catching of frames
for i=1:iter,
    kolor=uint8(nrp(i));
    xl=MAT(1,kolor).*x+MAT(2,kolor).*y+MAT(5,kolor);
    y =MAT(3,kolor).*x+MAT(4,kolor).*y+MAT(6,kolor);
    x=xl;
    Y=(szy/2+magnif.*(yy0+y));
    X=(szx/2+magnif.*(xx0+x));
    mapsA(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(1+nrp(i));
    mapsB(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(1+nrp(i));
    if mod(i,step)==0
```

Fig.12 Second page of printout, for the script `fract.m`

Thus, the very same set of parameters, but with distinct magnification coefficient, may result in producing quite different leading *motif* of the created fractal *object*, due to unavoidable interference of four edges of computer screen with randomly wandering drawing point.



```
        imshow([fliplr(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
        axis([0 szx + szx 0 szy]); axis manual; j=j+1; [i j]
        M(:,j)=getframe;
  end;
 end;
 im_final=ind2rgb([fliplr(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
 movie(M);
 RandName=100+899*rand(1,1);
 movie2avi(M,['triangle2fract' int2str(RandName) '.avi'],'compression',...
          'Cinepak','FPS',4);
 %close;
```

Fig. 13 Third page of printout, for the script `fract.m`

<u>Resuming, the final visual aspect of the fractal object created is determined by:</u>

1) values of set of coefficients of *linear correlations* (example with script formerly tagged as `triangle.m`, with 1D tables `A` and `D`)

2) values of set of coefficients of *skew correlations* (example with script formerly tagged as `triangle.m`, with 1D tables `B` and `C`)

3) values of set of coefficients, used in function of spanning sub-regions, within which the trajectory of the iteratively resolved point's coordinates is being obtained for the particular fractal object (example with script formerly tagged as `triangle.m` with 1D tables `E` and `F`)

<u>Moreover, some important role plays also:</u>

a) uniformity/non-uniformity of values of coefficients, taken into account row-wisely in `MAT` matrix of 6 by 6 elements (such a matrix, is basically used in transformation in implementation of the script `fract.m`)

b) positive /non-positive values of coefficients in each of the rows of `MAT` matrix

c) arithmetic average equal to or not equal to zero in all rows of matrix `MAT`

d) and potentially: std of values of coefficients in each of the rows of matrix `MAT`

<u>Another concluding remarks:</u>

a) with bigger degree of freedom, in random choosing of coefficients, one can possibly render task of introducing yet unknown, therefore innovative leading motifs of the created fractal object, much more easily; and such an object probably is characterized by some constant inherently embedded fractal dimensions, for there do occur with distinct saliency the features of self-similarities of sub-parts of the object, in relation to the whole object area,

b) one should also note, that increased number of degree of freedom, renders whole generation process more risky, accordingly to saliency and stability of the leading motifs, within areas of the generated fractal object,

c) based on increased number of degree of freedom, one would be able to consciously span this created fractal object on a bunch of several flat geometrical figures, i.e. triangles, rhombs, squares, pentagrams, hexagons, etc,

d) after series of initially realized analyses here on plane in this lectures, there are some substantial clues for starting analogously realized tasks, but in 3D space, or more generally in multi-dimensional space data representations, i.e. in R^n,

e) there does exist possibility for realization of spanning/contracting coordinates' function of the drawn point for the created fractal object, so does also there exist possibility for setting another non-zeroed in its coordinates point, i.e. there exists possibility for establishment another nonzero 'atractor' in one of the two basic operand's function, i.e. in contraction of point's coordinates.

For the case of the 3D space, the relationship (1) would be rendered as the following (accordingly to author's knowledge, concepts and intuition, a few years ago):

$$\begin{aligned} x_{i+1} &= A*x_i + B*y_i + C*z_i + J \\ y_{i+1} &= D*x_i + E*y_i + F*z_i + K \\ z_{i+1} &= G*x_i + H*y_i + I*z_i + L \end{aligned} \tag{2}$$

Obviously, there is readily need for checking another trivial possibility, like this - the expected coordinates value on *x* and *y* coordinates, respectively:

$$\begin{aligned} x_{i+1} &= A*x_i*\bar{E}(x_i - \overline{Ex})^2 + B*y_i*\bar{E}(x_i - \overline{Ey})^2 + E \\ y_{i+1} &= C*x_i*\bar{E}(y_i - \overline{Ex})^2 + D*y_i*\bar{E}(y_i - \overline{Ey})^2 + F \end{aligned} \tag{3}$$

The relationship presented here above, should include moments of the second order, as the some kind of short term memories or trails of the previously determined trajectory of the drawn points, of coordinates iteratively resolved, with influence of lengths of this memory on leading *motifs* for the newly generated object.

This would be expected value, counted from within totally taken set of coordinates in whole trajectory of iteratively resolved points, or the expected values for the short term memories, with only a few of them (i.e. previously visited coordinates on screen) taken into account.

Likewise, biased or unbiased estimator of the expected value of the coordinates of the points iteratively resolved, would contribute some distinct differences.

Just one more approach in trials and experiments - the proposed approach however, has not been yet extensively tested. This what is idempotent with (3):

$$\begin{aligned} x_{i+1} &= A*x_i*\bar{E}(x_i - \overline{Ex})^2 + B*y_i*\bar{E}(x_i - \overline{Ey})^2 + E \\ y_{i+1} &= C*x_i*\bar{E}(y_i - \overline{Ex})^2 + D*y_i*\bar{E}(y_i - \overline{Ey})^2 + F \end{aligned} \quad , \tag{4}$$

should actually be provided with another self-contracting mechanism in movements of point's coordinates iteratively resolved:

$$\begin{aligned} x_{i+1} &= A*x_i*\frac{1}{1+\bar{E}(x_i - \overline{Ex})^2} + B*y_i*\frac{1}{1+\bar{E}(x_i - \overline{Ey})^2} + E \\ y_{i+1} &= C*x_i*\frac{1}{1+\bar{E}(y_i - \overline{Ex})^2} + D*y_i*\frac{1}{1+\bar{E}(y_i - \overline{Ey})^2} + F \end{aligned} \tag{5}$$

Introductory lecture no 6C *(continued..)*
on Basics of Data Processing
(fractals motifs/objects spanned on trival geometric plan ar figures)

Accordingly to the content of part *A* of this lecture no *6*, one decided to present here some exemplary introduced fractal objects, generated with 5 and 6 degree of freedom, in random choosing of the values of coefficients, of both *linear* and *skew* correlations (both here considered with multiplicative character of summing up all of the components), and moreover in randomly realized selection of values of coefficients, which plays a role of spanning function of point's coordinates (iteratively resolved) (and these are the components considered in additive character).

