Notes to help using algorithms based on the symmetrical Recurrences concept

Jean-Marc Girault,

I. Introduction

This note aims to help reader to use algorithms dedicated to measure the level of symmetry present in a time series, in a couple of time series and in between 2 time series. These algorithms are developed with Matlab.

The proposed methods were inspired from several refrences: [Abdennaji (2021)], [Girault (2015)], [Marwan (2007)], [Marwan (2009)],[BenTal (2002)].

In the following, a recall on recurrence plot and symmetrical recurrence plot are proposed.

A. Recurrences concept

For a binary sequence, a m-motif is defined recurrent if it exists at least another m-motif that is similar to it, this m-motif can be the m-motif itself: it is an auto-recurrent m-motif. In other words for a binary sequence $\mathbf{x} = \{\mathbf{x(1)}, \mathbf{x(2)}, \ldots, \mathbf{x(L)}\}$ composed of L bits, with $1 \leq i \leq L - m + 1$ a m-motif $\mathbf{x_m(i)} = [\mathbf{x(i)}, ..., \mathbf{x(i + m - 1)}]$ is recurrent if it exists at least a second m-motif $\mathbf{x_m(j)}$ with $1 \leq j \leq L - m + 1$ that is similar or auto-similar. A recurrent matrice $\mathbf{M}$ can be calculated from each element i, j as followed :

$$M(i,j) = \Theta\left[\epsilon - \|\mathbf{x_m(i)} - \mathbf{x_m(j)}\|\right],$$

where $\Theta[\bullet]$is the Heaviside function, ϵ is a threshold/tolerance that will be set to zero for binary sequences and $\|\mathbf{x_m(i)} - \mathbf{x_m(j)}\|$ is a distance. When 2 m-motifs are similar then $M(i,j) = 1$ otherwise $M(i,j) = 0$.

The number of similar m-motifs is obtained here for $m > 1$ by calculating:

$$N_r = \sum_{m=2}^{m=L} n_r(m),$$

with $n_r(m) = n_{ar}(m) + n_{rec}(m)$ the number of recurrences, with $n_{ar}(m) = \sum_{j=1,j=i}^{L-m} M(i,j)$ the number of auto-recurrences and with $n_{rec}(m) = \frac{1}{2}\sum_{j=1,j\neq i}^{N-m} M(i,j)$ the number of recurrences without auto-recurrences.

Note that $n_{ar}(m)$ corresponds to the summation of the diagonal elements of $\mathbf{M}$ and that N_r corresponds to the summation of all element different from zero in the upper triangular matrix of $\mathbf{M}$. For instance for the binary sequence $\mathbf{x_1} = \{\square\blacksquare\blacksquare\blacksquare\}$ with $L = 4$ bits (see Fig.1), the total number of recurrences of 2-motifs is $n_{ar}(2) + n_r(2) = 3 + 1$, the double in the lower triangular matrix $\mathbf{M}$ was not considered here.

B. Symmetrical Recurrences

Recently in [Girault (2015)], new types of recurrences was proposed. These recurrences are symmetrical recurrences directly related to the 4 isometries necessary to generate them. Four recurrent matrices $\mathbf{M^{(k)}}$ with $k \in \{T, R, I, G\}$ can be calculated from each element i, j as followed:

$$M^{(k)}(i,j) = \Theta\left[\epsilon - \|\mathbf{x_m(i)} - \mathbf{\Gamma^{(k)}}\left[\mathbf{x_m(j)}\right]\|\right],$$

Corresponding Author: Jean-Marc Girault, ESEO GRoup; Email, jean-marc.girault@eseo.fr; Phone, +33 (0)6 27 84 05 02

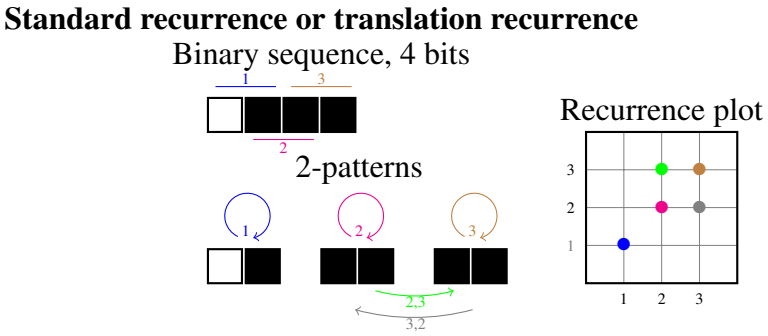

Fig. 1. Standard recurrence plot of 2-patterns for a binary sequence of 4 bits. The x-axis represents the position of each 2-patterns and the y-axis represents the position of the 2-patterns tested. For instance, the first 2-pattern □■ at position $x = 1$ is similar/auto-recurrrent with the pattern at position $y = 1$ and a blue dot is plotted at position $x = 1, y = 1$; the first 2-pattern being not similar to 2-patterns at positions $x = 2, 3$, none dot is plotted. This procedure is repeated till the end of the binary sequence. The total number of 2-recurrent patterns is $n_{ar}(2) + n_{rec}(2) = 3 + 1$ with 3 auto-recurrences (dots blue, magenta and brown in the main diagonal) and 1 recurrence (green dot in the upper triangle), the duplicate (bullet in gray) being not counted.

where $k \in \{T, R, I, G\}$ corresponds to the 4 types of isometries (T: Translation, R: Reflection, I: Inversion, G: Glide reflection), $\Theta[\bullet]$ is the Heaviside function, ϵ is a threshold set to zero for binary sequences, $\|\mathbf{x_m(i)} - \mathbf{\Gamma^{(k)}}\left[\mathbf{x_m(j)}\right]\|$ is a distance. When 2 m-motifs are similar then $M^{(k)} = 1$ otherwise $M^{(k)} = 0$.

The number of similar m-motifs is obtained here for $m > 1$ by calculating:

$$N_r^{(k)} = \sum_{m=2}^{m=L} n_r^{(k)}(m),$$

with $n_r^{(k)}(m) = n_{ar}^{(k)}(m) + n_{rec}^{(k)}(m)$ the number of recurrences, with $n_{ar}^{(k)}(m) = \sum_{j=1,j=i}^{L-m} M^{(k)}(i,j)$ the number of auto-recurrences and with $n_{rec}^{(k)}(m) = \frac{1}{2}\sum_{j=1,j\neq i}^{L-m} M^{(k)}(i,j)$ the number of recurrences without auto-recurrences. Note that $n_{ar}^{(k)}(m)$ corresponds to the summation of the diagonal elements of $\mathbf{M^{(k)}}$ and that $N_r^{(k)}$ corresponds to the summation of all element different from zero in the upper triangular matrix of $\mathbf{M^{(k)}}$.

Lot of other quantitative descriptors can also be calculated as the ones proposeb by [Marwan (2007)], [Marwan (2009)].

Let's consider the following binary sequence □■■■ and let's evaluate the different types of symmetrical recurrences:

- the symmetrical recurrent plot of mirror 2-motifs is represented in Fig.2. The first 2-motif □■ is not mirror auto-similar since the mirror 2-motif is ■□. The first 2-motif □■ is not similar to another 2-motif in the binary sequence. However, the second 2-motif ■■ and its mirror ■■ are auto-similar (magenta dot at position $2, 2$). Furthermore, the 2-motif ■■ is also similar to the third 2-motif (green dot at position $2, 3$). Finally, as the second 2-motif, the third 2-motif is mirror auto-recurrent (brown dot at position $3, 3$).The third 2-motif is also mirror recurrent with the second 2-motif (gray dot at position $3, 2$);
- the symmetrical recurrent plot of opposite 2-motifs is represented in Fig.3a. The first 2-motif □■ is opposite auto-similar since the opposite 2-motif obtained by inversion is □■. The first 2-motif □■ is not similar to another 2-motif in the binary sequence;
- the symmetrical recurrent plot of glide 2-motifs is represented in Fig.3b. The first 2-motif □■ is not glide auto-similar since the glide 2-motif obtained with glide reflection is ■□. The first 2-motif □■ is not glide similar to another 2-motif in the binary sequence.

