SUPPLEMENTARY MATERIALS

J. J. Barba-Franco,[1] A. Gallegos,[1] R. Jaimes-Reátegui,[1] Jesús Muñoz-Maciel,[1] and A. N. Pisarchik[2]

[1] *Departamento de Ciencias Exactas y Tecnología, Centro Universitario de los Lagos, Universidad de Guadalajara, Enrique Díaz de León 1144, Colonia Paseos de la Montaña, 47460 Lagos de Moreno, Jalisco, Mexico*

[2] *Center for Biomedical Technology, Universidad Politécnica de Madrid, Campus de Montegancedo, Pozuelo de Alarcón, 28223 Madrid, Spain*

(*Electronic mail: alexander.pisarchik@ctb.upm.es)

We introduce MATLAB algorithms designed for numerical simulations of a system composed of ring-coupled Duffing oscillators. Our algorithms encompass various functionalities, including the computation of time series, Lyapunov exponents, power spectra, and instantaneous frequency. These codes provide a comprehensive toolkit for analyzing and understanding the dynamics of the ring-coupled Duffing oscillator system.

I. ALGORITHM FOR NUMERICAL SOLUTIONS OF THE UNIDIRECTIONAL RING OF N NODES

A. Function file with ODEs

```
function Z=RNN(t,Z)
alpha=0.4;omega=-0.25;delta=0.5;
global sigma NO % Global variables
if NO==3
        x1=Z(1);    y1=Z(2);
        x2=Z(3);    y2=Z(4);
        x3=Z(5);    y3=Z(6);
        elseif NO==4
        x4=Z(7);    y4=Z(8);
        elseif NO==5
        x5=Z(9);    y5=Z(10);
        elseif NO==6
        x6=Z(11);   y6=Z(12);
        elseif NO==7
        x7=Z(13);   y7=Z(14);
        elseif NO==8
        x8=Z(15);   y8=Z(16);
        elseif NO==9
        x9=Z(17);   y9=Z(18);
        elseif NO==10
        x10=Z(19);  y10=Z(20);
        elseif NO==11
        x11=Z(21);  y11=Z(22);
        elseif NO==12
        x12=Z(23);  y12=Z(24);
end
if NO==12
```

```
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x12), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3), ...
y5,-alpha*y5-omega.*x5-delta*x5.^3-sigma.*(x5-x4), ...
y6,-alpha*y6-omega.*x6-delta*x6.^3-sigma.*(x6-x5), ...
y7,-alpha*y7-omega.*x7-delta*x7.^3-sigma.*(x7-x6), ...
y8,-alpha*y8-omega.*x8-delta*x8.^3-sigma.*(x8-x7), ...
y9,-alpha*y9-omega.*x9-delta*x9.^3-sigma.*(x9-x8), ...
y10,-alpha*y10-omega.*x10-delta*x10.^3-sigma.*(x10-x9), ...
y11,-alpha*y11-omega.*x11-delta*x11.^3-sigma.*(x11-x10), ...
y12,-alpha*y12-omega.*x12-delta*x12.^3-sigma.*(x12-x11)];

elseif NO==11
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x11), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3), ...
y5,-alpha*y5-omega.*x5-delta*x5.^3-sigma.*(x5-x4), ...
y6,-alpha*y6-omega.*x6-delta*x6.^3-sigma.*(x6-x5), ...
y7,-alpha*y7-omega.*x7-delta*x7.^3-sigma.*(x7-x6), ...
y8,-alpha*y8-omega.*x8-delta*x8.^3-sigma.*(x8-x7), ...
y9,-alpha*y9-omega.*x9-delta*x9.^3-sigma.*(x9-x8), ...
y10,-alpha*y10-omega.*x10-delta*x10.^3-sigma.*(x10-x9), ...
y11,-alpha*y11-omega.*x11-delta*x11.^3-sigma.*(x11-x10)];

elseif NO==10
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x10), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3), ...
```

```
y5,-alpha*y5-omega.*x5-delta*x5.^3-sigma.*(x5-x4), ...
y6,-alpha*y6-omega.*x6-delta*x6.^3-sigma.*(x6-x5), ...
y7,-alpha*y7-omega.*x7-delta*x7.^3-sigma.*(x7-x6), ...
y8,-alpha*y8-omega.*x8-delta*x8.^3-sigma.*(x8-x7), ...
y9,-alpha*y9-omega.*x9-delta*x9.^3-sigma.*(x9-x8), ...
y10,-alpha*y10-omega.*x10-delta*x10.^3-sigma.*(x10-x9)];

elseif NO==9
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x9), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3), ...
y5,-alpha*y5-omega.*x5-delta*x5.^3-sigma.*(x5-x4), ...
y6,-alpha*y6-omega.*x6-delta*x6.^3-sigma.*(x6-x5), ...
y7,-alpha*y7-omega.*x7-delta*x7.^3-sigma.*(x7-x6), ...
y8,-alpha*y8-omega.*x8-delta*x8.^3-sigma.*(x8-x7), ...
y9,-alpha*y9-omega.*x9-delta*x9.^3-sigma.*(x9-x8)];

elseif NO==8
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x8), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3), ...
y5,-alpha*y5-omega.*x5-delta*x5.^3-sigma.*(x5-x4), ...
y6,-alpha*y6-omega.*x6-delta*x6.^3-sigma.*(x6-x5), ...
y7,-alpha*y7-omega.*x7-delta*x7.^3-sigma.*(x7-x6), ...
y8,-alpha*y8-omega.*x8-delta*x8.^3-sigma.*(x8-x7)];

elseif NO==7
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x7), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
```

```
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3), ...
y5,-alpha*y5-omega.*x5-delta*x5.^3-sigma.*(x5-x4), ...
y6,-alpha*y6-omega.*x6-delta*x6.^3-sigma.*(x6-x5), ...
y7,-alpha*y7-omega.*x7-delta*x7.^3-sigma.*(x7-x6)];

elseif NO==6
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x6), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3), ...
y5,-alpha*y5-omega.*x5-delta*x5.^3-sigma.*(x5-x4), ...
y6,-alpha*y6-omega.*x6-delta*x6.^3-sigma.*(x6-x5)];

elseif NO==5
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x5), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3), ...
y5,-alpha*y5-omega.*x5-delta*x5.^3-sigma.*(x5-x4)];

elseif NO==4
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x4), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2), ...
y4,-alpha*y4-omega.*x4-delta*x4.^3-sigma.*(x4-x3)];

elseif NO==3
Z=[y1,-alpha*y1-omega.*x1-delta*x1.^3-sigma.*(x1-x3), ...
y2,-alpha*y2-omega.*x2-delta*x2.^3-sigma.*(x2-x1), ...
y3,-alpha*y3-omega.*x3-delta*x3.^3-sigma.*(x3-x2)];
```

```
end
```

