Primary Data Description: On equilibrium fluctuations, von Storch, August 2022

The data are produced by the following MATLAB scripts given below in blue. Most of them use the functions **lorz_n.m, lorz_f_n_shift.m** to get Lorenz solutions x and its differential forcing f. lorz_f_n_shift writes out not only x (as it is done in lorz_n) also the differential forcing f.

	Scripts	Functions & data needed
Figure 2	**plot_ts** calculates and plots two different Lorenz equilibrium solutions	**lorz_n** **var.mat** contains the initial conditions produced by **get_var**
Figure 3	**get_var** calculates variance a) by time averaging a single equilibrium solution, and b) by averaging over an ensemble of equilibrium solutions. The time averaged variance is a function of the length of the solution. The ensemble averaged variance is a function of ensemble size. Equilibrium solutions are obtained by - starting from a state (x1,y1,z1), which is an equilibrium state that is obtained by integrating the Lorenz model over a long time period (stored in **var.mat**) and then slightly perturbed by adding a normal random variable - integrate this slightly-perturbed state for another nts time steps to produce a spin-up integration - integrate the Lorenz model from the end of the spin-up integration to produce an equilibrium solution **plot_var** plots the result	**lorz_n** produces a solution of the Lorenz model using the Runge Kutta method and a time step dt=0.01. The solution starts from (x1,y1,z1) and has the length n
Figure 4	**get_c** calculates auto-correlation functions of x from a single equilibrium solution of three different lengths **plot_c** plots the time-averaged auto-correlation functions	**lorz_n** **var.mat**
Figure 5	**get_gamma** calculates auto-correlation functions of x from an ensemble of three different ensemble sizes **plot_gamma** plots the ensemble-averaged auto-correlation functions	**lorz_n** **var.mat**
Figure 6	**get_gamma_f_n** calculates auto-covariance functions of f from an ensemble of size en=1e+6 **get_gamma_xf** calculates cross-covariance functions between x and f from an ensemble of size en=1e+6 **plot_gamma_xf** plots the results	**lorz_f_n_shift** **var.mat**

	Scripts	Functions & data needed
Figure 7	**get_spxf_shift** calculates ensemble averaged spectra, including both the power spectra of x and those of f, and the co-spectra and the coherence and phase spectra between x and f, using one ensemble of size 1000 consisting equilibrium solutions of length 1e+7+1 **get_white_lf_var** a) identifies the frequency at which the spectrum deviates 5% from its low-frequency plateau and stores the result in **omega.mat** (needed in plot_spxf), b) calculates the variance associated with the low-frequency plateau **plot_spxf** plots power spectra of x and f and the coherence and phase spectra between x and f	**lorz_n** **lorz_f_n_shift** **var.mat** **omega.mat**
Figure 8	**plot_spf_linear** plots the spectra of f and the amplitude of the cross-spectrum between x and f (output of **get_spxf_shift**) in linear scale (rather than log scale)	
Figure9	**get_signal** calculates the spectra of two impulse-like signals. The non-zero values at the central piece of a signal is obtained from an AR1-process **plot_signal** plots the resulting spectra	**ar1**
Figure B1	**get_sp_m** calculate spectra derived from three ensembles of size 1, 10 100. The ensembles contain equilibrium solutions of size 1e+7+1. The spectra derived from an ensemble of size 1000 is calculated by get_spxf_shift. **plot_sp_m** plots spectra of the first component obtained from four ensembles of sizes 1, 10, 100, and 1000.	**lorz_n** **var.mat**
Figure B2	**get_sp_n** calculates spectra averaged over 4 ensembles of size=1000, each consisting equilibrium solutions of length 2N=1e+3, 1e+4, 1e+5, 1e+6. **plot_sp_n** plots the result	**lorz_n** **var.mat**