Let's consider another binary sequence ■□■□■■□□ where symmetrical recurrent plots are represented in Fig.4. In this simulation 1-motifs and 2-motifs are considered. With 1-motifs, symmetrical recurrent plots are represented on the left. False recurrences are present. With 2-motifs, symmetrical recurrent plots

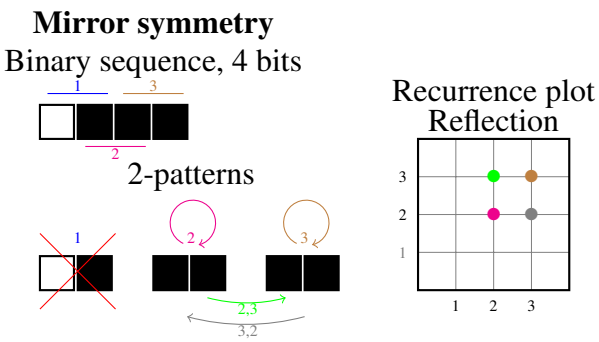

Fig. 2. Plot of mirror recurrences of 2-motif for a binary sequence of 4 bits. The total number of mirror 2-motif i.e. obtained witha vertical reflection is $n_{ar}^{(R)}(2) + n_r^{(R)}(2) = 2 + 1$ with 2 mirror auto-recurrences (magenta and brown bullets) and 1 recurrence (green bullet), the duplicate (bullet in gray) being not counted.

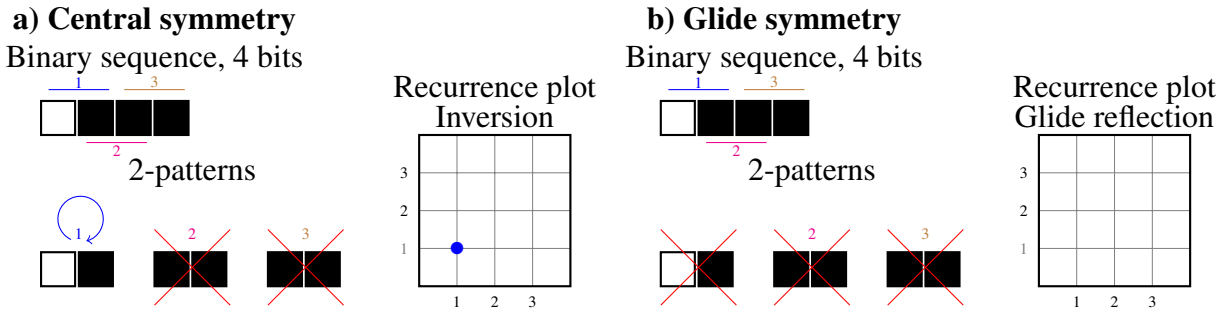

Fig. 3. Recurrence plots of 2-patterns with different type of symmetry for a binary sequence of 4 bits. The total number of 2-recurrent patterns with inversion is $n_{ar}^{(I)}(2) + n_r^{(I)}(2) = 1$ with 1 auto-recurrence (blue bullet). The total number of 2-recurrent patterns with glide reflection is $n_{ar}^{(G)}(2) + n_r^{(G)}(2) = 0$.

are represented on the right. False recurrences are removed. To redo this simulation, Matlab codes (Example1binSeq, SymPlotJMGi, rqamarwan) are reported in appendix.

By coding protein sequences in binary form, Symmetrical Recurrence Quantification Analysis can be advantageously used to classify different type of protein as reported in [Abdennaji (2021)].

II. Cross Symmetrical Recurrences

To evaluate distorsions, i.e. here losses or gains in symmetry level, when a reference signal is available, then the concept of symmetrical recurrences and its descriptors can be of great help.

Consider here a triangular signal. By nature this signal has the 4 types of symmetrical recurrences as indicated in Fig. 5 (top left). Many axes of symmetry and centers of inversion are present periodically. We see diagonal and anti-diagonal lines that appear in the recurrence graphs obtained with the 4 different types of isometries ('T', 'R', 'I', 'G'). Diagonals indicate the deterministic nature of the triangular signal, the spacing between diagonal lines indicates the period of the signal.

By playing with the duty cycle, we can destroy symmetries as shown in Fig. 5 (bottom left). Mirror or glide type of symmetries are more strongly reduced than symmetries resulting from translation and inversion.

We can also compare 2 signals by calculating the joint m-motifs. For example by considering 2 triangular signals shifted by ϕ degrees, we again observe diagonal and anti-diagonal lines (see Fig. 5 at the top right). The distance between diagonals indicates the period and the position of the diagonal lines with respect to the main diagonal indicating the delay between the 2 signals. By reducing the level of symmetry by adjusting the duty cycle, again we see diagonal segments in Fig. 5 (bottom right). In Fig. 6, 12 descriptors (SRQA) are shown as a function of the duty cycle. We see that certain descriptors resulting from the isometries 'R' and

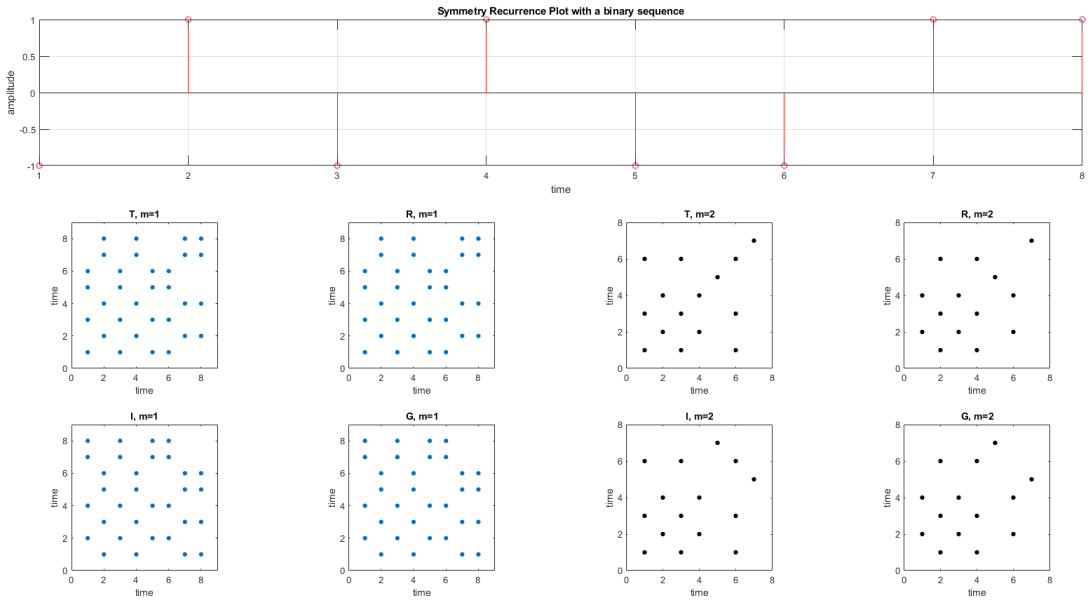

Fig. 4. Symmetrical recurrence plots obtained from a binary sequence. At the left, the 4 symmetry recurrence plots ('T', 'R', 'I', 'G') are obtained with m=1. At the right, the 4 symmetry recurrence plots ('T', 'R', 'I', 'G') are obtained with m=2, all false recurrences are removed.

'G' are very sensitive to the loss or to the gain of symmetry.

To redo the simulations, the Matlab codes (Example3DisymmetrySawtoothSignal.m, Example4SRQASawtoothSig are given in the appendix.

III. 2D-Phase Space Symmetrical Recurrences

The idea here is to be able to evaluate the loss or the gain of symmetry as a function of a change in the regime of the nonlinear dynamic system under studied. Here the Duffing system is studied. New symmetries appear when the control parameter exceeds 0.847.

To redo the simulations, the Matlab codes (PhaseSpaceSymPlot.m, duffing.m, Example2ChaosDuffing.m) are given in the appendix.