B. Main (script) file

```
clear all
clc
a=0;b=15000; %Time
Mi=2^17; %Number of steps
h=(15000-a)/Mi; %Step size
M=(b-a)/h;  % Full number of steps
global sigma NO % Global variables
%
for NO=3:1:12 % Number of oscillators (Nodes) in the ring
        for i=1:1:1000
        sigma=0.004*i;  % Coupling (0<=sigma<=4)
        sigma_2(i)=sigma;
        % Initial Conditions for Xj,Yj, taken aleatory from -10 to 10
        Za=-10+(10+10)*rand(2*NO,1); % Range of IC (-10 -> +10)
        % where NO is the size of the ring
        [T,Z]=rks4('RNN',a,b,Za,M);
        SX=[T' Z];  % Matriz of solutions [Time Xj Yj] for j=1,...NO
        % We save the solutions of the ODES for each coupling value
eval(['save Sx_',int2str(NO),'N_ICrd_s_',int2str(i),'.dat SX -ascii']);
        end
end
```

C. Function file with the Runge-Kutta algorithm

```
        function [T,Z]=rks4(F,a,b,Za,M)
        h=(b-a)/M;
```

```
T=zeros(1,M+1);
T=a:h:b;  Z(1,:)=Za;
 for  j=1:M
k1=h*feval(F,T(j),Z(j,:));
k2=h*feval(F,T(j)+h/2,Z(j,:)+k1/2);
k3=h*feval(F,T(j)+h/2,Z(j,:)+k2/2);
k4=h*feval(F,T(j)+h,Z(j,:)+k3);
Z(j+1,:)=Z(j,:)+(k1+2*k2+2*k3+k4)/6;
end
```