Figure C1	**get_sp_n_plus1** calculates the difference between two sample spectra, $S_{k,N+1}$-$S_{k,N,}$ The two sample spectra are derived from two non-overlapping pieces of the same equilibrium solution, with length 2(N+1)+1 and 2N+1 **plot_sp_n_plus1** plots the result	**lorz_n** **var.mat**
Figure C2	**get_fc** calculates the Fourier coefficient at the same frequency from 100 equilibrium solutions of length T **get_fc_1** calculates the the Fourier coefficient at the same frequency from 100 records of length T from a single equilibrium solution **plot_fc** plots the Fourier coefficients	**lorz_n** **var.mat**

```
##########################################################################
###################    lorz_n.m                                  ###################
##########################################################################

function [x] = lorz_n(x1,y1,z1,dt,n)
% integrating the Lorenz model from (x1,y1,z1) with time step dt for n time steps
% jsvs, Mai 2022
%

x=zeros(n,3);
f=zeros(n,3);

a      = 10;
b      = 28;
c      = 8/3;

for i=1:n
   Fx     = a*y1 - a*x1;
   k1     = Fx ;
   k2     = a*y1 - a*(x1+0.5*k1*dt);
   k3     = a*y1 - a*(x1+0.5*k2*dt);
   k4     = a*y1 - a*(x1+k3*dt);
   f(i,1)  =(k1 + 2*k2 + 2*k3 + k4)/6;
   x(i,1)   = x1 + dt*f(i,1);

   Fy     = x1*(b - z1) - y1;
   k1     = Fy ;
   k2     = x1*(b - z1) - (y1+0.5*k1*dt);
   k3     = x1*(b - z1) - (y1+0.5*k2*dt);
   k4     = x1*(b - z1) - (y1+k3*dt);
   f(i,2)  =(k1 + 2*k2 + 2*k3 + k4)/6;
   x(i,2)   = y1 + dt*f(i,2);

   Fz     = x1*y1 - c*z1;
   k1     = Fz ;
   k2     = x1*y1 - c*(z1+0.5*k1*dt);
   k3     = x1*y1 - c*(z1+0.5*k2*dt);
   k4     = x1*y1 - c*(z1+k3*dt);
   f(i,3)  =(k1 + 2*k2 + 2*k3 + k4)/6;
   x(i,3)   = z1 + dt*f(i,3);

   x1=x(i,1);
   y1=x(i,2);
   z1=x(i,3);
end

end


##########################################################################
###################    lorz_f_n_shift.m                          ###################
##########################################################################


function [sx,sf] = lorz_f_n_shift(x1,y1,z1,dt,n)
% integrating the Lorenz model from (x1,y1,z1) with time step dt for n time steps
%
% Note:
% the script performs the following integration:
%   x(s) = x(s-1) + f(x(s-1))*dt
```

```
% and write out x(s) in x(i) and f(x(s-1)) in f(i).
% In order to get the timing right, f is shifted to get
%   sf(1:n)=f(2:n+1)
%   sx(1:n)=x(1:n)
%
% Without shifting: the cospectrum between x and f is essentially positive
% With shifting:  the cospectrum is essentially negative
%
% jsvs, Mai 2022
%

x=zeros(n+1,3);
f=zeros(n+1,3);
sx=zeros(n,3);
sf=zeros(n,3);

a       = 10;
b       = 28;
c       = 8/3;

for i=1:n+1
    Fx      = a*y1 - a*x1;
    k1      = Fx ;
    k2      = a*y1 - a*(x1+0.5*k1*dt);
    k3      = a*y1 - a*(x1+0.5*k2*dt);
    k4      = a*y1 - a*(x1+k3*dt);
    f(i,1)   =(k1 + 2*k2 + 2*k3 + k4)/6;
    x(i,1)   = x1 + dt*f(i,1);

    Fy      = x1*(b - z1) - y1;
    k1      = Fy ;
    k2      = x1*(b - z1) - (y1+0.5*k1*dt);
    k3      = x1*(b - z1) - (y1+0.5*k2*dt);
    k4      = x1*(b - z1) - (y1+k3*dt);
    f(i,2)   =(k1 + 2*k2 + 2*k3 + k4)/6;
    x(i,2)   = y1 + dt*f(i,2);

    Fz      = x1*y1 - c*z1;
    k1      = Fz ;
    k2      = x1*y1 - c*(z1+0.5*k1*dt);
    k3      = x1*y1 - c*(z1+0.5*k2*dt);
    k4      = x1*y1 - c*(z1+k3*dt);
    f(i,3)   =(k1 + 2*k2 + 2*k3 + k4)/6;
    x(i,3)   = z1 + dt*f(i,3);

    x1=x(i,1);
    y1=x(i,2);
    z1=x(i,3);
end

sx(1:n,:)=x(1:n,:);
sf(1:n,:)=f(2:n+1,:);

end
```

```
##########################################################################
##################    plot_ts.m   (for Fig.2)                 ##################
##########################################################################


% Get and plot two equilibrium Lorenz solutions
%
% The solution is obtained by
% - starting from a state (x1,y1,z1), obtained by slightly perturbing an