Due to the fact, that the most important details of implementation, have been already given in previous part of this lectures, in elucidating the meaning of contents of the scripts: `triangle.m`, `fern.m`, `drag.m`, `fract.m` , here below only some extra details will be presented in much more condensed form. These are the following implicitly defined calling argument lists of newly implemented scripts: `fractpentagram.m` and `fracthexagon.m`.

```
>> help fractpentagram
```

```
Artur Bernat all rights reserved
13 May 2006, triangle's fractal with stochastic transforms.
[M,im_final]=fractpentagram(szx,szy,magnif,iter,nop,opts,DoF,x0,y0);
szy,szx <=dimensions of 2D map(output size doubled in x)
magnif  <=magnification coeff. for the figure
iter    <=number of iteraton,
nop     <=number of intervals in getting of frames
opts    <=number of matrix of linear transformation coefficients
DoF     <=DoF in choosing of coefficients in random walks
x0,y0   <=coords. starting points,default: (0,0) middle of the screen
[M,img]=fractpentagram(300,450,180,90000,16,0,3,-0.5,-1); %DEFAULT TRIANGLE NO.0
[M,img]=fractpentagram(300,450,75,90000,16,1,4,-0.0,-0.0);%default params.FRACT.no.1A
[M,img]=fractpentagram(300,450,75,90000,16,1,5,-0.0,-0.0);%default params.FRACT.no.1B
[M,img]=fractpentagram(300,450,75,90000,16,2,5,-0.0,-0.0);%default params.FRACT.no.2A
[M,img]=fractpentagram(300,450,75,90000,16,2,5,-0.0,-0.0);%default params.FRACT.no.2B
[M,img]=fractpentagram(300,450,75,90000,16,3,5,-0.0,-0.0);%default params.FRACT.no.3A
[M,img]=fractpentagram(300,450,75,90000,16,3,5,-0.0,-0.0);%default params.FRACT.no.3B
[M,img]=fractpentagram(300,450,75,90000,16,4,5,-0.0,-0.0);%def. call prms.FRACT.no.4
[M,img]=fractpentagram(300,450,75,90000,16,5,5,-0.0,-0.0);%def. call prms.FRACT.no.5
[M,img]=fractpentagram(300,450,75,90000,16,6,6); %default call params. fract.no.6
```

```
>> help fracthexagon
```

```
Artur Bernat all rights reserved
12 May 2006, triangle's fractal with stochastic transforms.
[M,im_final]=fracthexagon(szx,szy,magnif,iter,nop,opts,DoF,x0,y0);
szy,szx <=dimensions of 2D map(output size doubled in x)
magnif  <=magnification coeff. for the figure
iter    <=number of iteraton,
nop     <=number of intervals in getting of frames
opts    <=number of matrix of linear transformation coefficients
DoF     <=DoF in choosing of coefficients in random walks
x0,y0   <=coords. starting points,default: (0,0) middle of the screen