II. ALGORITHM FOR BUILDING BIFURCATION DIAGRAMS OF LOCAL MAXIMA

A. Function file to calculate local maxima

1. Function file to calculate local maxima for the ring of 12 nodes

```
 if  abs(vx1(end)-vx1(end-1))==0
%          x1
 if  abs(vx1(end)-vx1(end-1))<amin
%        1
 bifx1(m,:)=vx1(end-45:end-1);
 else
%        2
vx1  =  vx1(1:max(size(vx1)));
lmx1  =  locmax(vx1);
vx1  =  vx1(lmx1);
 for  jv  =  0:1:44
 bifx1(m,jv+1)  =  vx1(max(size(vx1))-jv);
end
end
```

```
%          x2
if abs(vx2(end)-vx2(end-1))<amin
%        1
bifx2(m,:)=vx2(end-45:end-1);
else
%        2
vx2 = vx2(1:max(size(vx2)));
lmx2 = locmax(vx2);
vx2 = vx2(lmx2);
for jv = 0:1:44
bifx2(m,jv+1) = vx2(max(size(vx2))-jv);
end
end
%          x3
if abs(vx3(end)-vx3(end-1))<amin
%        1
bifx3(m,:)=vx3(end-45:end-1);
else
%        2
vx3 = vx3(1:max(size(vx3)));
lmx3 = locmax(vx3);
vx3 = vx3(lmx3);
for jv = 0:1:44
bifx3(m,jv+1) = vx3(max(size(vx3))-jv);
end
end
%          x4
if abs(vx4(end)-vx4(end-1))<amin
%        1
bifx4(m,:)=vx4(end-45:end-1);
else
```

```
%         2
vx4 = vx4(1:max(size(vx4)));
lmx4 = locmax(vx4);
vx4 = vx4(lmx4);
for jv = 0:1:44
bifx4(m,jv+1) = vx4(max(size(vx4))-jv);
end
end
%          x5
if abs(vx5(end)-vx5(end-1))<amin
%         1
bifx5(m,:)=vx5(end-45:end-1);
else
%         2
vx5 = vx5(1:max(size(vx5)));
lmx5 = locmax(vx5);
vx5 = vx5(lmx5);
for jv = 0:1:44
bifx5(m,jv+1) = vx5(max(size(vx5))-jv);
end
end
%          x6
if abs(vx6(end)-vx6(end-1))<amin
%         1
bifx6(m,:)=vx6(end-45:end-1);
else
%         2
vx6 = vx6(1:max(size(vx6)));
lmx6 = locmax(vx6);
vx6 = vx6(lmx6);
for jv = 0:1:44
```

```
bifx6(m,jv+1) = vx6(max(size(vx6))-jv);
end
end
%       x7
if abs(vx7(end)-vx7(end-1))<amin
%      1
bifx7(m,:)=vx7(end-45:end-1);
else
%      2
vx7 = vx7(1:max(size(vx7)));
lmx7 = locmax(vx7);
vx7 = vx7(lmx7);
for jv = 0:1:44
bifx7(m,jv+1) = vx7(max(size(vx7))-jv);
end
end
%       x8
if abs(vx8(end)-vx8(end-1))<amin
%      1
bifx8(m,:)=vx8(end-45:end-1);
else
%      2
vx8 = vx8(1:max(size(vx8)));
lmx8 = locmax(vx8);
vx8 = vx8(lmx8);
for jv = 0:1:44
bifx8(m,jv+1) = vx8(max(size(vx8))-jv);
end
end
%       x9
if abs(vx9(end)-vx9(end-1))<amin
```

```
%       1
bifx9(m,:)=vx9(end-45:end-1);
else
%       2
vx9 = vx9(1:max(size(vx9)));
lmx9 = locmax(vx9);
vx9 = vx9(lmx9);
for jv = 0:1:44
bifx9(m,jv+1) = vx9(max(size(vx9))-jv);
end
end
%         x10
if abs(vx10(end)-vx10(end-1))<amin
%       1
bifx10(m,:)=vx10(end-45:end-1);
else
%       2
vx10 = vx10(1:max(size(vx10)));
lmx10 = locmax(vx10);
vx10 = vx10(lmx10);
for jv = 0:1:44
bifx10(m,jv+1) = vx10(max(size(vx10))-jv);
end
end
%         x11
if abs(vx11(end)-vx11(end-1))<amin
%       1
bifx11(m,:)=vx11(end-45:end-1);
else
%       2
vx11 = vx11(1:max(size(vx11)));
```

```
lmx11 = locmax(vx11);
vx11 = vx11(lmx11);
for jv = 0:1:44
bifx11(m,jv+1) = vx11(max(size(vx11))-jv);
end
end
%       x12
if abs(vx12(end)-vx12(end-1))<amin
%      1
bifx12(m,:)=vx12(end-45:end-1);
else
%      2
vx12 = vx12(1:max(size(vx12)));
lmx12 = locmax(vx12);
vx12 = vx12(lmx12);
for jv = 0:1:44
bifx12(m,jv+1) = vx12(max(size(vx12))-jv);
end
end
%       y1
if abs(vy1(end)-vy1(end-1))<amin
%      1
bify1(m,:)=vy1(end-45:end-1);
else
%      2
vy1 = vy1(1:max(size(vy1)));
lmy1 = locmax(vy1);
vy1 = vy1(lmy1);
for jv = 0:1:44
bify1(m,jv+1) = vy1(max(size(vy1))-jv);
end
```

```
end
%         y2
if abs(vy2(end)-vy2(end-1))<amin
%       1
bify2(m,:)=vy2(end-45:end-1);
else
%       2
vy2 = vy2(1:max(size(vy2)));
lmy2 = locmax(vy2);
vy2 = vy2(lmy2);
for jv = 0:1:44
bify2(m,jv+1) = vy2(max(size(vy2))-jv);
end
end
%         y3
if abs(vy3(end)-vy3(end-1))<amin
%       1
bify3(m,:)=vy3(end-45:end-1);
else
%       2
vy3 = vy3(1:max(size(vy3)));
lmy3 = locmax(vy3);
vy3 = vy3(lmy3);
for jv = 0:1:44
bify3(m,jv+1) = vy3(max(size(vy3))-jv);
end
end
%         y4
if abs(vy4(end)-vy4(end-1))<amin
%       1
bify4(m,:)=vy4(end-45:end-1);
```

```
else
%        2
vy4 = vy4(1:max(size(vy4)));
lmy4 = locmax(vy4);
vy4 = vy4(lmy4);
for jv = 0:1:44
bify4(m,jv+1) = vy4(max(size(vy4))-jv);
end
end
%          y5
if abs(vy5(end)-vy5(end-1))<amin
%        1
bify5(m,:)=vy5(end-45:end-1);
else
%        2
vy5 = vy5(1:max(size(vy5)));
lmy5 = locmax(vy5);
vy5 = vy5(lmy5);
for jv = 0:1:44
bify5(m,jv+1) = vy5(max(size(vy5))-jv);
end
end
%          y6
if abs(vy6(end)-vy6(end-1))<amin
%        1
bify6(m,:)=vy6(end-45:end-1);
else
%        2
vy6 = vy6(1:max(size(vy6)));
lmy6 = locmax(vy6);
vy6 = vy6(lmy6);
```

```
for jv = 0:1:44
bify6(m,jv+1) = vy6(max(size(vy6))-jv);
end
end
%      y7
if abs(vy7(end)-vy7(end-1))<amin
%     1
bify7(m,:)=vy7(end-45:end-1);
else
%     2
vy7 = vy7(1:max(size(vy7)));
lmy7 = locmax(vy7);
vy7 = vy7(lmy7);
for jv = 0:1:44
bify7(m,jv+1) = vy7(max(size(vy7))-jv);
end
end
%      y8
if abs(vy8(end)-vy8(end-1))<amin
%     1
bify8(m,:)=vy8(end-45:end-1);
else
%     2
vy8 = vy8(1:max(size(vy8)));
lmy8 = locmax(vy8);
vy8 = vy8(lmy8);
for jv = 0:1:44
bify8(m,jv+1) = vy8(max(size(vy8))-jv);
end
end
%      y9
```

```
if abs(vy9(end)-vy9(end-1))<amin
%        1
bify9(m,:)=vy9(end-45:end-1);
else
%        2
vy9 = vy9(1:max(size(vy9)));
lmy9 = locmax(vy9);
vy9 = vy9(lmy9);
for jv = 0:1:44
bify9(m,jv+1) = vy9(max(size(vy9))-jv);
end
end
%          y10
if abs(vy10(end)-vy10(end-1))<amin
%        1
bify10(m,:)=vy10(end-45:end-1);
else
%        2
vy10 = vy10(1:max(size(vy10)));
lmy10 = locmax(vy10);
vy10 = vy10(lmy10);
for jv = 0:1:44
bify10(m,jv+1) = vy10(max(size(vy10))-jv);
end
end
%          y11
if abs(vy11(end)-vy11(end-1))<amin
%        1
bify11(m,:)=vy11(end-45:end-1);
else
%        2
```

```
vy11 = vy11(1:max(size(vy11)));
lmy11 = locmax(vy11);
vy11 = vy11(lmy11);
for jv = 0:1:44
bify11(m,jv+1) = vy11(max(size(vy11))-jv);
end
end
%       y12
if abs(vy12(end)-vy12(end-1))<amin
%      1
bify12(m,:)=vy12(end-45:end-1);
else
%      2
vy12 = vy12(1:max(size(vy12)));
lmy12 = locmax(vy12);
vy12 = vy12(lmy12);
for jv = 0:1:44
bify12(m,jv+1) = vy12(max(size(vy12))-jv);
end
end
else
%       x1
vx1 = vx1(1:max(size(vx1)));
lmx1 = locmax(vx1);
vx1 = vx1(lmx1);
for jv = 0:1:44
bifx1(m,jv+1) = vx1(max(size(vx1))-jv);
end
%       x2
vx2 = vx2(1:max(size(vx2)));
lmx2 = locmax(vx2);
```

```
vx2 = vx2(lmx2);
for jv = 0:1:44
bifx2(m,jv+1) = vx2(max(size(vx2))-jv);
end
%       x3
vx3 = vx3(1:max(size(vx3)));
lmx3 = locmax(vx3);
vx3 = vx3(lmx3);
for jv = 0:1:44
bifx3(m,jv+1) = vx3(max(size(vx3))-jv);
end
%       x4
vx4 = vx4(1:max(size(vx4)));
lmx4 = locmax(vx4);
vx4 = vx4(lmx4);
for jv = 0:1:44
bifx4(m,jv+1) = vx4(max(size(vx4))-jv);
end
%       x5
vx5 = vx5(1:max(size(vx5)));
lmx5 = locmax(vx5);
vx5 = vx5(lmx5);
for jv = 0:1:44
bifx5(m,jv+1) = vx5(max(size(vx5))-jv);
end
%       x6
vx6 = vx6(1:max(size(vx6)));
lmx6 = locmax(vx6);
vx6 = vx6(lmx6);
for jv = 0:1:44
bifx6(m,jv+1) = vx6(max(size(vx6))-jv);
```

```
end
%        x7
vx7 = vx7(1:max(size(vx7)));
lmx7 = locmax(vx7);
vx7 = vx7(lmx7);
for jv = 0:1:44
bifx7(m,jv+1) = vx7(max(size(vx7))-jv);
end
%        x8
vx8 = vx8(1:max(size(vx8)));
lmx8 = locmax(vx8);
vx8 = vx8(lmx8);
for jv = 0:1:44
bifx8(m,jv+1) = vx8(max(size(vx8))-jv);
end
%        x9
vx9 = vx9(1:max(size(vx9)));
lmx9 = locmax(vx9);
vx9 = vx9(lmx9);
for jv = 0:1:44
bifx9(m,jv+1) = vx9(max(size(vx9))-jv);
end
%        x10
vx10 = vx10(1:max(size(vx10)));
lmx10 = locmax(vx10);
vx10 = vx10(lmx10);
for jv = 0:1:44
bifx10(m,jv+1) = vx10(max(size(vx10))-jv);
end
%        x11
vx11 = vx11(1:max(size(vx11)));
```

```
lmx11 = locmax(vx11);
vx11 = vx11(lmx11);
for jv = 0:1:44
bifx11(m,jv+1) = vx11(max(size(vx11))-jv);
end
%        x12
vx12 = vx12(1:max(size(vx12)));
lmx12 = locmax(vx12);
vx12 = vx12(lmx12);
for jv = 0:1:44
bifx12(m,jv+1) = vx12(max(size(vx12))-jv);
end
%        y1
vy1 = vy1(1:max(size(vy1)));
lmy1 = locmax(vy1);
vy1 = vy1(lmy1);
for jv = 0:1:44
bify1(m,jv+1) = vy1(max(size(vy1))-jv);
end
%        y2
vy2 = vy2(1:max(size(vy2)));
lmy2 = locmax(vy2);
vy2 = vy2(lmy2);
for jv = 0:1:44
bify2(m,jv+1) = vy2(max(size(vy2))-jv);
end
%        y3
vy3 = vy3(1:max(size(vy3)));
lmy3 = locmax(vy3);
vy3 = vy3(lmy3);
for jv = 0:1:44
```

```
bify3(m,jv+1) = vy3(max(size(vy3))-jv);
end
%       y4
vy4 = vy4(1:max(size(vy4)));
lmy4 = locmax(vy4);
vy4 = vy4(lmy4);
for jv = 0:1:44
bify4(m,jv+1) = vy4(max(size(vy4))-jv);
end
%       y5
vy5 = vy5(1:max(size(vy5)));
lmy5 = locmax(vy5);
vy5 = vy5(lmy5);
for jv = 0:1:44
bify5(m,jv+1) = vy5(max(size(vy5))-jv);
end
%       y6
vy6 = vy6(1:max(size(vy6)));
lmy6 = locmax(vy6);
vy6 = vy6(lmy6);
for jv = 0:1:44
bify6(m,jv+1) = vy6(max(size(vy6))-jv);
end
%       y7
vy7 = vy7(1:max(size(vy7)));
lmy7 = locmax(vy7);
vy7 = vy7(lmy7);
for jv = 0:1:44
bify7(m,jv+1) = vy7(max(size(vy7))-jv);
end
%       y8
```

```
vy8 = vy8(1:max(size(vy8)));
lmy8 = locmax(vy8);
vy8 = vy8(lmy8);
for jv = 0:1:44
bify8(m,jv+1) = vy8(max(size(vy8))-jv);
end
%       y9
vy9 = vy9(1:max(size(vy9)));
lmy9 = locmax(vy9);
vy9 = vy9(lmy9);
for jv = 0:1:44
bify9(m,jv+1) = vy9(max(size(vy9))-jv);
end
%       y10
vy10 = vy10(1:max(size(vy10)));
lmy10 = locmax(vy10);
vy10 = vy10(lmy10);
for jv = 0:1:44
bify10(m,jv+1) = vy10(max(size(vy10))-jv);
end
%       y11
vy11 = vy11(1:max(size(vy11)));
lmy11 = locmax(vy11);
vy11 = vy11(lmy11);
for jv = 0:1:44
bify11(m,jv+1) = vy11(max(size(vy11))-jv);
end
%       y12
vy12 = vy12(1:max(size(vy12)));
lmy12 = locmax(vy12);
vy12 = vy12(lmy12);
```

```
        for jv = 0:1:44
        bify12(m,jv+1) = vy12(max(size(vy12))-jv);
        end
        end
```