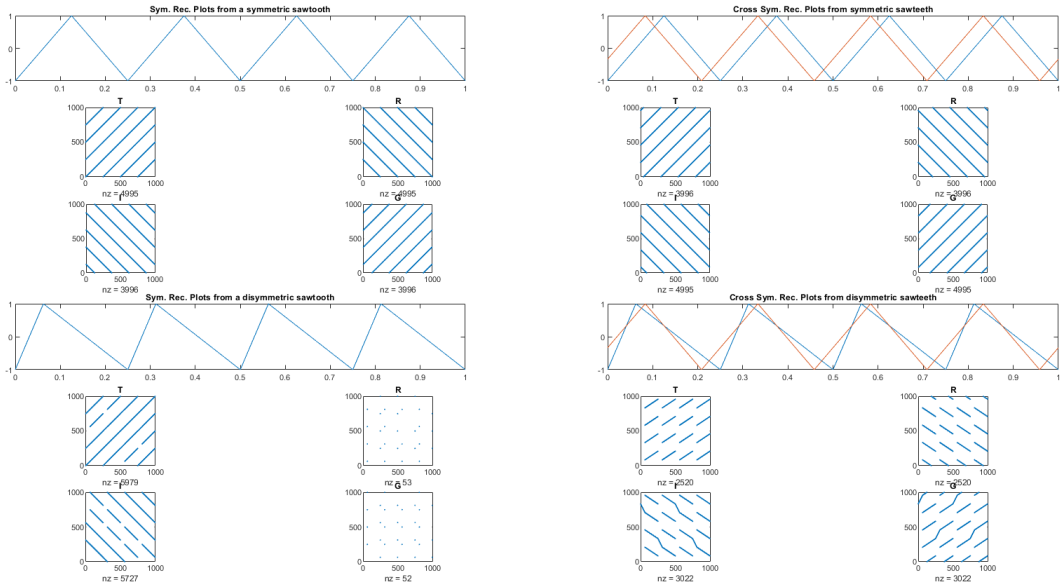

Fig. 5. Symmetrical Recurrence Plot (SRP) with distorted sawtooth signal. At the left top corner, a triangular signal is reported (sawtooth signal with a duty cycle equals to 0.5). The 4 SRP reported below show diagonal and anti diagonal lines. At the left down corner, a sawtooth signal with a duty cycle equals to 0.25 is reported. The 4 SRP reported below show not complete diagonal and anti diagonal lines (for 'T', 'I') and some dots (for 'R', 'G'). At the right top corner, triangular and shifted triangular signals are reported. The 4 SRP reported below show diagonal and anti diagonal lines. At the right down corner, two sawtooth signals with duties cycle equals to 0.5 and 0.25 are reported. The 4 SRP reported below show broken diagonal and anti diagonal lines (for 'T', 'I', 'R', 'G').

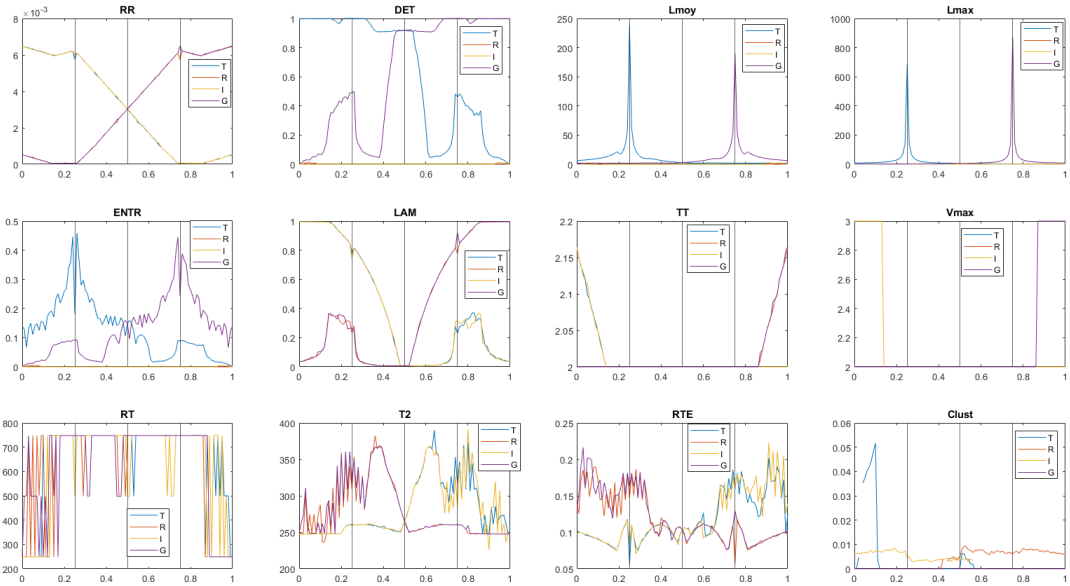

Fig. 6. Symmetrical Quantification Recurrence Analysis (SRQA). Twelve SQRA features as a function of the duty cycle ranging from 0 to 1 are reported. Most of features reveals the gain or the loss of symmetry.

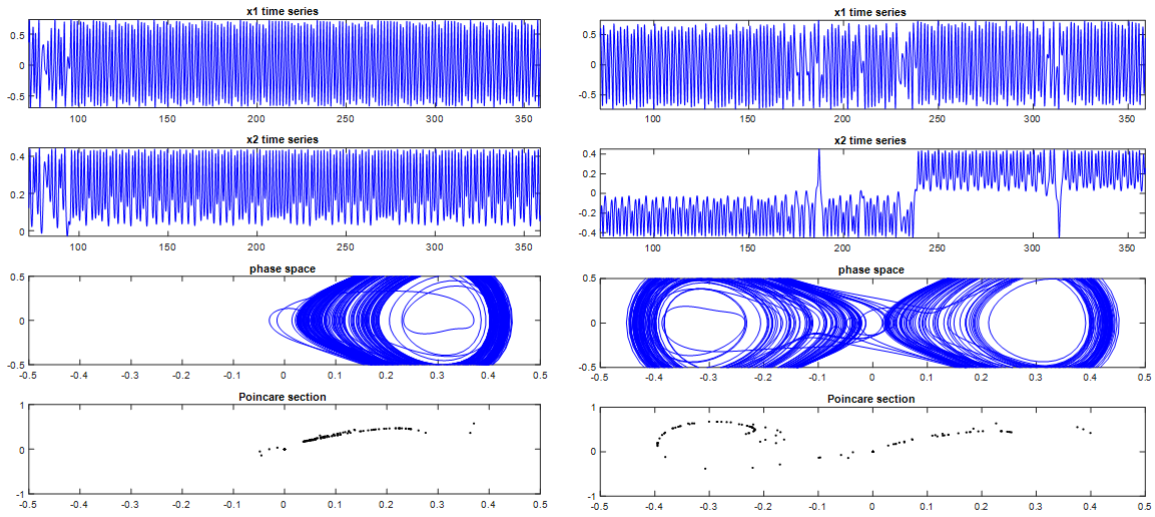

Fig. 7. Duffing time series, phase space and Poincare section. At the left, Duffing system simulated with mu=10, c=100, beta=1, a=0.848, omega=3.5. At the right, Duffing system simulated with mu=10, c=100, beta=1, a=0.850, omega=3.5. A double scroll appears with new symmetry.

IV. Appendix

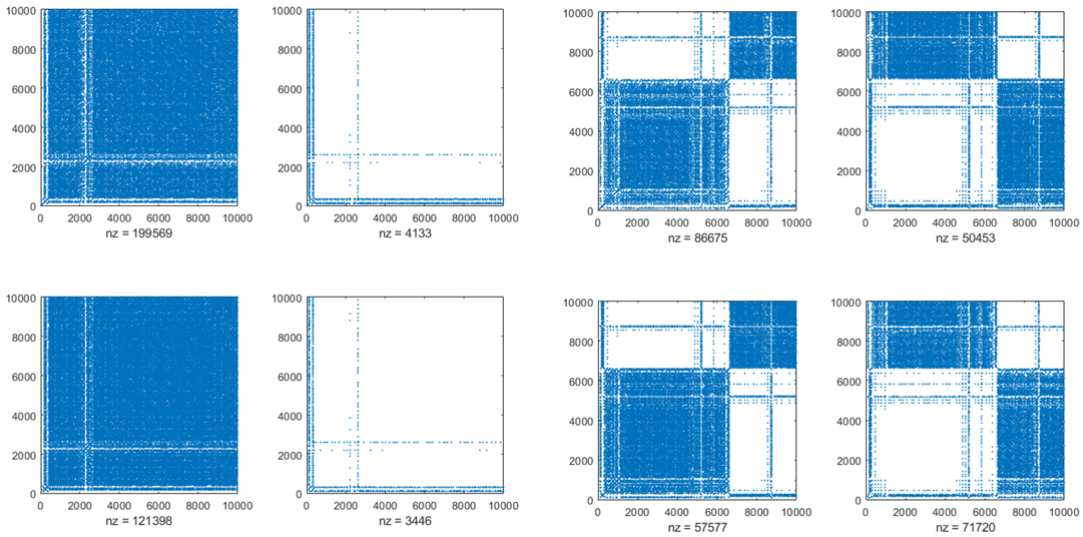

Fig. 8. Symmetrical Recurrence Plot of the Duffing time series. At the left, Symmetrical Recurrence Plot obtained with the Duffing system simulated with mu=10, c=100, beta=1, a=0.848, omega=3.5. At the right, Symmetrical Recurrence Plot obtained with the Duffing system simulated with mu=10, c=100, beta=1, a=0.850, omega=3.5. New patterns appear.