[M,img]=fracthexagon(300,450,180,90000,16,0,3,-0.5,-1); %DEFAULT TRIANGLE NO.0
[M,img]=fracthexagon(300,450,75,90000,16,1,4,-0.0,-0.0);%default params.FRACT.no.1A
[M,img]=fracthexagon(300,450,75,90000,16,1,6,-0.0,-0.0);%default params.FRACT.no.1B
[M,img]=fracthexagon(300,450,75,90000,16,2,5,-0.0,-0.0);%default params.FRACT.no.2A
[M,img]=fracthexagon(300,450,75,90000,16,2,6,-0.0,-0.0);%default params.FRACT.no.2B
[M,img]=fracthexagon(300,450,75,90000,16,3,5,-0.0,-0.0);%default params.FRACT.no.3A
[M,img]=fracthexagon(300,450,75,90000,16,3,6,-0.0,-0.0);%default params.FRACT.no.3B
[M,img]=fracthexagon(300,450,75,90000,16,4,6,-0.0,-0.0);%def. call prms.FRACT.no.4
[M,img]=fracthexagon(300,450,75,90000,16,5,5,-0.0,-0.0);%def. call prms.FRACT.no.5
[M,img]=fracthexagon(300,450,75,90000,16,6,6); %default call params. fract.no.6
```

Main remark here concerns the fact, that the irregular pentagram has been chosen here, as an outlines for spanning Sierpinsky-triangle-like fractal object. Possibly, five-arms-star is yet to be implemented, in much more analogous manner. Anyway, as to the second script, it spans the region of the fractal object on regular hexagon. Values of the 5th and 6th row, which are responsible the generation of point's coordinates (iteratively resolved) are approximated with 3, 4 digits (at decimal base), in accordance to the required the top-most precision in plotting of the fractal objects. The results of the implicitly defined calling argument lists for each of two scripts in form of the sub-plotting arrangements on one whole drawing panel, have been given below:

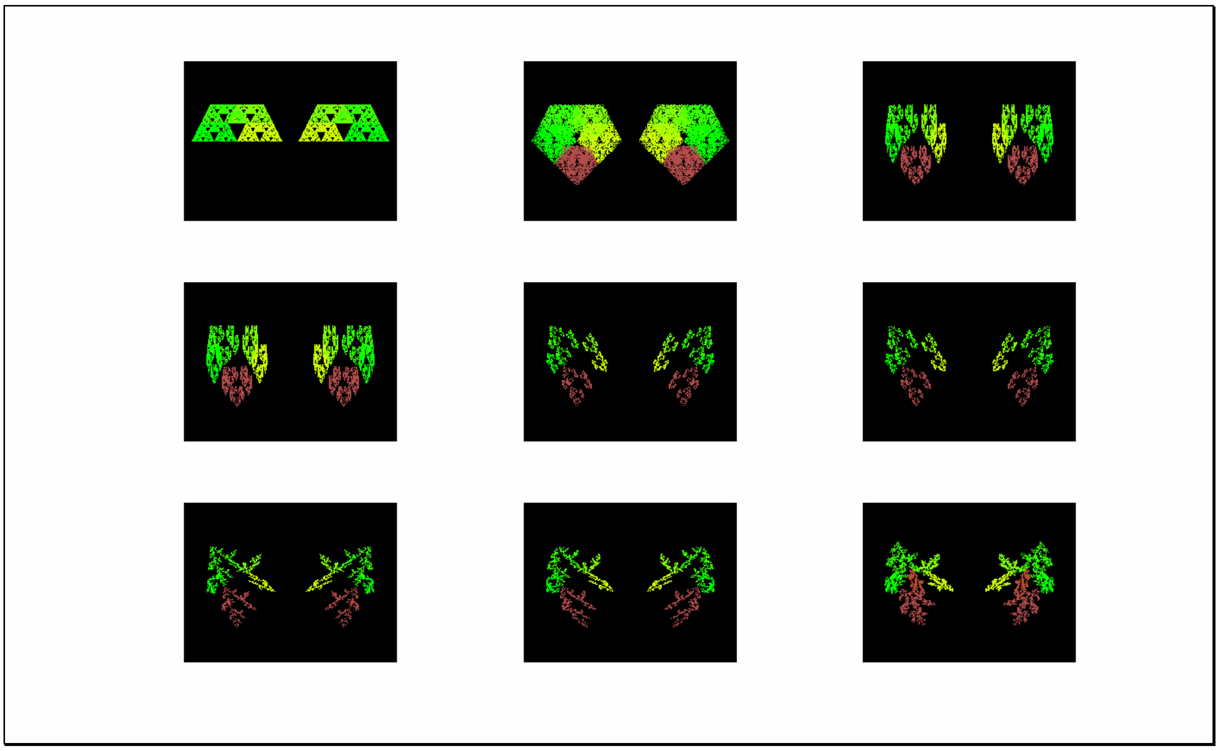


Fig. 14 Output results of calling of the script `fractpentagram.m` *with 9 implicitly defined calling argument list*

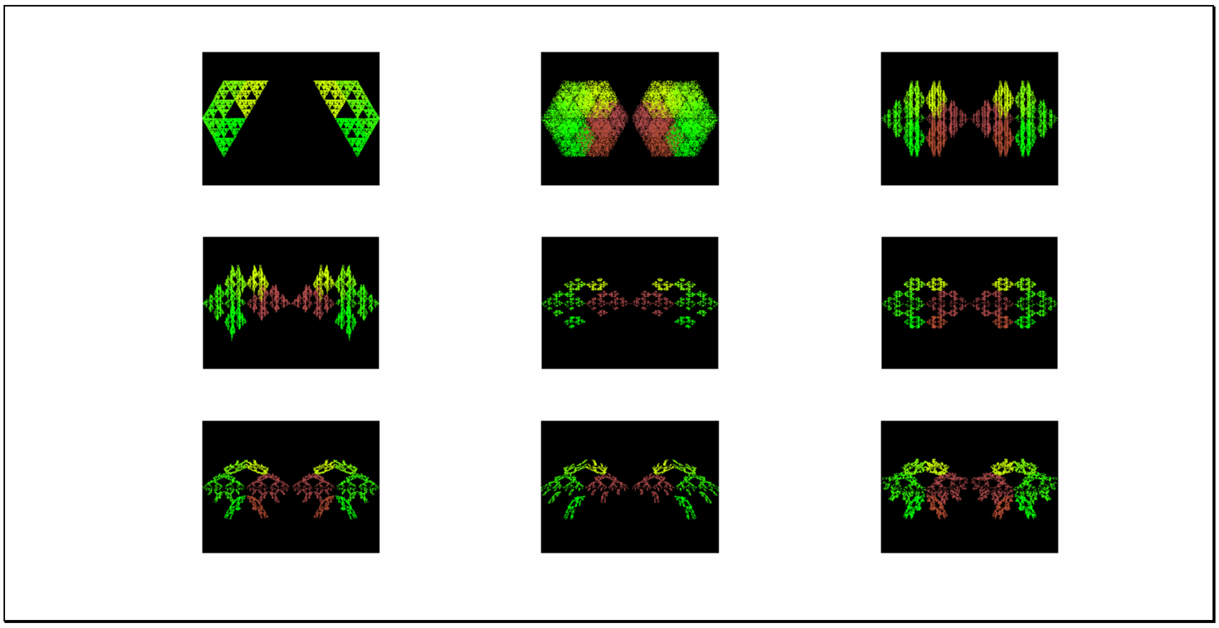


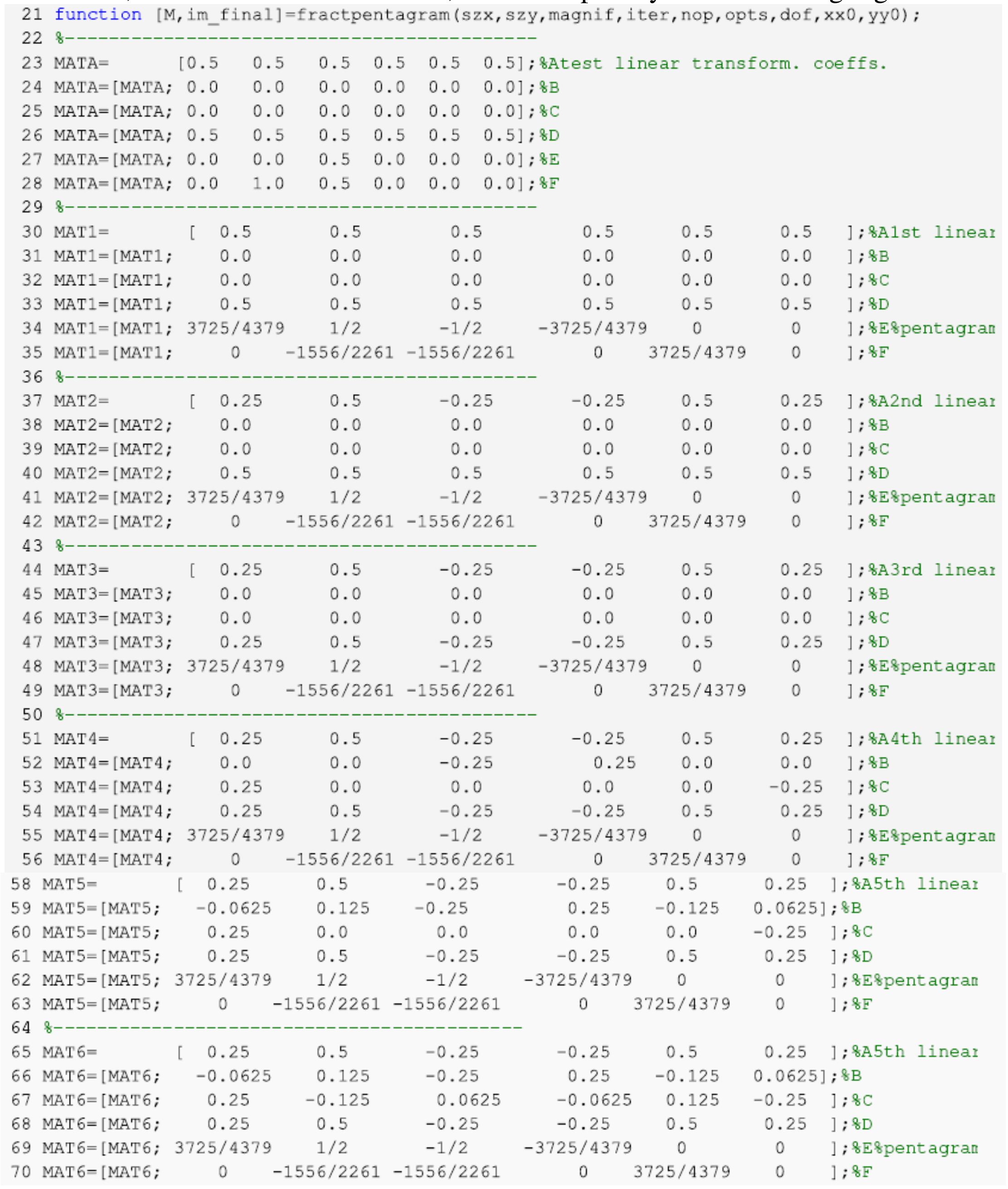
Fig. 15 Output results of calling of the script `fracthexagon.m` *with 9 implicitly defined calling arugment list*

For implicitly taken 1st and 2nd set of calling argument lists, in case of the two above scripts, the leading *motif*, known from construction of the famous Sierpinsky triangle has been spanned into one half/whole outline of pentagram and hexagon, respectively. For the implicitly defined 8th and 9th set of calling argument lists, there do occur the most intriguing and simultaneously complicated patterns or the so-called *leading motifs*, which however both of them do preserve some fractal dimension, for these two created fractal objects, respectively.