2. Function file to calculate local maximum for the ring of 3–11 nodes

The construction of the bifurcations diagrams for the rings of oscillators from 3 to 11 nodes is very similar to 12 nodes, but it depends on the number of oscillators in the ring. Therefore, it is not necessary to add all others algorithms.

B. Main (script) file

```
close all, clear all, clc
%% Parameters
a=0;b=15000; %Time
Mi=2^18;
h=(15000-a)/Mi; %Step length
M=(b-a)/h; %Number of steps
amin=1e-1;
format short
for NO=3:1:12
        for k=1:1:1000
        i=(1001-k);
        sigma=0.004*i;
% We load the solutions for each case of study and rename the matrix
        eval(['load Sx_',int2str(NO),'N_ICrd_s_',int2str(i),'.dat']);
        eval(['V = Sx_',int2str(NO),'N_ICrd_s_',int2str(i),';']);
        clear Sx** % Clear memory
        m=i;
        %% The data is identified
        t=V(:,1);
```

```
u=V(:,2:end);
dt=h;
    for ij=1:1:NO
        axi=2*ij-1;
        ayi=2*ij;
        eval(['x',int2str(ij),'= u(:,',int2str(axi),');']);
        eval(['y',int2str(ij),'= u(:,',int2str(ayi),');']);
        eval(['vx',int2str(ij),'= u(:,',int2str(axi),');']);
        eval(['vy',int2str(ij),'= u(:,',int2str(ayi),');']);
    end
        %% The maxima are calculate
        if NO==3
                BD3;
        elseif NO==4
                BD4;
        elseif NO==5
                BD5;
        elseif NO==6
                BD6;
        elseif NO==7
                BD7;
        elseif NO==8
                BD8;
        elseif NO==9
                BD9;
        elseif NO==10
                BD10;
        elseif NO==11
                BD11;
        elseif NO==12
                BD12;
```

```
                    end


          end
% We save the data for bifurcation diagrams
for ij=1:1:NO
eval(['save BDx',int2str(ij),'_R',int2str(NO),'N.dat bifx',num2str(ij),
' -ascii']);
% For x's
eval(['save BDy',int2str(ij),'_R',int2str(NO),'N.dat bify',num2str(ij),
' -ascii']);
% For y's
end

end
```

C. Function file with local maxima algorithm

```
%         local maxima
function y = locmax(x)

x = 2000*x./(max(x)-min(x));
dy = x(2:max(size(x)))-x(1:max(size(x))-1);
z = find(dy>0)+1;
dz = z(2:max(size(z)))-z(1:max(size(z))-1);
zz = find(dz>1);
y = z(zz);

% sintax: z = locmax(u)
```