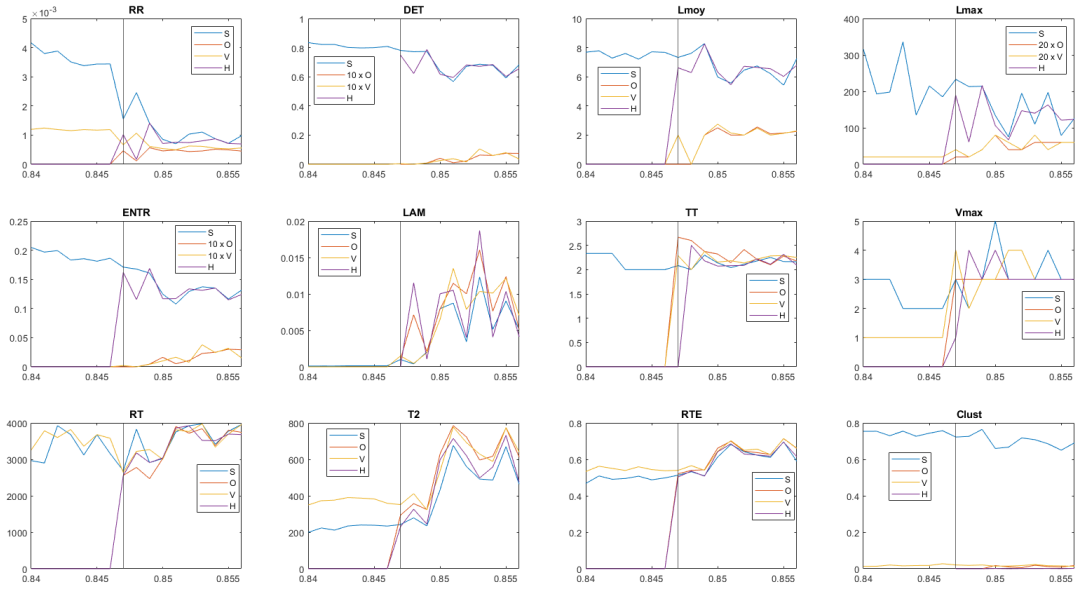

Fig. 9. Symmetrical Quantification Recurrence Analysis (SRQA) obtained with the Duffing system. Twelve SQRA features as a function of the control parameter ranging from 0.840 to 0.855 are reported. Most of features reveals the gain or the loss of symmetry.

Example1binSeq.m

```
% Symmetry plot calculated for a binary sequence
% with m ranging from 1 to 2.
% z: binary sequence ogf length L
% Isometry: 'T','R', 'I', 'G';
% m: Size of the pattern analyzed
% M: maximal value of m, M<=L
% Tolerance: norm(z)/20
% Centering: 0 no centering
% Display: 0 no display
clear;clc;close all;
z=[-1 1 -1 1 -1 -1 1 1];
% With m=1, we notice the presence of false recurrences
m=1;
[SPt1]=SymPlotJMGi(z,'T',m,norm(z)/20,0,0);
[SPr1]=SymPlotJMGi(z,'R',m,norm(z)/20,0,0);
[SPi1]=SymPlotJMGi(z,'I',m,norm(z)/20,0,0);
[SPg1]=SymPlotJMGi(z,'G',m,norm(z)/20,0,0);
%-
RQA_t1 = rqamarwan(SPt1)
RQA_r1 = rqamarwan(imrotate(SPr1,90))
RQA_i1 = rqamarwan(imrotate(SPi1,90))
RQA_g1 = rqamarwan(SPg1)
% With m=2, false recurrences are removed.
m=2;
%
[SPt2]=SymPlotJMGi(z,'T',m,norm(z)/20,0,0);
[SPr2]=SymPlotJMGi(z,'R',m,norm(z)/20,0,0);
[SPi2]=SymPlotJMGi(z,'I',m,norm(z)/20,0,0);
[SPg2]=SymPlotJMGi(z,'G',m,norm(z)/20,0,0);
%
RQA_t2 = rqamarwan(SPt2)
RQA_r2 = rqa_marwan(imrotate(SPr2,90))
RQA_i2 = rqa_marwan(imrotate(SPi2,90))
RQA_g2 = rqa_marwan(SPg2)
%
figure
subplot(311);stem(z,'ro-'); xlabel('time');
ylabel('amplitude');xlim([1 8]);grid
title('Symmetry Recurrence Plot with a binary sequence')
subplot(345);spy(SPt1);axis xy;
title('T, m=1');xlabel('time');ylabel('time')
subplot(346);spy(SPr1);axis xy;
title('R, m=1');xlabel('time');ylabel('time')
subplot(347);spy(SPt2,'k');axis xy;
title('T, m=2');xlabel('time');ylabel('time')
subplot(348);spy(SPr2,'k');axis xy;
title('R, m=2');xlabel('time');ylabel('time')
subplot(349);spy(SPi1);axis xy
```

```
;title('I, m=1');xlabel('time');ylabel('time')
subplot(3,4,10);spy(SPg1);axis xy;
title('G, m=1');xlabel('time');ylabel('time')
subplot(3,4,11);spy(SPi2,'k');axis xy;
title('I, m=2');xlabel('time');ylabel('time')
subplot(3,4,12);spy(SPg2,'k');axis xy;
title('G, m=2');xlabel('time');ylabel('time')
```

SymPlotJMGi.m

```
function [SP]=SymPlotJMGi(signal,type,m,r_factor,Centering,display)
% Input variables:
% signal – signal
% type : 1–>Translation, 2–> Reflection, 3–>Inversion, 4–> Glide reflection
% m : size of the pattern (uplet)
% r_factor – tolerance   r_factor*std(signal)
%r_factor between 0.1*SD and 0.25*SD,
% where SD is the signal standard deviation
% Centering : Centering=1 leads to the
% centering of the pattern, otherwise nothing
% Output variables:
% SP : Symetry plot
% Example of display: spy(SP);axis xy;
% clear,clc
% r_factor=0.1;
% signal=[0 1 2 3 2 1 0 1 2 3 2 1 0 1 2 3 2 1];
%m=2
%

X=[];
N=length(signal);
    for k=1:N–m+1
       if Centering==0; Y(k,:)=signal(k:k+m–1);
       else  Y(k,:)=signal(k:k+m–1)–mean(signal(k:k+m–1));end
        switch type
            case 'T';X(k,:)=(Y(k,:));
            case 'R';X(k,:)=fliplr(Y(k,:));
            case 'I';X(k,:)=–fliplr(Y(k,:));
            case 'G';X(k,:)=–(Y(k,:));
        end
    end
    SP=zeros(N–m+1,N–m+1);SPwsr=zeros(N–m+1,N–m+1);

    for j=1:N–m+1
        if m==1
             aux=repmat(signal(j),N–m+1,1);
            Nrec=find(abs(X–aux) <= r_factor);
            SP(j,Nrec)=1;
       else
            aux=repmat(Y(j,:),N–m+1,1);%
             Nrec=find(max(abs(X–aux)')' <= r_factor);
            SP(j,Nrec)=1;
        end
       end
 if display==1; %figure; spy(SP);axis xy;end
```