Finally, it is clearly shown, that with decreasing number of degrees of freedom (from 6 to 5) in script *fractpentagram.m*, the created fractal objects are slightly better in their esthetics and overall performance, than in the case of applications of 6 degrees of freedom, as it was in case of the script *fracthexagon.m*.

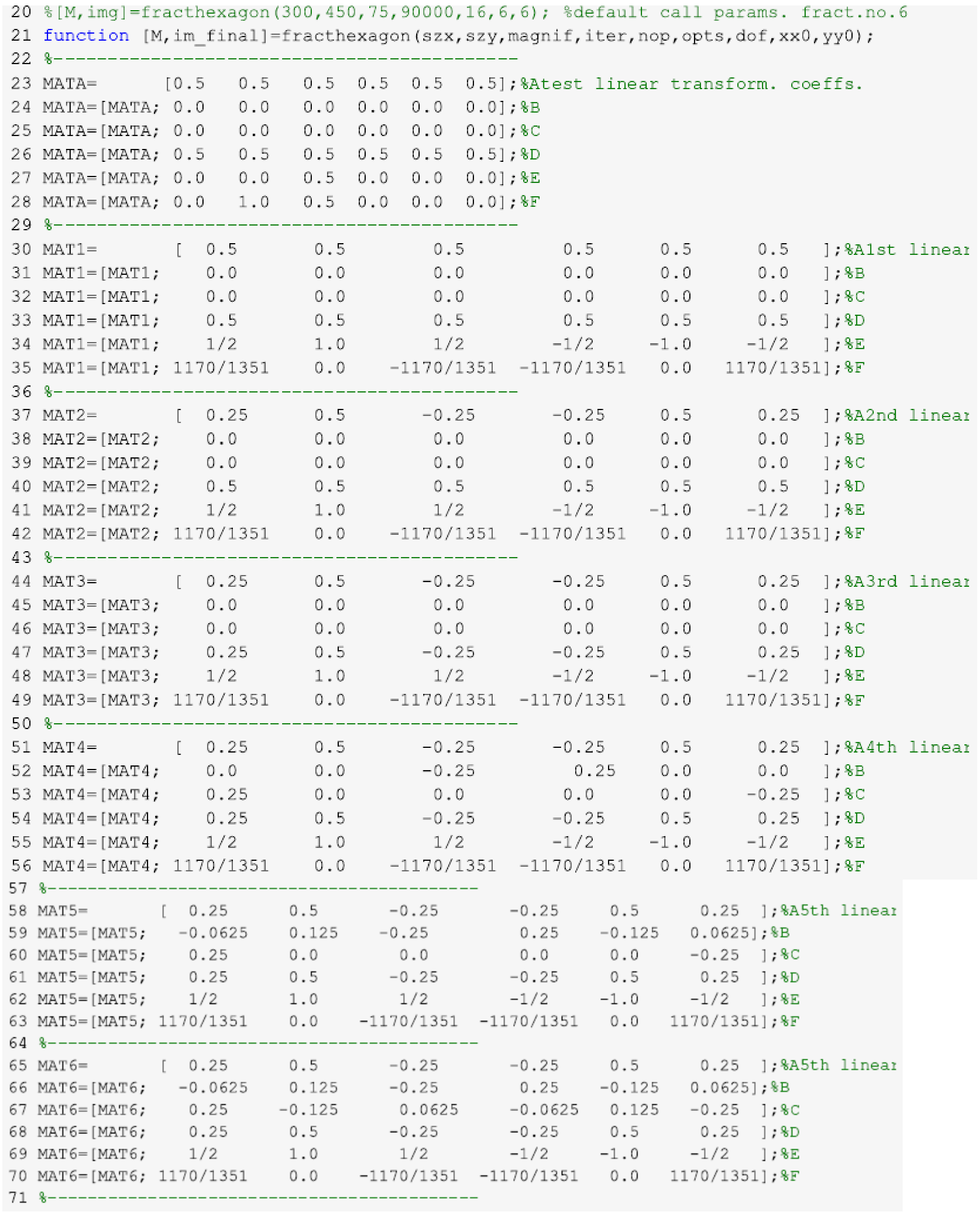
In case of increased number of degree of freedom (above 6), there is also an increased chance for producing rather random garbage-like-figure, than fractal-like-object. And this last one, here mentioned, should be characterized by occurrence of salient features of strong self-similarities of the leading *motifs*, accordingly to the whole created fractal object.

Here below only parts or some relevant scratches of the whole contents of the two mentioned contents have been presented. And these two scripts, here newly introduced in part 6B of this lecture, are basically the repetitions of the scripts' code from the former part of this lectures. In fact, the most important here are the headers, including matrices of transformations, with values of coefficients, as well implicitly defined calling argument lists:



```
function [M,im_final]=fractpentagram(szx,szy,magnif,iter,nop,opts,dof,xx0,yy0);
%-------------------------------------------
MATA=      [0.5    0.5    0.5   0.5   0.5   0.5];%Atest linear transform. coeffs.