III. ALGORITHM FOR CALCULATING LYAPUNOV EXPONENTS FROM TIME SERIES

This algorithm loads each time series calculated before, evaluates it, and generates the respective Lyapunov exponents. Then it saves the data in the (.dat) format.

```
%%
clear all; close all; clc;
format long;
a=0;b=15000; %Time
Mi=2^17; %Number of steps
h=(15000-a)/Mi; %Step size
M=(b-a)/h;  % Full number of steps
global sigma NO % Global variables
for NO=3:1:12 % Number of oscillators (Nodes) in the ring
        for j=1:1:1000   % Coupling (0<=sigma<=4)
                sigma=0.004*j;
                sigma_2(j)=sigma;
                for no=1:1:NO % For rings form 3 to 12
                        i=2*no;     % x's
                        k=2*no+1;   % y's
        % load the data (time series)
        eval(['load Sx_',int2str(NO),'N_ICrd_s_',int2str(j),'.dat']);
        eval(['v = Sx_',int2str(NO),'N_ICrd_s_',int2str(j),';']);
        clear Sx**
        % Idetifies the time series
        if ((locmax(v(end-2^15:end,:)))>0)
        siz=2^16;
        else
        siz=2^18;
        end
        eval (['x = v(end-siz+1:end,',int2str(i),');']); % x's
        eval (['y = v(end-siz+1:end,',int2str(k),');']); % y's
```

```
% Save the time series in .lor format
save xa1.lor x -ascii % x's
save ya1.lor y -ascii % y's

%% Solves for x and y
% x's
fnamex  = 'xa1.lor';  % for x's
% y's
fnamey  = 'ya1.lor';  % for y's
% Inicialization
datcnt= length(x)
tau = 5;
ndim = 2;
ires = 11;
maxbox = 6000;
% Funcion Lyap
% x's
dbx  = basgen(fnamex,  tau, ndim, ires, datcnt, maxbox); %x's
% y's
dby  = basgen(fnamey,  tau, ndim, ires, datcnt, maxbox); %y's

% System parameters
dt = h;
evolve = 20;
dismin = 0.001;
dismax = 0.3;
thmax = 30;
%
% x's
[ELx,  SUMx]  = fet(dbx,  dt, evolve, dismin, dismax, thmax);
% y's
```

```
          [ELy,  SUMy]  = fet(dby,  dt, evolve, dismin, dismax, thmax);

          % Outputs
          lyap_x = ELx(:,4); % x's
          lyap_y = ELy(:,4); % y's

          % We save the LE for each oscillator and coupling value
          EL=[lyap_x lyap_y];
eval(['save EL_',int2str(NO),'N_n_',int2str(no),'_s_',int2str(j),'.dat
EL -ascii']);

                    end
          end
end
```

A. Algorithm for choosing the largest Lyapunov exponent as a function of the coupling parameter

This algorithm loads each Lyapunov exponent calculated before, then it chooses the largest Lypunov exponent for each coupling value and each oscillator. Then it saves the data in (.dat) format.

```
close all
clear all
clc
global sigma NO % Global variables
for NO=3:1:12 % Number of oscillators (Nodes) in the ring
          for j=1:1:1000   % Coupling (0<=sigma<=4)
                    sigma=0.004*j;
                    sigma_2(j)=sigma;
                    for no=1:1:NO % For rings form 3 to 12
                              i=2*no;     % x's
```

```
                        k=2*no+1;  % y's
                        % load the data (LE)
eval(['load EL_',int2str(NO),'N_n_',int2str(no),'_s_',int2str(j),'.dat']);
eval(['EL = EL_',int2str(NO),'N_n_',int2str(no),'_s_',int2str(j),';']);

                        x = EL(:,1); % x's
                        y = EL(:,2); % y's
      %% Last value of EL (We eliminated all NaN and +- Inf)
      % x(end)
                        TFx = isnan(x);
                        Nx = find(~TFx);
                        xNx = x(Nx);
                        TFix = isfinite(xNx);
                        Nix = find(TFix);
                        xNix = xNx(Nix);
                        LEfix(j)=xNix(end);

                        % y(end)
                        TFy = isnan(y);
                        Ny = find(~TFy);
                        yNy = y(Ny);
                        TFiy = isfinite(yNy);
                        Niy = find(TFiy);
                        yNiy = yNy(Niy);
                        LEfiy(j)=yNiy(end);

                  end

%% We save the data
eval (['lyap_x',int2str(no),'= LEfix;']); % x's
eval (['lyap_y',int2str(no),'= LEfiy;']); % y's
```

```
%
eval(['save ELend_',int2str(NO),'N_x_',num2str(no),'.dat LEfix -ascii']);
eval(['save ELend_',int2str(NO),'N_y_',num2str(no),'.dat LEfiy -ascii']);

        end
end
```

1. Function (fet) file used to calculate Lyapunov exponents

```
function [out, SUM] = fet(db, dt, evolve, dismin, dismax, thmax)
% Computes Lyapunov exponent of given data and parameters
% Generates output
% textfile, exact replica of Fortran 77 version of fet
% Taehyeun Park, The Cooper Union, EE'15

out = [];

ndim = db.ndim;
ires = db.ires;
tau = db.tau;
datcnt = db.datcnt;
datmin = db.datmin;
boxlen = db.boxlen;

datptr = db.datptr;
nxtbox = db.nxtbox;
where = db.where;
nxtdat = db.nxtdat;
data = db.data;

delay = 0:tau:(ndim-1)*tau;
```

```
datuse = datcnt-(ndim-1)*tau-evolve;

its = 0;
SUM = 0;
savmax = dismax;

oldpnt = 1;
newpnt = 1;

fileID = fopen('fetout.txt', 'w');

goto50 = 1;
while goto50 == 1;
goto50 = 0;
[bstpnt, bstdis, thbest] = search(0, ndim, ires, datmin, boxlen, nxtbox,
where, ...
datptr, nxtdat, data, delay, oldpnt, newpnt, datuse, dismin, dismax,...
thmax, evolve);

while bstpnt == 0
dismax = dismax * 2;
[bstpnt, bstdis, thbest] = search(0, ndim, ires, datmin, boxlen, nxtbox,
where, ...
datptr, nxtdat, data, delay, oldpnt, newpnt, datuse, dismin, dismax,...
thmax, evolve);
end

dismax = savmax;
newpnt = bstpnt;
disold = bstdis;
iang = -1;
```

```
goto60 = 1;
while goto60 == 1;
goto60 = 0;

oldpnt = oldpnt + evolve;
newpnt = newpnt + evolve;

if oldpnt >= datuse
return
end

if newpnt >= datuse
oldpnt = oldpnt - evolve;
goto50 = 1;
break
end

p1 = data(oldpnt + delay);
p2 = data(newpnt + delay);
disnew = sqrt(sum((p2 - p1).^2));

its = its + 1;

SUM = SUM + log(disnew/disold);
zlyap = SUM/(its*evolve*dt*log(2));
out = [out; its*evolve, disold, disnew, zlyap, (oldpnt-evolve),
(newpnt-evolve)];

if iang == -1
fprintf(fileID,'%-d\t\t\t%-8.4f\t\t%-8.4f\t\t%-8.4f\n',out(end,1:4)');
```

```
else
fprintf(fileID,'%-d\t\t\t%-8.4f\t\t%-8.4f\t\t
%-8.4f\t\t%-d\n',[out(end,1:4),iang]');
end

if disnew <= dismax
disold = disnew;
iang = -1;
goto60 = 1;
continue
end

[bstpnt, bstdis, thbest] = search(1, ndim, ires, datmin, boxlen, nxtbox,
where, ...
datptr, nxtdat, data, delay, oldpnt, newpnt, datuse, dismin, dismax,...
thmax, evolve);

if bstpnt ~= 0
newpnt = bstpnt;
disold = bstdis;
iang = floor(thbest);
goto60 = 1;
continue
else
goto50 = 1;
break;
end
end
end
fclose(fileID);
```