rqamarwan.m

```
function RQA = rqamarwan(varargin)
%RQA    Calculates recurrence quantification analysis
%Q=RQA(R,L,T) calculates measures of recurrence quantification analysis
% for recurrence plot R using  minimal line length L and a Theiler window T.
%
%     Output:
%       Y(1) = RR      (recurrence rate)
%       Y(2) = DET     (determinism)
%       Y(3) = <L>     (mean diagonal line length)
%       Y(4) = Lmax    (maximal diagonal line length)
%       Y(5) = ENTR    (entropy of the diagonal line lengths)
%       Y(6) = LAM     (laminarity)
%       Y(7) = TT      (trapping time)
%       Y(8) = Vmax    (maximal vertical line length)
%       Y(9) = RTmax   (maximal white vertical line length)
%       Y(10) = T2      (recurrence time of 2nd type)
%       Y(11) = RTE     (recurrence time entropy, i.e., RPDE)
%       Y(12) = Clust   (clustering coefficient)
%       Y(13) = Trans   (transitivity)
%
%Reference:
%Marwan, N., Romano, M. C., Thiel, M., Kurths, J. (2007).
%Recurrence plots for the analysis of
% complex systems.Physics Reports, 438, 237-329.
% Marwan, N., Donges, J. F., Zou, Y., Donner, R. V., Kurths, J. (2009).
% Complex network approach for recurrence
% analysis of time series. Physics Letters A, 373, 4246-4254.
%
%     Example:
%          N = 300; % length of time series
%          x = .9*sin((1:N)*2*pi/70); % exemplary time series
%          xVec = embed(x,2,17);
%          R = rp(xVec,.1);
%          Y = rqa_marwan(R);

% Copyright (c) 2016-2019
% Potsdam Institute for Climate Impact Research, Germany
% Institute of Geosciences, University of Potsdam, Germany
% Norbert Marwan, K. Hauke Kraemer
% http://www.pik-potsdam.de

%% check input
narginchk(1,3)
nargoutchk(0,1)

%% set default values for input parameters
theiler_window = 1; % Theiler window
l_min = 2; % minimal line length
```

```
%% get input arguments
% Theiler window
if nargin > 2
    theiler_window = varargin{3};
end

% minimal line length
if nargin > 1
    l_min = varargin{2};
end

% recurrence matrix
x = varargin{1};
N = size(x); % size of recurrence plot

%% apply Theiler window to recurrence plot
if theiler_window
   x_theiler = double(triu(x,theiler_window) + tril(x,-theiler_window));
else
   x_theiler = double(x);
end

% reduce the number of possible recurrence points by the Theiler window
N_all = N(1)*N(2); % all possible recurrence points
N_all = N_all - N(1) - 2*((theiler_window-1)*N(1) - sum(1:(theiler_window-1))
% reduced by Theiler window

%% calculation
y = zeros(13,1); % allocate result matrix

% recurrence rate
N_recpoints = sum(x_theiler(:)); % number of rec. points (in complete RP)
%y(1) = N_recpoints/N_all;
% Correction by JMG
y(1) = N_recpoints/2/((N(1)-1)*(N(2)-1)/2);

% histogram of diagonal lines (look at complete RP)
l_hist = zeros(1,N(1)); % allocate vector
for i = (1+theiler_window):N(1)
% walk along the raws (upper triangle)
   cnt = 0;
   for j = 1:(N(2)-(i-1))
      if x_theiler(i+j-1,j)
      % are we on a rec. point? (walk along a diagonal)
         cnt = cnt+1;
         % count number of points on a diagonal line
      else % line has ended
         if cnt
```

```
            l_hist(cnt) = l_hist(cnt) + 1; % store line length
        end
        cnt = 0; % set back to zero for a new line
    end
  end
  if cnt
      l_hist(cnt) = l_hist(cnt) + 1;
  end
end

% 2nd triangle
for j = (1+theiler_window):N(2)  % walk along the columns (lower triangle)
  cnt = 0;
  for i = 1:(N(1)-(j-1))
    if x_theiler(i,j+i-1) % are we on a rec. point? (walk along a diagonal)
      cnt = cnt+1; % count number of points on a diagonal line
    else % line has ended
      if cnt
          l_hist(cnt) = l_hist(cnt) + 1; % store line length
      end
      cnt = 0; % set back to zero for a new line
    end
  end
  if cnt
      l_hist(cnt) = l_hist(cnt) + 1;
  end
end

% determinism
y(2) = sum(l_hist(l_min:N(1)) .* (l_min:N(1))) / N_recpoints;

% mean diagonal line length
if isnan(sum(l_hist(l_min:N(1)) .* (l_min:N(1))) / sum(l_hist(l_min:N(1))))
    y(3) = 0;
else
    y(3) = sum(l_hist(l_min:N(1)) .* (l_min:N(1))) / sum(l_hist(l_min:N(1)));
end

% maximal line length
if any(l_hist)
  y(4) = find(l_hist,1,'last');
end

% line length entropy
l_classes = sum(l_hist~=0);
% number of occupied bins (for normalization of entropy)
l_prob = l_hist/sum(l_hist);
% get probability distribution from histogram
ent_Sum = (l_prob .* log(l_prob));
```

```
if l_classes > 1
    y(5) = -nansum(ent_Sum)/log(N(1));
else
    y(5) = -nansum(ent_Sum);
end

% histogram of vertical lines
v_hist = zeros(1,N(1)); % allocate vector
for i = 1:N(1) % walk along the columns
   cnt = 0;
   for j = 1:N(2)
      if x_theiler(j,i) % are we on a rec. point? (walk along a column)
         cnt = cnt+1; % count number of points on a vertical line
      else % line has ended
         if cnt
             v_hist(cnt) = v_hist(cnt) + 1; % store line length
         end
         cnt = 0; % set back to zero for a new line
      end
   end
   if cnt
       v_hist(cnt) = v_hist(cnt) + 1;
   end
end

% laminarity
y(6) = sum(v_hist(l_min:N(1)) .* (l_min:N(1))) / N_recpoints;

% mean vertical line length (trapping time)
if isnan(sum(v_hist(l_min:N(1)) .* (l_min:N(1))) / sum(v_hist(l_min:N(1))))
    y(7) = 0;
else
    y(7) = sum(v_hist(l_min:N(1)) .* (l_min:N(1))) / sum(v_hist(l_min:N(1)));
end

% maximal vertical length
if any(v_hist)
   y(8) = find(v_hist,1,'last');
end

% recurrence times ("white" vertical lines)
rt_hist = zeros(1,N(1)); % allocate vector
for i = 1:N(1)
   cnt = 0;

   % boolean variable to avoid counting white lines at the edges of RP
   first_flag = false;

   for j = 1:N(2)
```

```
        if ~x(j,i) % are we on a white line?
            if first_flag % line does not cross the RP's edges
                cnt = cnt + 1; % count number of points along the vertical line
            end
        else % we meet a recurrence point
            first_flag = true; % we are for sure within the RP
            if cnt
                rt_hist(cnt) = rt_hist(cnt) + 1; % store line length
            end
            cnt = 0;
        end
    end
end

% maximal white vertical line length
if any(rt_hist)
    y(9) = find(rt_hist,1,'last');
end

% recurrence time
y(10) = sum(rt_hist .* (1:N(1))) / sum(rt_hist);
if isnan(y(10))
    y(10)=0;
end

% recurrence time entropy
rt_classes = sum(rt_hist~=0);
% number of occupied bins (for normalization of entropy)
rt_prob = rt_hist/sum(rt_hist);
% get probability distribution from histogram
ent_Sum = (rt_prob .* log(rt_prob));
if rt_classes > 1
    y(11) = -nansum(ent_Sum)/log(N(1));
else
    y(11) = -nansum(ent_Sum);
end

% clustering
kv = sum(x_theiler,1); % degree of nodes
y(12) = nanmean(diag(x_theiler*x_theiler*x_theiler)' ./ (kv .* (kv-1)));

% transitivity
denom = sum(sum(x_theiler * x_theiler));
y(13) = trace(x_theiler*x_theiler*x_theiler)/denom;

RQA.RR    = y(1);%      Y(1) = RR     (recurrence rate)
RQA.DET   = y(2);%      Y(2) = DET    (determinism)
RQA.Lmoy  = y(3);%      Y(3) = <L>    (mean diagonal line length)
RQA.Lmax  = y(4);%      Y(4) = Lmax   (maximal diagonal line length)
```

```
RQA.ENTR  = y(5);%      Y(5) = ENTR   (entropy of the diagonal line lengths)
RQA.LAM   = y(6);%      Y(6) = LAM    (laminarity)
RQA.TT    = y(7);%      Y(7) = TT     (trapping time)
RQA.Vmax  = y(8);%      Y(8) = Vmax   (maximal vertical line length)
RQA.RTmax = y(9);%      Y(9) = RTmax  (maximal white vertical line length)
RQA.T2    = y(10);%     Y(10) = T2     (recurrence time of 2nd type)
RQA.RTE   = y(11);%     Y(11) = RTE    (recurrence time entropy, i.e., RPDE)
RQA.Clust = y(12);%     Y(12) = Clust  (clustering coefficient)
RQA.Trans = y(13);%     Y(13) = Trans  (transitivity)
```