MATA=[MATA; 0.0    0.0    0.0   0.0   0.0   0.0];%B
MATA=[MATA; 0.0    0.0    0.0   0.0   0.0   0.0];%C
MATA=[MATA; 0.5    0.5    0.5   0.5   0.5   0.5];%D
MATA=[MATA; 0.0    0.0    0.5   0.0   0.0   0.0];%E
MATA=[MATA; 0.0    1.0    0.5   0.0   0.0   0.0];%F
%-------------------------------------------
MAT1=       [  0.5        0.5         0.5          0.5       0.5       0.5    ];%A1st linear
MAT1=[MAT1;    0.0        0.0         0.0          0.0       0.0       0.0    ];%B
MAT1=[MAT1;    0.0        0.0         0.0          0.0       0.0       0.0    ];%C
MAT1=[MAT1;    0.5        0.5         0.5          0.5       0.5       0.5    ];%D
MAT1=[MAT1; 3725/4379     1/2        -1/2      -3725/4379     0         0     ];%E%pentagram
MAT1=[MAT1;      0     -1556/2261 -1556/2261       0     3725/4379     0     ];%F
%-------------------------------------------
MAT2=       [  0.25       0.5        -0.25       -0.25       0.5       0.25   ];%A2nd linear
MAT2=[MAT2;    0.0        0.0         0.0          0.0       0.0       0.0    ];%B
MAT2=[MAT2;    0.0        0.0         0.0          0.0       0.0       0.0    ];%C
MAT2=[MAT2;    0.5        0.5         0.5          0.5       0.5       0.5    ];%D
MAT2=[MAT2; 3725/4379     1/2        -1/2      -3725/4379     0         0     ];%E%pentagram
MAT2=[MAT2;      0     -1556/2261 -1556/2261       0     3725/4379     0     ];%F
%-------------------------------------------
MAT3=       [  0.25       0.5        -0.25       -0.25       0.5       0.25   ];%A3rd linear
MAT3=[MAT3;    0.0        0.0         0.0          0.0       0.0       0.0    ];%B
MAT3=[MAT3;    0.0        0.0         0.0          0.0       0.0       0.0    ];%C
MAT3=[MAT3;    0.25       0.5        -0.25       -0.25       0.5       0.25   ];%D
MAT3=[MAT3; 3725/4379     1/2        -1/2      -3725/4379     0         0     ];%E%pentagram
MAT3=[MAT3;      0     -1556/2261 -1556/2261       0     3725/4379     0     ];%F
%-------------------------------------------
MAT4=       [  0.25       0.5        -0.25       -0.25       0.5       0.25   ];%A4th linear
MAT4=[MAT4;    0.0        0.0        -0.25         0.25      0.0       0.0    ];%B
MAT4=[MAT4;    0.25       0.0         0.0          0.0       0.0      -0.25   ];%C
MAT4=[MAT4;    0.25       0.5        -0.25       -0.25       0.5       0.25   ];%D
MAT4=[MAT4; 3725/4379     1/2        -1/2      -3725/4379     0         0     ];%E%pentagram
MAT4=[MAT4;      0     -1556/2261 -1556/2261       0     3725/4379     0     ];%F
MAT5=       [  0.25       0.5        -0.25       -0.25       0.5       0.25   ];%A5th linear
MAT5=[MAT5;   -0.0625     0.125      -0.25         0.25     -0.125    0.0625];%B
MAT5=[MAT5;    0.25       0.0         0.0          0.0       0.0      -0.25   ];%C
MAT5=[MAT5;    0.25       0.5        -0.25       -0.25       0.5       0.25   ];%D
MAT5=[MAT5; 3725/4379     1/2        -1/2      -3725/4379     0         0     ];%E%pentagram
MAT5=[MAT5;      0     -1556/2261 -1556/2261       0     3725/4379     0     ];%F
%-------------------------------------------
MAT6=       [  0.25       0.5        -0.25       -0.25       0.5       0.25   ];%A5th linear
MAT6=[MAT6;   -0.0625     0.125      -0.25         0.25     -0.125    0.0625];%B
MAT6=[MAT6;    0.25      -0.125       0.0625     -0.0625     0.125    -0.25   ];%C
MAT6=[MAT6;    0.25       0.5        -0.25       -0.25       0.5       0.25   ];%D
MAT6=[MAT6; 3725/4379     1/2        -1/2      -3725/4379     0         0     ];%E%pentagram
MAT6=[MAT6;      0     -1556/2261 -1556/2261       0     3725/4379     0     ];%F
```

Fig. 16 Values of matrices MATn for code script fractpentagram.m



```
%[M,img]=fracthexagon(300,450,75,90000,16,6,6); %default call params. fract.no.6
function [M,im_final]=fracthexagon(szx,szy,magnif,iter,nop,opts,dof,xx0,yy0);
%-------------------------------------------
MATA=       [0.5   0.5   0.5  0.5  0.5  0.5];%Atest linear transform. coeffs.
MATA=[MATA; 0.0   0.0   0.0  0.0  0.0  0.0];%B
MATA=[MATA; 0.0   0.0   0.0  0.0  0.0  0.0];%C
MATA=[MATA; 0.5   0.5   0.5  0.5  0.5  0.5];%D
MATA=[MATA; 0.0   0.0   0.5  0.0  0.0  0.0];%E
MATA=[MATA; 0.0   1.0   0.5  0.0  0.0  0.0];%F
%-------------------------------------------
MAT1=        [  0.5        0.5         0.5         0.5       0.5       0.5   ];%A1st linear
MAT1=[MAT1;     0.0        0.0         0.0         0.0       0.0       0.0   ];%B
MAT1=[MAT1;     0.0        0.0         0.0         0.0       0.0       0.0   ];%C
MAT1=[MAT1;     0.5        0.5         0.5         0.5       0.5       0.5   ];%D
MAT1=[MAT1;     1/2        1.0         1/2        -1/2      -1.0      -1/2   ];%E
MAT1=[MAT1; 1170/1351      0.0    -1170/1351  -1170/1351    0.0   1170/1351];%F
%-------------------------------------------
MAT2=        [  0.25       0.5        -0.25       -0.25      0.5       0.25  ];%A2nd linear
MAT2=[MAT2;     0.0        0.0         0.0         0.0       0.0       0.0   ];%B
MAT2=[MAT2;     0.0        0.0         0.0         0.0       0.0       0.0   ];%C
MAT2=[MAT2;     0.5        0.5         0.5         0.5       0.5       0.5   ];%D
MAT2=[MAT2;     1/2        1.0         1/2        -1/2      -1.0      -1/2   ];%E
MAT2=[MAT2; 1170/1351      0.0    -1170/1351  -1170/1351    0.0   1170/1351];%F
%-------------------------------------------
MAT3=        [  0.25       0.5        -0.25       -0.25      0.5       0.25  ];%A3rd linear
MAT3=[MAT3;     0.0        0.0         0.0         0.0       0.0       0.0   ];%B