2. Function (makeplot) file used to calculate Lyapunov exponents

```
function [] = makeplot(db, out, evolve, loc)
% Plots 2D or 3D attractor evolution by evolution, 4th parameter is the
% location of legend
% Taehyeun Park, The Cooper Union, EE'15

datcnt = db.datcnt;
ndim = db.ndim;
tau = db.tau;
dataplot = [];
freerun = 0;

delay = 0:tau:(ndim-1)*tau;
data = db.data;

for ii = 1:(datcnt-(ndim-1)*tau)
dataplot = [dataplot; data(ii+delay)];
end

figure, bar(out(:,1),out(:,3)), hold on;
mle = max(dataplot(:)) - min(dataplot(:));
plot([0, out(end,1)], [mle, mle], 'r', 'LineWidth', 1.5), hold off;
set(gca,'YTick', [0, mle])
axis([0, out(end,1), 0, 1.1*mle])
title('d_f of evolutions scaled to the maximum linear extent of the
attractor')

if ndim == 2
figure('Position', [100, 100, 800, 500]);
plot(dataplot(:,1), dataplot(:,2), '.', 'MarkerSize', 3), hold on;
display('To see the next evolution, press enter')
```

```
display('To clear the screen and then see the next evolution, ...
type c and press enter')
display('To proceed without stopping, type r and press enter')
display('To terminate plot generating, type g and press enter')

for ii = 1:size(out,1)
if freerun == 0
RESET = input('Next evolution?  ', 's');
if strcmp(RESET, 'c')
display('Screen cleared')
hold off;
clf;
plot(dataplot(:,1), dataplot(:,2), '.', 'MarkerSize', 3), hold on;
elseif strcmp(RESET, 'r')
display('Evolving without stopping...')
display('Press ctrl+c to terminate')
freerun = 1;
elseif strcmp(RESET, 'g')
display('Plot generating stopped')
return;
else
if ii > 1
delete(ann)
end
end
end

tmpold = out(ii,5);
oldpnt = tmpold + evolve;
tmpnew = out(ii,6);
newpnt = tmpnew + evolve;
```

```
plot(data(tmpold:oldpnt), data((tmpold+tau):(oldpnt+tau)), 'r',
'LineWidth', 1);
plot(data(tmpnew:newpnt), data((tmpnew+tau):(newpnt+tau)), 'g',
'LineWidth', 1);
for aa = 0:evolve;
plot([data(tmpold+aa), data(tmpnew+aa)], ...
[data(tmpold+aa+tau), data(tmpnew+aa+tau)], 'LineWidth', 1)
end

ann = legend(['Iteration: ', num2str(out(ii,1)), '/',
num2str(out(end,1)), ...
char(10) 'd_i:', num2str(out(ii,2)), char(10)...
'd_f:', num2str(out(ii,3)), char(10)...
'Current Estimate:' num2str(out(ii,4))], ...
'location', loc);
if freerun == 1
drawnow
end
end

elseif ndim == 3
figure('Position', [100, 100, 800, 500]);
plot3(dataplot(:,1), dataplot(:,2), dataplot(:,3), '.', 'MarkerSize', 3),
hold on;
display('To see the next evolution, press enter')
display('To clear the screen and then see the next evolution, ...
type c and press enter')
display('To proceed without stopping, type r and press enter')
display('To terminate plot generating, type g and press enter')
```

```
for ii = 1:size(out,1)
if freerun == 0
RESET = input('Next evolution?  ', 's');
if strcmp(RESET, 'c')
display('Screen cleared')
hold off;
clf;
plot3(dataplot(:,1), dataplot(:,2), dataplot(:,3), '.', 'MarkerSize', 3),
hold on;
elseif strcmp(RESET, 'r')
display('Evolving without stopping...')
display('Press ctrl+c to terminate')
freerun = 1;
elseif strcmp(RESET, 'g')
display('Plot generating stopped')
return;
else
if ii > 1
delete(ann)
end
end
end

tmpold = out(ii,5);
oldpnt = tmpold + evolve;
tmpnew = out(ii,6);
newpnt = tmpnew + evolve;

plot3(data(tmpold:oldpnt), data((tmpold+tau):(oldpnt+tau)), ...
data((tmpold+(2*tau)):(oldpnt+(2*tau))), 'r', 'LineWidth', 1);
plot3(data(tmpnew:newpnt), data((tmpnew+tau):(newpnt+tau)), ...
```

```
data((tmpnew+(2*tau)):(newpnt+(2*tau))), 'g', 'LineWidth', 1);
for aa = 0:evolve;
plot3([data(tmpold+aa), data(tmpnew+aa)], [data(tmpold+aa+tau),
data(tmpnew+aa+tau)], ...
[data(tmpold+aa+(2*tau)), data(tmpnew+aa+(2*tau))], 'LineWidth', 1)
end

ann = legend(['Iteration: ', num2str(out(ii,1)), '/',
num2str(out(end,1)),...
char(10) 'd_i:', num2str(out(ii,2)), char(10)...
'd_f:', num2str(out(ii,3)), char(10)...
'Current Estimate:' num2str(out(ii,4))], ...
'location', loc);
if freerun == 1
drawnow
end
end
end
```