Exemple3DisymmetrySawtoothSignal.m.

```
%  This algorithim shows a mean to measure
% the similarity degree between 2
% signals : symmetrical sawteeth with a delay
% and disymmetrical sawteeth with a delay.
clear,clc;close all
N=1000;
t = 0:1/(N-1):1;f0=4;
x1 = sawtooth(2*pi*f0*t,0.5);
x2 = sawtooth(2*pi*f0*t+pi/3,0.5);
%
epsilon=0.01;Centering=0;display=0;m=2;
[SPt]=SymPlot_JMGi(x1,'T',m,epsilon,Centering,display);
[SPr]=SymPlot_JMGi(x1,'R',m,epsilon,Centering,display);
[SPi]=SymPlot_JMGi(x1,'I',m,epsilon,Centering,display);
[SPg]=SymPlot_JMGi(x1,'G',m,epsilon,Centering,display);
%
figure;subplot(6,2,1);plot(t,x1);
title('Sym. Rec. Plots from a symmetric sawtooth')
subplot(6,4,5);spy(SPt);axis xy;title('T')
subplot(6,4,6);spy(SPr);axis xy;title('R')
subplot(6,4,9);spy(SPi);axis xy;title('I')
subplot(6,4,10);spy(SPg);axis xy;title('G')
%
epsilon=0.01;Centering=0;display=0;m=2;
[CSPt]=Cross2SymPlot_JMGi(x1,x2,'T',m,epsilon,Centering,display);
[CSPr]=Cross2SymPlot_JMGi(x1,x2,'R',m,epsilon,Centering,display);
[CSPi]=Cross2SymPlot_JMGi(x1,x2,'I',m,epsilon,Centering,display);
[CSPg]=Cross2SymPlot_JMGi(x1,x2,'G',m,epsilon,Centering,display);
%
subplot(6,2,2);plot(t,x1,t,x2);
title('Cross Sym. Rec. Plots from symmetric sawteeth')
subplot(6,4,7);spy(CSPt);axis xy;title('T')
subplot(6,4,8);spy(CSPr);axis xy;title('R')
subplot(6,4,11);spy(CSPi);axis xy;title('I')
subplot(6,4,12);spy(CSPg);axis xy;title('G')
%
clear,clc;
N=1000;t = 0:1/(N-1):1;f0=4;
x1 = sawtooth(2*pi*f0*t,0.25);
x2 = sawtooth(2*pi*f0*t+pi/3,0.5);
%
epsilon=0.01;Centering=0;display=0;m=2;
[SPt]=SymPlot_JMGi(x1,'T',m,epsilon,Centering,display);
[SPr]=SymPlot_JMGi(x1,'R',m,epsilon,Centering,display);
[SPi]=SymPlot_JMGi(x1,'I',m,epsilon,Centering,display);
[SPg]=SymPlot_JMGi(x1,'G',m,epsilon,Centering,display);
%
subplot(6,2,7);plot(t,x1);
```

```
title('Sym. Rec. Plots from a disymmetric sawtooth')
subplot(6,4,17);spy(SPt);axis xy;title('T')
subplot(6,4,18);spy(SPr);axis xy;title('R')
subplot(6,4,21);spy(SPi);axis xy;title('I')
subplot(6,4,22);spy(SPg);axis xy;title('G')
%
epsilon=0.01;Centering=0;display=0;m=2;
[CSPt]=Cross2SymPlot_JMGi(x1,x2,'T',m,epsilon,Centering,display);
[CSPr]=Cross2SymPlot_JMGi(x1,x2,'R',m,epsilon,Centering,display);
[CSPi]=Cross2SymPlot_JMGi(x1,x2,'I',m,epsilon,Centering,display);
[CSPg]=Cross2SymPlot_JMGi(x1,x2,'G',m,epsilon,Centering,display);
%
subplot(6,2,8);plot(t,x1,t,x2);
title('Cross Sym. Rec. Plots from disymmetric sawteeth')
subplot(6,4,19);spy(CSPt);axis xy;title('T')
subplot(6,4,20);spy(CSPr);axis xy;title('R')
subplot(6,4,23);spy(CSPi);axis xy;title('I')
subplot(6,4,24);spy(CSPg);axis xy;title('G')
```