MAT3=[MAT3;     0.0        0.0         0.0         0.0       0.0       0.0   ];%C
MAT3=[MAT3;     0.25       0.5        -0.25       -0.25      0.5       0.25  ];%D
MAT3=[MAT3;     1/2        1.0         1/2        -1/2      -1.0      -1/2   ];%E
MAT3=[MAT3; 1170/1351      0.0    -1170/1351  -1170/1351    0.0   1170/1351];%F
%-------------------------------------------
MAT4=        [  0.25       0.5        -0.25       -0.25      0.5       0.25  ];%A4th linear
MAT4=[MAT4;     0.0        0.0        -0.25        0.25      0.0       0.0   ];%B
MAT4=[MAT4;     0.25       0.0         0.0         0.0       0.0      -0.25  ];%C
MAT4=[MAT4;     0.25       0.5        -0.25       -0.25      0.5       0.25  ];%D
MAT4=[MAT4;     1/2        1.0         1/2        -1/2      -1.0      -1/2   ];%E
MAT4=[MAT4; 1170/1351      0.0    -1170/1351  -1170/1351    0.0   1170/1351];%F
%-------------------------------------------
MAT5=        [  0.25       0.5        -0.25       -0.25      0.5       0.25  ];%A5th linear
MAT5=[MAT5;    -0.0625     0.125      -0.25        0.25     -0.125     0.0625];%B
MAT5=[MAT5;     0.25       0.0         0.0         0.0       0.0      -0.25  ];%C
MAT5=[MAT5;     0.25       0.5        -0.25       -0.25      0.5       0.25  ];%D
MAT5=[MAT5;     1/2        1.0         1/2        -1/2      -1.0      -1/2   ];%E
MAT5=[MAT5; 1170/1351      0.0    -1170/1351  -1170/1351    0.0   1170/1351];%F
%-------------------------------------------
MAT6=        [  0.25       0.5        -0.25       -0.25      0.5       0.25  ];%A5th linear
MAT6=[MAT6;    -0.0625     0.125      -0.25        0.25     -0.125     0.0625];%B
MAT6=[MAT6;     0.25      -0.125       0.0625     -0.0625    0.125    -0.25  ];%C
MAT6=[MAT6;     0.25       0.5        -0.25       -0.25      0.5       0.25  ];%D
MAT6=[MAT6;     1/2        1.0         1/2        -1/2      -1.0      -1/2   ];%E
MAT6=[MAT6; 1170/1351      0.0    -1170/1351  -1170/1351    0.0   1170/1351];%F
%-------------------------------------------
```

Fig. 17 Values of matrices MATn *for code script* `fracthexagon.m`

Last note: it has been decided to publish two last m-scripts, as well. It means: the most advanced mscripts so far presented within these two lectures, used for fractal generations. And they are presented below as an html-published-versions, embedded in MS Word document, while other parts of mscript codes, stayed within older version of these two lectures (V and VAB), were presented merely as bitmaps.

```
%Artur Bernat all rights reserved
%13 May 2006, triangle's fractal with stochastic transforms.
%[M,im_final]=fractpentagram(szx,szy,magnif,iter,nop,opts,DoF,x0,y0);
%szy,szx <=dimensions of 2D map(output size doubled in x)
%magnif  <=magnification coeff. for the figure
%iter    <=number of iteraton,
%nop     <=number of intervals in getting of frames
%opts    <=number of matrix of linear transformation coefficients
%DoF     <=DoF in choosing of coefficients in random walks
%x0,y0   <=coords. starting points,default: (0,0) middle of the screen
%[M,img]=fractpentagram(300,450,180,90000,16,0,3,-0.5,-1); %DEFAULT TRIANGLE NO.0
%[M,img]=fractpentagram(300,450,75,90000,16,1,4,-0.0,-0.0);%default params.FRACT.no.1A
%[M,img]=fractpentagram(300,450,75,90000,16,1,5,-0.0,-0.0);%default params.FRACT.no.1B
%[M,img]=fractpentagram(300,450,75,90000,16,2,5,-0.0,-0.0);%default params.FRACT.no.2A
%[M,img]=fractpentagram(300,450,75,90000,16,2,5,-0.0,-0.0);%default params.FRACT.no.2B
%[M,img]=fractpentagram(300,450,75,90000,16,3,5,-0.0,-0.0);%default params.FRACT.no.3A
%[M,img]=fractpentagram(300,450,75,90000,16,3,5,-0.0,-0.0);%default params.FRACT.no.3B
%[M,img]=fractpentagram(300,450,75,90000,16,4,5,-0.0,-0.0);%def. call prms.FRACT.no.4
%[M,img]=fractpentagram(300,450,75,90000,16,5,5,-0.0,-0.0);%def. call prms.FRACT.no.5
%[M,img]=fractpentagram(300,450,75,90000,16,6,6); %default call params. fract.no.6
function [M,im_final]=fractpentagram(szx,szy,magnif,iter,nop,opts,dof,xx0,yy0);
%--------------------------------------------
MATA=      [0.5   0.5   0.5  0.5  0.5  0.5];%Atest linear transform. coeffs.
MATA=[MATA; 0.0   0.0   0.0  0.0  0.0  0.0];%B
MATA=[MATA; 0.0   0.0   0.0  0.0  0.0  0.0];%C
MATA=[MATA; 0.5   0.5   0.5  0.5  0.5  0.5];%D
MATA=[MATA; 0.0   0.0   0.5  0.0  0.0  0.0];%E
MATA=[MATA; 0.0   1.0   0.5  0.0  0.0  0.0];%F
%--------------------------------------------
MAT1=       [  0.5       0.5        0.5         0.5      0.5      0.5   ];%A1st linear
transform. coeffs.
MAT1=[MAT1;    0.0       0.0        0.0         0.0      0.0      0.0   ];%B
MAT1=[MAT1;    0.0       0.0        0.0         0.0      0.0      0.0   ];%C
MAT1=[MAT1;    0.5       0.5        0.5         0.5      0.5      0.5   ];%D
MAT1=[MAT1; 3725/4379    1/2       -1/2     -3725/4379    0        0    ];%E%pentagram's
corners
MAT1=[MAT1;     0    -1556/2261 -1556/2261       0    3725/4379    0    ];%F
%--------------------------------------------
MAT2=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A2nd linear
transform. coeffs.
MAT2=[MAT2;    0.0       0.0        0.0         0.0      0.0      0.0   ];%B
MAT2=[MAT2;    0.0       0.0        0.0         0.0      0.0      0.0   ];%C
MAT2=[MAT2;    0.5       0.5        0.5         0.5      0.5      0.5   ];%D
MAT2=[MAT2; 3725/4379    1/2       -1/2     -3725/4379    0        0    ];%E%pentagram's
corners
MAT2=[MAT2;     0    -1556/2261 -1556/2261       0    3725/4379    0    ];%F
%--------------------------------------------
MAT3=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A3rd linear
transform. coeffs.