3. Function (search) file used to calculate Lyapunov exponents

```
function [bstpnt, bstdis, thbest] = search(iflag, ndim, ires, datmin,...
boxlen, nxtbox, where, datptr, nxtdat, data, delay, oldpnt, newpnt,...
datuse, dismin, dismax, thmax, evolve)
% Searches for the most viable point for fet.m
% Taehyeun Park, The Cooper Union, EE'15

target = zeros(1,ndim);
oldcrd = zeros(1,ndim);
zewcrd = zeros(1,ndim);
```

```
oldcrd(1:ndim) = data(oldpnt+delay);
zewcrd(1:ndim) = data(newpnt+delay);
igcrds = floor((oldcrd - datmin)./boxlen);
oldist = sqrt(sum((oldcrd - zewcrd).^2));

irange = round(dismin/boxlen);
if irange == 0;
irange = 1;
end

thbest = thmax;
bstdis = dismax;
bstpnt = 0;

goto30 = 1;
while goto30 == 1
goto30 = 0;
for icnt = 0:((2*irange+1)^ndim)-1
goto140 = 0;
icounter = icnt;
for ii = 1:ndim;
ipower = (2*irange+1)^(ndim-ii);
ioff = floor(icounter./ipower);
icounter = icounter - ioff*ipower;
target(ii) = igcrds(ii) - irange + ioff;

if target(ii) < 0
goto140 = 1;
break;
end
if target(ii) > ires-1
```

```
goto140 = 1;
break
end
end

if goto140 == 1;
continue
end

if irange ~= 1
iskip = 1;
for ii = 1:ndim
if abs(round(target(ii) - igcrds(ii))) == irange
iskip = 0;
end
end
if iskip == 1
continue
end
end

runner = 1;
for ii = 1:ndim
goto80 = 0;
goto70 = 1;
while goto70 == 1;
goto70 = 0;
if where(runner,ii) == target(ii)
goto80 = 1;
break
end
```

```
runner = nxtbox(runner, ii);
if runner ~= 0
goto70 = 1;
end
end

if goto80 == 1
continue
end
goto140 = 1;
break
end

if goto140 == 1
continue
end

if runner == 0
continue
end
runner = datptr(runner);
if runner == 0
continue
end
goto90 = 1;
while goto90 == 1
goto90 = 0;
while 1;
if abs(round(runner - oldpnt)) < evolve
break
end
```

```
if abs(round(runner - datuse)) < (2*evolve)
break
end

bstcrd = data(runner + delay);

abc1 = oldcrd(1:ndim) - bstcrd(1:ndim);
abc2 = oldcrd(1:ndim) - zewcrd(1:ndim);
tdist = sum(abc1.*abc1);
tdist = sqrt(tdist);
dot = sum(abc1.*abc2);

if tdist < dismin
break
end
if tdist >= bstdis
break
end
if tdist == 0
break
end
goto120 = 0;
if iflag == 0
goto120 = 1;
end
if goto120 == 0
ctheta = min(abs(dot/(tdist*oldist)),1);
theta = 57.3*acos(ctheta);
if theta >= thbest
break
end
```

```
thbest = theta;
end
bstdis = tdist;
bstpnt = runner;
break;
end
runner = nxtdat(runner);

if runner ~= 0
goto90 = 1;
end
end
end
irange = irange + 1;
if irange <= (0.5 + round((dismax/boxlen)))
goto30 = 1;
continue;
end
return
end
```

4. Function (basegen) file used to calculate Lyapunov exponents

```
function db = basgen(fname, tau, ndim, ires, datcnt, maxbox)
% Database generator for fet.m function
% Taehyeun Park, The Cooper Union, EE'15

x = fileread(fname);
data = zeros(1,datcnt);
trck = 1;
start = 1;
```

```
fin = 0;

for ii = 1:length(x)
if strcmp(x(ii), char(32)) || strcmp(x(ii), char(13)) || strcmp(x(ii), ...
char(10)) || strcmp(x(ii), char(26))
if fin >= start
data(trck) = str2num(x(start:fin));
trck = trck + 1;
if trck > 8*floor(datcnt/8)
break
end
end
start = ii + 1;
else
fin = ii;
end
end

delay = 0:tau:(ndim-1)*tau;

nxtbox = zeros(maxbox, ndim);
where = zeros(maxbox, ndim);
datptr = zeros(1,maxbox);
nxtdat = zeros(1,datcnt);

datmin = min(data);
datmax = max(data);

datmin = datmin - 0.01*(datmax - datmin);
datmax = datmax + 0.01*(datmax - datmin);
boxlen = (datmax - datmin)/ires;
```

```
boxcnt = 1;

for ii = 1:(datcnt-(ndim-1)*tau)
target = floor((data(ii+delay)-datmin)/boxlen);
runner = 1;
chaser = 0;

jj = 1;
while jj <= ndim
tmp = where(runner,jj)-target(jj);
if tmp < 0
chaser = runner;
runner = nxtbox(runner,jj);
if runner ~= 0
continue
end
end
if tmp ~= 0
boxcnt = boxcnt + 1;

if boxcnt == maxbox
error('Grid overflow, increase number of box count')
end

for kk = 1:ndim
where(boxcnt,kk) = where(chaser,kk);
end
where(boxcnt,jj) = target(jj);
nxtbox(chaser,jj) = boxcnt;
nxtbox(boxcnt,jj) = runner;
```

```
runner = boxcnt;
end
jj = jj + 1;
end
nxtdat(ii) = datptr(runner);
datptr(runner) = ii;
end

used = 0;
for ii = 1:boxcnt
if datptr(ii) ~= 0;
used = used + 1;
end
end
display(['Created:  ', num2str(boxcnt)]);
display(['Used:  ', num2str(used)]);

db.ndim = ndim;
db.ires = ires;
db.tau = tau;
db.datcnt = datcnt;
db.boxcnt = boxcnt;
db.datmax = datmax;
db.datmin = datmin;
db.boxlen = boxlen;

db.datptr = datptr(1:boxcnt);
db.nxtbox = nxtbox(1:boxcnt, 1:ndim);
db.where = where(1:boxcnt, 1:ndim);
db.nxtdat = nxtdat(1:datcnt);
db.data = data;
```

IV. ALGORITHM TO BUILD FOURIER TRANSFORMATIONS

A. Main (script) file to generate Fourier transformations

```
clear all
clc
a=0;b=15000; %Time
Mi=2^17; %Number of steps
h=(15000-a)/Mi; %Step size
M=(b-a)/h;  % Full number of steps
global sigma NO % Global variables
%% Power spectrum (Fast Fourier transformations)
for i=1:1:1000
        sigma=0.004*(k-1); % Coupling (0<=sigma<=4)
        % Load the time series calculated before for each coupling value
        eval(['load EPL_sigma_',int2str(i),'.dat']);
        % We rename the data
        eval(['V = EPL_sigma_',int2str(i),';']);
        clear EPL** % clear memory

        t=V(:,1); %time
        x=V(:,2:end); % ODES solutions
        %
        x1=x(:,1); %x1
        y1=x(:,2); %y1
        x2=x(:,3); %x2
        y2=x(:,4); %y2
        x3=x(:,5); %x3
        y3=x(:,6); %y3
        %
        %%  Forier transformations
        nstep=max(size(x1));
```

```
tau=h;
% Response in frequency and power spectrum
f(1:nstep) =2*pi*(0:(nstep -1))/(tau*nstep);
f(1:nstep) =(0:(nstep -1))/(tau*nstep);

x1fft = fft(x1); % Fourier transform of displacement
spect_1 = abs(x1fft).^2; % Power spectrum of displacement
x2fft = fft(x2); % Fourier transform of displacement
spect_2= abs(x2fft).^2; % Power spectrum of displacement
x3fft = fft(x3); % Fourier transform of displacement
spect_3 = abs(x3fft).^2; % Power spectrum of displacement
%
y1fft = fft(y1); % Fourier transform of displacement
spect_y1 = abs(y1fft).^2; % Power spectrum of displacement
y2fft = fft(y2); % Fourier transform of displacement
spect_y2= abs(y2fft).^2; % Power spectrum of displacement
y3fft = fft(y3); % Fourier transform of displacement
spect_y3 = abs(y3fft).^2; % Power spectrum of displacement
%
spect_1_a = spect_1(1:(nstep/2)); % Power spectrum of displacement
f_1_a=f(1:(nstep/2)); % Fourier transform of displacement
spect_2_a = spect_2(1:(nstep/2)); % Power spectrum of displacement
f_2_a=f(1:(nstep/2)); % Fourier transform of displacement
spect_3_a = spect_3(1:(nstep/2)); % Power spectrum of displacement
f_3_a=f(1:(nstep/2)); % Fourier transform of displacement
%
spect_1_y = spect_y1(1:(nstep/2)); % Power spectrum of displacement
f_1_y=f(1:(nstep/2)); % Fourier transform of displacement
spect_2_y = spect_y2(1:(nstep/2)); % Power spectrum of displacement
f_2_y=f(1:(nstep/2)); % Fourier transform of displacement
spect_3_y = spect_y3(1:(nstep/2)); % Power spectrum of displacement
```

```
        f_3_y=f(1:(nstep/2));

        % We save the solutions for each coupling value
        eval(['save FTSx_s_',int2str(i),'.dat SX -ascii']);
```