Example4SRQASawtoothSignal.m

```
% This algorithm measures the degree of similarity between 2 time series
clear;close all;clc
epsilon=0.01;% epsilon is a tolerance require to test if 2 points
% are recurrent/similar in the PhaseSpaceSymPlot algorithm.
k=1;N=1000;
t = 0:1/(N-1):1;f0=4;x1 = sawtooth(2*pi*f0*t,0.25);
epsilon=0.01;Centering=0;display=0;m=2;
[SPt]=SymPlot_JMGi(x1,'T',m,epsilon,Centering,display);
[SPr]=SymPlot_JMGi(x1,'R',m,epsilon,Centering,display);
[SPi]=SymPlot_JMGi(x1,'I',m,epsilon,Centering,display);
[SPg]=SymPlot_JMGi(x1,'G',m,epsilon,Centering,display);
%
for a=0:0.01:1;
      x2 = sawtooth(2*pi*f0*t+pi/2,a);
    [CSPt]=Cross2SymPlot_JMGi(x1,x2,'T',m,epsilon,Centering,display);
    [CSPr]=Cross2SymPlot_JMGi(x1,x2,'R',m,epsilon,Centering,display);
    [CSPi]=Cross2SymPlot_JMGi(x1,x2,'I',m,epsilon,Centering,display);
    [CSPg]=Cross2SymPlot_JMGi(x1,x2,'G',m,epsilon,Centering,display);
% Calculation of the 4 kinds of Symmetrical RQA
% (T: Translation, R: Reflection, I: Inversion, G: Glide reflection ) as
% suggested in J.-M. Girault, "Recurrence and symmetry of time series:
% Application to transition detection"
% Chaos, Solitons and Fractals 77(2015)11--28, http://dx.doi.org/10.1016/j.ch
% RQA are calculated with the algorithm proposed by Marwan, and al.
% (2007). Recurrence plots for the analysis of complex systems,Physics Reports
% Marwan, and al. (2009). Complex network approach
% for recurrence analysis of time series.
%Physics Letters A, 373, 4246-4254
    RQAT =rqa_marwan(CSPt);
    RQAR = rqa_marwan(CSPr);
    RQAI =rqa_marwan(CSPi);
    RQAG = rqa_marwan(CSPg);
    % Extraction of each feature
    RRt(k)=RQAT.RR;DETt(k)=RQAT.DET;
    Lmoyt(k)=RQAT.Lmoy;Lmaxt(k)=RQAT.Lmax;
    ENTRt(k)=RQAT.ENTR;LAMt(k)=RQAT.LAM;TTt(k)=RQAT.TT;
    Vmaxt(k)=RQAT.Vmax;RTmaxt(k)=RQAT.RTmax;
    T2t(k)=RQAT.T2
    ;RTEt(k)=RQAT.RTE;Clustt(k)=RQAT.Clust;Transt(k)=RQAT.Trans;
%
    RRr(k)=RQAR.RR;DETr(k)=RQAR.DET;
    Lmoyr(k)=RQAR.Lmoy;Lmaxr(k)=RQAR.Lmax;
    ENTRr(k)=RQAR.ENTR;LAMr(k)=RQAR.LAM;TTr(k)=RQAR.TT;
    Vmaxr(k)=RQAR.Vmax;RTmaxr(k)=RQAR.RTmax;
    T2r(k)=RQAR.T2;RTEr(k)=RQAR.RTE;
    Clustr(k)=RQAR.Clust;Transr(k)=RQAR.Trans;
%
    RRi(k)=RQAI.RR;DETi(k)=RQAI.DET;
```

```
      Lmoyi(k)=RQAI.Lmoy;Lmaxi(k)=RQAI.Lmax;
      ENTRi(k)=RQAI.ENTR;LAMi(k)=RQAI.LAM;TTi(k)=RQAI.TT;
      Vmaxi(k)=RQAI.Vmax;RTmaxi(k)=RQAI.RTmax;
      T2i(k)=RQAI.T2;RTEi(k)=RQAI.RTE;
      Clusti(k)=RQAI.Clust;Transi(k)=RQAI.Trans;
%
      RRg(k)=RQAG.RR;DETg(k)=RQAG.DET;
      Lmoyg(k)=RQAG.Lmoy;Lmaxg(k)=RQAG.Lmax;
      ENTRg(k)=RQAG.ENTR;LAMg(k)=RQAG.LAM;TTg(k)=RQAG.TT;
      Vmaxg(k)=RQAG.Vmax;RTmaxg(k)=RQAG.RTmax;
      T2g(k)=RQAG.T2;RTEg(k)=RQAG.RTE;
      Clustg(k)=RQAG.Clust;Transg(k)=RQAG.Trans;
%
k=k+1
clear x2
end
%
save TestSawtooth
%
% Display each feature versus the  duty cycle
a=0:0.01:1;
subplot(341);plot(a,RRt,a,RRr,a,RRi,a,RRg);
title('RR');hold on;xline(0.5);
xline(0.25);xline(0.75);legend('T','R','I','G');xlim([0 1])
subplot(342);plot(a,DETt,a,DETr,a,DETi,a,DETg);
title('DET');hold on;xline(0.5);xline(0.25);xline(0.75);
legend('T','R','I','G');xlim([0 1])
subplot(343);plot(a,Lmoyt,a,Lmoyr,a,Lmoyi,a,Lmoyg);
title('Lmoy');hold on;xline(0.5);xline(0.25);xline(0.75);
legend('T','R','I','G');xlim([0 1])
subplot(344);plot(a,Lmaxt,a,Lmaxr,a,Lmaxi,a,Lmaxg);
title('Lmax');hold on;xline(0.5);xline(0.25);xline(0.75);
legend('T','R','I','G');xlim([0 1])
subplot(345);plot(a,ENTRt,a,ENTRr,a,ENTRi,a,ENTRg);
title('ENTR');hold on;xline(0.5);xline(0.25);xline(0.75);
legend('T','R','I','G');xlim([0 1])
subplot(346);plot(a,LAMt,a,LAMr,a,LAMi,a,LAMg);
title('LAM');hold on;xline(0.5);xline(0.25);xline(0.75);
legend('T','R','I','G');xlim([0 1])
subplot(347);plot(a,TTt,a,TTr,a,TTi,a,TTg);
title('TT');hold on;xline(0.5);
xline(0.25);xline(0.75);legend('T','R','I','G');xlim([0 1])
subplot(348);plot(a,Vmaxt,a,Vmaxr,a,Vmaxi,a,Vmaxg);
title('Vmax');hold on;xline(0.5);xline(0.25);xline(0.75);
legend('T','R','I','G');xlim([0 1])
subplot(349);plot(a,RTmaxt,a,RTmaxr,a,RTmaxi,a,RTmaxg);
title('RT');hold on;xline(0.5);xline(0.25);xline(0.75);
legend('T','R','I','G');xlim([0 1])
subplot(3,4,10);plot(a,T2t,a,T2r,a,T2i,a,T2g);
```

```
title('T2');hold on;xline(0.5);
xline(0.25);xline(0.75);legend('T','R','I','G');xlim([0 1] )
subplot(3,4,11);plot(a,RTEt,a,RTEr,a,RTEi,a,RTEg);
title('RTE');hold on;xline(0.5);xline(0.25);xline(0.75);
legend('T','R','I','G');xlim([0 1] )
subplot(3,4,12);plot(a,Clustt,a,Clustr,a,Clusti,a,Clustg);
title('Clust');hold on;xline(0.5);xline(0.25);xline(0.75);legend('T','R','I',
xlim([0 1] )
figure;plot(a,(Transt),a,(Transr),a,(Transi),a,(Transg));
title('Trans');hold on;xline(0.5);xline(0.25);
xline(0.75);legend('T','R','I','G');xlim([0 1] )
```