MAT3=[MAT3;    0.0       0.0        0.0         0.0      0.0      0.0   ];%B
MAT3=[MAT3;    0.0       0.0        0.0         0.0      0.0      0.0   ];%C
MAT3=[MAT3;    0.25      0.5       -0.25       -0.25     0.5      0.25  ];%D
MAT3=[MAT3; 3725/4379    1/2       -1/2     -3725/4379    0        0    ];%E%pentagram's
corners
MAT3=[MAT3;     0    -1556/2261 -1556/2261       0    3725/4379    0    ];%F
%--------------------------------------------
MAT4=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A4th linear
transform. coeffs.
MAT4=[MAT4;    0.0       0.0       -0.25         0.25    0.0      0.0   ];%B
MAT4=[MAT4;    0.25      0.0        0.0         0.0      0.0     -0.25  ];%C
MAT4=[MAT4;    0.25      0.5       -0.25       -0.25     0.5      0.25  ];%D
MAT4=[MAT4; 3725/4379    1/2       -1/2     -3725/4379    0        0    ];%E%pentagram's
corners
MAT4=[MAT4;     0    -1556/2261 -1556/2261       0    3725/4379    0    ];%F
%--------------------------------------------
MAT5=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A5th linear
transform. coeffs.
MAT5=[MAT5;   -0.0625    0.125    -0.25         0.25    -0.125   0.0625];%B
MAT5=[MAT5;    0.25      0.0        0.0         0.0      0.0     -0.25  ];%C
MAT5=[MAT5;    0.25      0.5       -0.25       -0.25     0.5      0.25  ];%D
MAT5=[MAT5; 3725/4379    1/2       -1/2     -3725/4379    0        0    ];%E%pentagram's
corners
MAT5=[MAT5;     0    -1556/2261 -1556/2261       0    3725/4379    0    ];%F
%--------------------------------------------
MAT6=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A5th linear
transform. coeffs.
MAT6=[MAT6;   -0.0625    0.125     -0.25        0.25    -0.125   0.0625];%B
MAT6=[MAT6;    0.25     -0.125      0.0625     -0.0625   0.125   -0.25  ];%C
MAT6=[MAT6;    0.25      0.5       -0.25       -0.25     0.5      0.25  ];%D
MAT6=[MAT6; 3725/4379    1/2       -1/2     -3725/4379    0        0    ];%E%pentagram's
corners
```

```
MAT6=[MAT6;     0    -1556/2261 -1556/2261       0    3725/4379    0    ];%F
%---------------------------------------------
switch opts
    case 0
        MAT=MATA;opts
    case 1
        MAT=MAT1;opts
    case 2
        MAT=MAT2;opts
    case 3
        MAT=MAT3;opts
    case 4
        MAT=MAT4;opts
    case 5
        MAT=MAT5;opts
    case 6
        MAT=MAT6;xx0=0;yy0=0;opts
end;
step=iter/nop;
MAX_TRANSF=dof;
mapsA=zeros(szy,szx);mapsB=zeros(szy,szx);
x=0;y=0;
nrp = ceil(MAX_TRANSF.*rand(1,iter));
[min(nrp(:)) max(nrp(:))]
%background;  1-8 palette's color  ;
paltsA=[0.0 0.0 0.0; 0.0 1.0 0.0;0.2 1.0 0.0; 0.5 1.0 0.0;...
                     0.8 1.0 0.0;0.7 0.3 0.3; 0.7 0.3 0.2;...
                     0.7 0.2 0.0;1.0 0.5 0.0; 0.8 1.0 1.0];
M=moviein(nop+2);
axis manual;
j=0;%control counter for catching of frames
for i=1:iter,
    kolor=uint8(nrp(i));
    xl=MAT(1,kolor).*x+MAT(2,kolor).*y+MAT(5,kolor);
    y =MAT(3,kolor).*x+MAT(4,kolor).*y+MAT(6,kolor);
    x=xl;
    Y=(szy/2+magnif.*(yy0+y));
    X=(szx/2+magnif.*(xx0+x));
    mapsA(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(1+nrp(i));
    mapsB(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(1+nrp(i));
    if mod(i,step)==0
     imshow([fliplr(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
     axis([0 szx + szx 0 szy]); axis manual; j=j+1; [i j]
     M(:,j)=getframe;
 end;
end;
im_final=ind2rgb([fliplr(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
movie(M);
RandName=100+899*rand(1,1);
movie2avi(M,['triangle2hex' int2str(RandName) '.avi'],'compression',...
         'Cinepak','FPS',4);
%close;
```

Fig. 18 Codescript for `fractpentagram.m`

```
%Artur Bernat all rights reserved
%12 May 2006, triangle's fractal with stochastic transforms.
%[M,im_final]=fracthexagon(szx,szy,magnif,iter,nop,opts,DoF,x0,y0);
%szy,szx <=dimensions of 2D map(output size doubled in x)
%magnif  <=magnification coeff. for the figure
%iter    <=number of iteraton,
%nop     <=number of intervals in getting of frames
%opts    <=number of matrix of linear transformation coefficients
%DoF     <=DoF in choosing of coefficients in random walks
%x0,y0   <=coords. starting points,default: (0,0) middle of the screen
%[M,img]=fracthexagon(300,450,180,90000,16,0,3,-0.5,-1); %DEFAULT TRIANGLE NO.0
%[M,img]=fracthexagon(300,450,75,90000,16,1,4,-0.0,-0.0);%default params.FRACT.no.1A
%[M,img]=fracthexagon(300,450,75,90000,16,1,6,-0.0,-0.0);%default params.FRACT.no.1B
%[M,img]=fracthexagon(300,450,75,90000,16,2,5,-0.0,-0.0);%default params.FRACT.no.2A
%[M,img]=fracthexagon(300,450,75,90000,16,2,6,-0.0,-0.0);%default params.FRACT.no.2B
%[M,img]=fracthexagon(300,450,75,90000,16,3,5,-0.0,-0.0);%default params.FRACT.no.3A
%[M,img]=fracthexagon(300,450,75,90000,16,3,6,-0.0,-0.0);%default params.FRACT.no.3B
%[M,img]=fracthexagon(300,450,75,90000,16,4,6,-0.0,-0.0);%def. call prms.FRACT.no.4
%[M,img]=fracthexagon(300,450,75,90000,16,5,5,-0.0,-0.0);%def. call prms.FRACT.no.5
%[M,img]=fracthexagon(300,450,75,90000,16,6,6); %default call params. fract.no.6
function [M,im_final]=fracthexagon(szx,szy,magnif,iter,nop,opts,dof,xx0,yy0);
%---------------------------------------------
MATA=      [0.5   0.5   0.5  0.5  0.5  0.5];%Atest linear transform. coeffs.