V. ALGORITHM TO COMPUTE THE INSTANTANEOUS FREQUENCY

A. Function file with Hilbert algorithm

```
        function xdbar =fhilb(x)
        %xdbar is the analituc signal for X
        % x must be row vector . dbar is olse row vctor
        n=length(x);
        X=fft(x);
        sft=[1 2*ones(1,n/2-1) 1 zeros(1,n/2-1) ];
        xdbar=ifft(sft.*X);
```

B. Main (script) file

```
clear all
clc
for k=1:1:1001
        sigma=0.004*(k-1); % Coupling (0<=sigma<=4)
        sigma_2(k)=sigma;
        % Load the time series calculated before for each coupling value
        eval(['load EPL_sigma_',int2str(k),'.dat']);
        % We rename the data
        eval(['V = EPL_sigma_',int2str(k),';']);
        clear EPL** % clear memory

        t=V(:,1); %time
```

```
x=V(:,2:end);
%
x1=x(:,1); %x1
y1=x(:,2); %y1
x2=x(:,3); %x2
y2=x(:,4); %y2
x3=x(:,5); %x3
y3=x(:,6); %y3
%
% t=t(1:end);
n = length(t);
%% Complex analytical function (Hilbert transformations)

xa1 = fhilb(x1); %x1
xe1 = abs(xa1);  %x1
xa2 = fhilb(x2); %x2
xe2 = abs(xa2);  %x2
xa3 = fhilb(x3); %x3
xe3 = abs(xa3);  %x3
ya1 = fhilb(y1); %y1
ye1 = abs(ya1);  %y1
ya2 = fhilb(y2); %y2
ye2 = abs(ya2);  %y2
ya3 = fhilb(y3); %y3
ye3 = abs(ya3);  %y3
% Envelope waves
exe1=xe1(1:end)';
exe2=xe2(1:end)';
exe3=xe3(1:end)';
eye1=ye1(1:end)';
eye2=ye2(1:end)';
```

```
eye3=ye3(1:end)';

% x1(t)
d_thetax1 = zeros(size(t));
thetasx1 = angle(xa1); %angle solution (phase)
thetax1 = unwrap(thetasx1); %reshape the phase
d_thetax1(1) = (thetax1(2)-thetax1(1))/dt;
d_thetax1(n) = (thetax1(n-1)-thetax1(n))/dt;
% x2(t)
d_thetax2 = zeros(size(t));
thetasx2 = angle(xa2);
thetax2 = unwrap(thetasx2);
d_thetax2(1) = (thetax2(2)-thetax2(1))/dt;
d_thetax2(n) = (thetax2(n-1)-thetax2(n))/dt;
% x3(t)
d_thetax3 = zeros(size(t));
thetasx3 = angle(xa3);
thetax3 = unwrap(thetasx3);
d_thetax3(1) = (thetax3(2)-thetax3(1))/dt;
d_thetax3(n) = (thetax3(n-1)-thetax3(n))/dt;
% y1(t)
d_thetay1 = zeros(size(t));
thetasy1 = angle(ya1);
thetay1 = unwrap(thetasy1);
d_thetay1(1) = (thetay1(2)-thetay1(1))/dt;
d_thetay1(n) = (thetay1(n-1)-thetay1(n))/dt;
% y2(t)
d_thetay2 = zeros(size(t));
thetasy2 = angle(ya2);
thetay2 = unwrap(thetasy2);
d_thetay2(1) = (thetay2(2)-thetay2(1))/dt;
```

```
d_thetay2(n) = (thetay2(n-1)-thetay2(n))/dt;
% y3(t)
d_thetay3 = zeros(size(t));
thetasy3 = angle(ya3);
thetay3 = unwrap(thetasy3);
d_thetay3(1) = (thetay3(2)-thetay3(1))/dt;
d_thetay3(n) = (thetay3(n-1)-thetay3(n))/dt;
%
for k = 2:n-1
d_thetax1(k) = (thetax1(k+1)-thetax1(k-1))/(2*dt); %x1(t)
d_thetax2(k) = (thetax2(k+1)-thetax2(k-1))/(2*dt); %x2(t)
d_thetax3(k) = (thetax3(k+1)-thetax3(k-1))/(2*dt); %x3(t)
d_thetay1(k) = (thetay1(k+1)-thetay1(k-1))/(2*dt); %y1(t)
d_thetay2(k) = (thetay2(k+1)-thetay2(k-1))/(2*dt); %y2(t)
d_thetay3(k) = (thetay3(k+1)-thetay3(k-1))/(2*dt); %y3(t)
end
%% Output variables
fi_x1=(d_thetax1/2/pi);
fi_x2=(d_thetax2/2/pi);
fi_x3=(d_thetax3/2/pi);
fi_y1=(d_thetay1/2/pi);
fi_y2=(d_thetay2/2/pi);
fi_y3=(d_thetay3/2/pi);
% We save the data
TH=[t fi_x1 fi_y1 fi_x2 fi_y2 fi_x3 fi_y3];
eval(['save FI_TH_EPL_s_',num2str(k),'.dat TH -ascii']);
```

C. Function file with local maxima algorithm

```
%       local maxima
function y = locmax(x)
```

```
x = 2000*x./(max(x)-min(x));
dy = x(2:max(size(x)))-x(1:max(size(x))-1);
z = find(dy>0)+1;
dz = z(2:max(size(z)))-z(1:max(size(z))-1);
zz = find(dz>1);
y = z(zz);

% sintax: z = locmax(u)
```