Example2ChaosDuffing.m

```
%% From the following reference "A. BenTal Physica D 171 (2002) 236-248",
%it is possible to show the interest of calculating indicators/features
%% derived from the Symmetrical Recurrence Quantification Analysis
% proposed in J.-M. Girault, "Recurrence and symmetry of time series:
% Application to transition detection"
% Chaos, Solitons and Fractals 77(2015)11-28 .
% The system under study is the Duffing system
% allowing the generation of new symmetries.
% The regime change appears at a=0.847,
%i.e passing from a system generating 1 scroll to a system with 2 scrolls:
%  --        --  --
% |  |  ==> |  ||  |
%  --        --  --
% Written by J.-M. Girault, jean-marc.girault@eseo.fr
clear;close all;clc
epsilon=0.01;% epsilon is a tolerance require to test if 2 points are
%recurrent/similar in the PhaseSpaceSymPlot algorithm.
k=1;
% Global variable required for the Duffing system
global mu c beta a omega
mu=10;c=100;beta=1;omega=3.5;M=20;
% Here, the control parameter is a ranging from 0.840 to 0.856

for a=0.840:0.001:0.856;
    % Generation of signal from the Duffing system
    [t,Out]=ode45(@duffing,0:2*pi/omega/M:200*2*pi/omega,[0 0]);
    % Display the 2D phase space of the Duffing system
    figure(1);subplot(221);plot(Out(:,1));subplot(223);plot(Out(:,2))
    subplot(122);plot(Out(:,1),Out(:,2));pause(0.1)
    % Calculation of the 4 symmetry Plots from the 2D phase space
    RPpsS=PhaseSpaceSymPlot(Out(:,1)',Out(:,2)','S',epsilon,epsilon);
    RPpsO=PhaseSpaceSymPlot(Out(:,1)',Out(:,2)','O',epsilon,epsilon);
    RPpsV=PhaseSpaceSymPlot(Out(:,1)',Out(:,2)','V',epsilon,epsilon);
    RPpsH=PhaseSpaceSymPlot(Out(:,1)',Out(:,2)','H',epsilon,epsilon);
    % Display the 4 symmetry Plots
    figure(2);subplot(311);plot(Out(:,1),Out(:,2));
    subplot(323);spy(RPpsS);axis xy;
    subplot(324);spy(RPpsO);axis xy;
    subplot(325);spy(RPpsV);axis xy;
    subplot(326);spy(RPpsH);axis xy;pause(0.1);
% Calculation of the 4 kinds of Symmetrical RQA
% (S: simple, O: Opposite, V: Vertical, H: Horizontal) as
% suggested in J.-M. Girault, "Recurrence and symmetry
% of time series: Application to transition detection"
%Chaos, Solitons and Fractals 77(2015)11-28,
% RQA are calculated with the algorithm proposed by Marwan, and al.,
%(2007). Recurrence plots for the analysis of complex systems,Physics Reports
% Marwan,  and al. (2009). Complex network approach for
```

```
% recurrence analysis of time series. Physics Letters A, 373, 4246-4254
    RQAS =rqamarwan(RPpsS);
    RQAO = rqamarwan(RPpsO);
    RQAV =rqamarwan(RPpsV);
    RQAH = rqamarwan(RPpsH);
    % Extraction of each feature
    RRs(k)=RQAS.RR;DETs(k)=RQAS.DET;Lmoys(k)=RQAS.Lmoy;Lmaxs(k)=RQAS.Lmax;
    ENTRs(k)=RQAS.ENTR;LAMs(k)=RQAS.LAM;TTs(k)=RQAS.TT;
    Vmaxs(k)=RQAS.Vmax;RTmaxs(k)=RQAS.RTmax;T2s(k)=RQAS.T2;RTEs(k)=RQAS.RTE;
    Clusts(k)=RQAS.Clust;Transs(k)=RQAS.Trans;

    RRo(k)=RQAO.RR;DETo(k)=RQAO.DET;Lmoyo(k)=RQAO.Lmoy;Lmaxo(k)=RQAO.Lmax;
    ENTRo(k)=RQAO.ENTR;LAMo(k)=RQAO.LAM;TTo(k)=RQAO.TT;
    Vmaxo(k)=RQAO.Vmax;RTmaxo(k)=RQAO.RTmax;T2o(k)=RQAO.T2;RTEo(k)=RQAO.RTE;
    Clusto(k)=RQAO.Clust;Transo(k)=RQAO.Trans;

    RRv(k)=RQAV.RR;DETv(k)=RQAV.DET;Lmoyv(k)=RQAV.Lmoy;Lmaxv(k)=RQAV.Lmax;
    ENTRv(k)=RQAV.ENTR;LAMv(k)=RQAV.LAM;TTv(k)=RQAV.TT;
    Vmaxv(k)=RQAV.Vmax;RTmaxv(k)=RQAV.RTmax;T2v(k)=RQAV.T2;RTEv(k)=RQAV.RTE;
    Clustv(k)=RQAV.Clust;Transv(k)=RQAV.Trans;

    RRh(k)=RQAH.RR;DETh(k)=RQAH.DET;Lmoyh(k)=RQAH.Lmoy;Lmaxh(k)=RQAH.Lmax;
    ENTRh(k)=RQAH.ENTR;LAMh(k)=RQAH.LAM;TTh(k)=RQAH.TT;
    Vmaxh(k)=RQAH.Vmax;RTmaxh(k)=RQAH.RTmax;T2h(k)=RQAH.T2;RTEh(k)=RQAH.RTE;
    Clusth(k)=RQAH.Clust;Transh(k)=RQAH.Trans;

k=k+1
clear t Out
end

save TestDuffingSystem

% Display each feature versus the control parameter a. Strong variations appe
a=0.840:0.001:0.856;
subplot(341);plot(a,RRs,a,RRo,a,RRv,a,RRh);title('RR');hold on;
xline(0.847);legend('S','O','V','H');xlim([0.840 0.856])
subplot(342);plot(a,DETs,a,10*DETo,a,10*DETv,a,DETh);
title('DET');hold on;xline(0.847);legend('S','10 x O','10 x V','H');
xlim([0.840 0.856])
subplot(343);plot(a,Lmoys,a,Lmoyo,a,Lmoyv,a,Lmoyh);
title('Lmoy');hold on;
xline(0.847);legend('S','O','V','H');xlim([0.840 0.856])
subplot(344);plot(a,Lmaxs,a,20*Lmaxo,a,20*Lmaxv,a,Lmaxh);
title('Lmax');hold on;
xline(0.847);legend('S','20 x O','20 x V','H');xlim([0.840 0.856])
subplot(345);plot(a,ENTRs,a,10*ENTRo,a,10*ENTRv,a,ENTRh);
title('ENTR');hold on;
xline(0.847);legend('S','10 x O','10 x V','H');xlim([0.840 0.856])
subplot(346);plot(a,LAMs,a,LAMo,a,LAMv,a,LAMh);
```

```
title('LAM');hold on;xline(0.847);
legend('S','O','V','H');xlim([0.840 0.856] )
subplot(347);plot(a,TTs,a,TTo,a,TTv,a,TTh);
title('TT');hold on;xline(0.847);
legend('S','O','V','H');xlim([0.840 0.856] )
subplot(348);plot(a,Vmaxs,a,Vmaxo,a,Vmaxv,a,Vmaxh);
title('Vmax');hold on;
xline(0.847);legend('S','O','V','H');xlim([0.840 0.856] )
subplot(349);plot(a,RTmaxs,a,RTmaxo,a,RTmaxv,a,RTmaxh);
title('RT');hold on;
xline(0.847);legend('S','O','V','H');xlim([0.840 0.856] )
subplot(3,4,10);plot(a,T2s,a,T2o,a,T2v,a,T2h);
title('T2');hold on;xline(0.847);
legend('S','O','V','H');xlim([0.840 0.856] )
subplot(3,4,11);plot(a,RTEs,a,RTEo,a,RTEv,a,RTEh);
title('RTE');hold on;
xline(0.847);legend('S','O','V','H');
xlim([0.840 0.856] )
subplot(3,4,12);plot(a,Clusts,a,Clusto,a,Clustv,a,Clusth);
title('Clust');hold on;
xline(0.847);legend('S','O','V','H');xlim([0.840 0.856] )
figure;plot(a,(Transs),a,10*(Transo),a,2*(Transv),a,20*(Transh));
title('Trans');
hold on;xline(0.847);legend('S','10 x O','2 x V','20 x H');
xlim([0.840 0.856] )
```

duffing.m

```
function xdot=duffing(t,x)
global mu c beta a omega
xdot(1)=mu*x(2)-c*x(2)^3-beta*x(1)+a*cos(omega*t);
xdot(2)=x(1);
xdot=xdot';
end

%% parameter derived from A. Ben-Tal Physica D 171 (2002) 236-248
%% collision of two conjugate chaotic attractors
    % mu=10;c=100;beta=1;a=0.848;omega=3.5;
%% explosion of two conjugate chaotic attractors
    %mu=10;c=100;beta=1;a=0.850;omega=3.5;%symmetry
%% explosion of two conjugate period one attractors
    %mu=1;c=1;beta=0.13;a=0.25210;omega=1;
    %mu=1;c=1;beta=0.13;a=0.25215;omega=1;
    %mu=1;c=1;beta=0.13;a=0.2522;omega=1;
%% symmetry restoring of a chaotic attractor results
% in a symmetric chaotic saddle
    %mu=1;c=1;beta=0.13;a=0.118;omega=1;
    %mu=1;c=1;beta=0.13;a=0.11839;omega=1;
    %mu=1;c=1;beta=0.13;a=0.1184;omega=1;
%% symmetry restoring of a chaotic saddle results
% in a symmetric chaotic saddle
    %mu=1;c=1;beta=0.13;a=0.0105;omega=1;
    %mu=1;c=1;beta=0.13;omega=1;a=0.01054
```

PhaseSpaceSymPlot.m

```
function PSSP=PhaseSpaceSymPlot(x1,x2,type,epsilon1,epsilon2);

% x1=[0 1 2 1 0 -1 -2 -1 0 1 2];epsilon1=0.01;
% x2=[2 1 0 -1 -2 -1 0 1 2 1 0];epsilon2=0.01;
% plot(x1);hold on;plot(x2,'r')
% type='V';
% PSSP=PhaseSpaceSymPlot(x1,x2,type,epsilon1,epsilon2);
N=length(x1);
switch type
            case 'S';X=[x1' x2'];
            case 'O';X=[x1' -x2'];
            case 'H';X=[-x1' -x2'];
            case 'V';X=[-x1' x2'];
end

PSSP=zeros(N,N);
for i=1:N;

Y=[repmat(x1(i),1,N)' repmat(x2(i),1,N)'];
D=abs(X-Y);
Ind=find( D(:,1) < epsilon1 & D(:,2) < epsilon2);
PSSP(i,Ind)=1;
end
%figure;spy(PSSP)
```

REFERENCES

[Girault (2015)] Girault, J.-M., "Recurrence and symmetry of time series: Application to transition detection", Chaos Solitons Fractals **2015**, 77, 11–28.

[Abdennaji (2021)] Abdennaji, I., Zaied, M. and Girault, J.-M., "Prediction of Protein Structural Classes Based on Symmetrical Recurrence Quantification Analysis", Computational Biology and Chemistry, 92, **2021**, 107450.

[Marwan (2007)] Marwan, N., Romano, M. C., Thiel, M., Kurths, J., "Recurrence plots for the analysis of complex systems", Physics Reports, **2007** 438, 237–329.

[Marwan (2009)] Marwan, N., Donges, J. F., Zou, Y., Donner, R. V., Kurths, J., "Complex network approach for recurrence analysis of time series", Physics Letters A, **2009** 373, 4246–4254.

[BenTal (2002)] BenTal, A., "Symmetry restoration in a class of forced oscillators", BenTal Physica D, **2002** 171, 236–2484.