MATA=[MATA; 0.0   0.0   0.0  0.0  0.0  0.0];%B
MATA=[MATA; 0.0   0.0   0.0  0.0  0.0  0.0];%C
MATA=[MATA; 0.5   0.5   0.5  0.5  0.5  0.5];%D
MATA=[MATA; 0.0   0.0   0.5  0.0  0.0  0.0];%E
MATA=[MATA; 0.0   1.0   0.5  0.0  0.0  0.0];%F
%---------------------------------------------
MAT1=       [  0.5       0.5        0.5         0.5      0.5      0.5   ];%A1st linear transform.
coeffs.
MAT1=[MAT1;    0.0       0.0        0.0         0.0      0.0      0.0   ];%B
MAT1=[MAT1;    0.0       0.0        0.0         0.0      0.0      0.0   ];%C
MAT1=[MAT1;    0.5       0.5        0.5         0.5      0.5      0.5   ];%D
MAT1=[MAT1;    1/2       1.0        1/2        -1/2     -1.0     -1/2   ];%E
MAT1=[MAT1; 1170/1351    0.0    -1170/1351  -1170/1351   0.0   1170/1351];%F
%---------------------------------------------
MAT2=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A2nd linear transform.
coeffs.
MAT2=[MAT2;    0.0       0.0        0.0         0.0      0.0      0.0   ];%B
MAT2=[MAT2;    0.0       0.0        0.0         0.0      0.0      0.0   ];%C
MAT2=[MAT2;    0.5       0.5        0.5         0.5      0.5      0.5   ];%D
MAT2=[MAT2;    1/2       1.0        1/2        -1/2     -1.0     -1/2   ];%E
MAT2=[MAT2; 1170/1351    0.0    -1170/1351  -1170/1351   0.0   1170/1351];%F
%---------------------------------------------
MAT3=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A3rd linear transform.
coeffs.
MAT3=[MAT3;    0.0       0.0        0.0         0.0      0.0      0.0   ];%B
MAT3=[MAT3;    0.0       0.0        0.0         0.0      0.0      0.0   ];%C
MAT3=[MAT3;    0.25      0.5       -0.25       -0.25     0.5      0.25  ];%D
MAT3=[MAT3;    1/2       1.0        1/2        -1/2     -1.0     -1/2   ];%E
MAT3=[MAT3; 1170/1351    0.0    -1170/1351  -1170/1351   0.0   1170/1351];%F
%---------------------------------------------
MAT4=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A4th linear transform.
coeffs.
MAT4=[MAT4;    0.0       0.0       -0.25         0.25    0.0      0.0   ];%B
MAT4=[MAT4;    0.25      0.0        0.0         0.0      0.0     -0.25  ];%C
MAT4=[MAT4;    0.25      0.5       -0.25       -0.25     0.5      0.25  ];%D
MAT4=[MAT4;    1/2       1.0        1/2        -1/2     -1.0     -1/2   ];%E
MAT4=[MAT4; 1170/1351    0.0    -1170/1351  -1170/1351   0.0   1170/1351];%F
%---------------------------------------------
MAT5=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A5th linear transform.
coeffs.
MAT5=[MAT5;  -0.0625     0.125    -0.25         0.25    -0.125   0.0625];%B
MAT5=[MAT5;    0.25      0.0        0.0         0.0      0.0     -0.25  ];%C
MAT5=[MAT5;    0.25      0.5       -0.25       -0.25     0.5      0.25  ];%D
MAT5=[MAT5;    1/2       1.0        1/2        -1/2     -1.0     -1/2   ];%E
MAT5=[MAT5; 1170/1351    0.0    -1170/1351  -1170/1351   0.0   1170/1351];%F
%---------------------------------------------
MAT6=       [  0.25      0.5       -0.25       -0.25     0.5      0.25  ];%A5th linear transform.
coeffs.
MAT6=[MAT6;  -0.0625     0.125     -0.25        0.25    -0.125   0.0625];%B
MAT6=[MAT6;    0.25     -0.125      0.0625     -0.0625   0.125   -0.25  ];%C
MAT6=[MAT6;    0.25      0.5       -0.25       -0.25     0.5      0.25  ];%D
MAT6=[MAT6;    1/2       1.0        1/2        -1/2     -1.0     -1/2   ];%E
MAT6=[MAT6; 1170/1351    0.0    -1170/1351  -1170/1351   0.0   1170/1351];%F
%---------------------------------------------
switch opts
    case 0
        MAT=MATA;opts
    case 1
```

```
        MAT=MAT1;opts
    case 2
        MAT=MAT2;opts
    case 3
        MAT=MAT3;opts
    case 4
        MAT=MAT4;opts
    case 5
        MAT=MAT5;opts
    case 6
        MAT=MAT6;xx0=0;yy0=0;opts
end;
step=iter/nop;
MAX_TRANSF=dof;
mapsA=zeros(szy,szx);mapsB=zeros(szy,szx);
x=0;y=0;
nrp = ceil(MAX_TRANSF.*rand(1,iter));
[min(nrp(:)) max(nrp(:))]
%background;  1-8 palette's color  ;
paltsA=[0.0 0.0 0.0; 0.0 1.0 0.0;0.2 1.0 0.0; 0.5 1.0 0.0;...
                     0.8 1.0 0.0;0.7 0.3 0.3; 0.7 0.3 0.2;...
                     0.7 0.2 0.0;1.0 0.5 0.0; 0.8 1.0 1.0];
M=moviein(nop+2);
axis manual;
j=0;%control counter for catching of frames
for i=1:iter,
    kolor=uint8(nrp(i));
    xl=MAT(1,kolor).*x+MAT(2,kolor).*y+MAT(5,kolor);
    y =MAT(3,kolor).*x+MAT(4,kolor).*y+MAT(6,kolor);
    x=xl;
    Y=(szy/2+magnif.*(yy0+y));
    X=(szx/2+magnif.*(xx0+x));
    mapsA(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(1+nrp(i));
    mapsB(uint16(1+sign(Y).*Y),uint16(1+sign(X).*X))=uint8(1+nrp(i));
    if mod(i,step)==0
     imshow([fliplr(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
     axis([0 szx + szx 0 szy]); axis manual; j=j+1; [i j]
     M(:,j)=getframe;
 end;
end;
im_final=ind2rgb([fliplr(mapsA(1:szy,1:szx)) mapsB(1:szy,1:szx)],paltsA);
movie(M);
RandName=100+899*rand(1,1);
movie2avi(M,['triangle2hex' int2str(RandName) '.avi'],'compression',...
         'Cinepak','FPS',4);
%close;
```

Fig. 18 Codescript for `fracthexagon.m`