%   equilibrated state
% - integrate this state for another nts time steps
% - integrate the Lorenz model for another nt time steps
% - plotting the solution over a short and a long time interval, starting
%   at nd1 and ends at nd2
%
%
%  jsvs, 2021
%

clear
load('var.mat');

nt=1e+5+1;       % length of time series considered for spectra

nts=1e+5;
varn=0.1;        % variance of the random number used to generate
                 % an ensemble

dt=0.01;         % time stepping of the Lorenz model,
                 % dt=0.01 is used in the paper

wk1=zeros(nt,3);  %  x
wk2=zeros(nt,3);  %  x

rng('default');

x1=x0+varn*randn(1,1);
y1=y0+varn*randn(1,1);
z1=z0+varn*randn(1,1);
[wk] = lorz_n(x1,y1,z1,dt,nts);
x1=wk(nts,1);
y1=wk(nts,2);
z1=wk(nts,3);
[wk1] = lorz_n(x1,y1,z1,dt,nt);
wk1=wk1-mean(wk1);

x1=x0+varn*randn(1,1);
y1=y0+varn*randn(1,1);
z1=z0+varn*randn(1,1);
[wk] = lorz_n(x1,y1,z1,dt,nts);
x1=wk(nts,1);
y1=wk(nts,2);
z1=wk(nts,3);
[wk2] = lorz_n(x1,y1,z1,dt,nt);
wk2=wk2-mean(wk2);

fs=12;
figure('Position', [0 0 800 400])

% plot the short time series
```

```
subplot(2,2,1);
nd1=1;       % the (nd1+1)-th to the nd2-th time steps will be plotted
nd2=2000;
t1=1:(nd2-nd1+1); % counting  the  (nd1+1)-th to thr nd2_th time steps
zz1=t1*0;
plot(t1,wk1(nd1:nd2,1),'g',t1,wk1(nd1:nd2,2),'r',t1,wk1(nd1:nd2,3),'b',t1,zz1,'k')
%xlabel('time step')
ylabel('x')
ylim([-25 25])
xlim([1 nd2-nd1])
legend('comp.1', 'comp.2', 'comp.3' )
grid on
ax = gca;
ax.FontSize = fs;

subplot(2,2,3);
plot(t1,wk2(nd1:nd2,1),'g',t1,wk2(nd1:nd2,2),'r',t1,wk2(nd1:nd2,3),'b',t1,zz1,'k')
xlabel('time step')
ylabel('x')
ylim([-25 25])
xlim([1 nd2-nd1])
grid on
ax = gca;
ax.FontSize = fs;

% plot the long time series
subplot(2,2,2);
nd1=20001;       % the (nd1+1)-th to the nd2-th time steps will be plotted
nd2=40000;
t1=1:(nd2-nd1+1);   % counting  the  (nd1+1)-th to thr nd2_th time steps
zz1=t1*0;
plot(t1,wk1(nd1:nd2,1),'g',t1,wk1(nd1:nd2,2),'r',t1,wk1(nd1:nd2,3),'b',t1,zz1,'k')
%xlabel('time step')
%ylabel('x')
ylim([-25 25])
xlim([1 nd2-nd1])
grid on
ax = gca;
ax.FontSize = fs;

subplot(2,2,4);
plot(t1,wk2(nd1:nd2,1),'g',t1,wk2(nd1:nd2,2),'r',t1,wk2(nd1:nd2,3),'b',t1,zz1,'k')
xlabel('time step')
%ylabel('x')
ylim([-25 25])
xlim([1 nd2-nd1])
grid on
ax = gca;
ax.FontSize = fs;


##########################################################################
###################    get.var.m    (for Fig.3)                 ###################
##########################################################################

% a) get variance v_en as ensemble variance obtained from an ensemble of
%    equilibrium solutions at one time point,  for different values of
%    ensemble size en (according to Eq.10b in the paper)
% b) get variance v_nt as time variance  of one equilibrium solution, for
%    different lengths of solution  nt (according to Eq.10a in the paper)
```

```
%
% Equilibrium solutions are obtained by
% - starting from a  state (x1,y1,z1), which is a slightly-perturbed
%   equilibrium state (x0,y0,z0) obtained by integrating the Lorenz model
%   over a long time period.
% - integrate this slightly-perturbed state for another nts time steps
% - integrate the Lorenz model for another nt time steps
%
% jsvs, October 2021
%
%

nt=[11 21 31 41 51 61 71 81 91 1e+2+1 5e+2+1 1e+3+1 5e+3+1 1e+4+1 5e+4+1 1e+5+1 5e+5+1
1e+6+1 5e+6+1 1e+7+1];
en=nt;
NN=length(nt);

nts0=5e+5;       % number of time steps needed to get an almost equilibrted Lorenz solution
nts=5e+3;        % number of time steps for spin up the Lorenz model

varn=0.1;        % variance of the random number used to generate an ensemble
dt=0.01;         % time stepping of the Lorenz model,
                 % dt=0.01 is used in the paper

v_en=zeros(3,NN);
v_nt=zeros(3,NN);

ofname='var.mat';

rng('default');

% get a nearly equilibrated state by integrating over nts0 time steps
x1=8.1184;
y1=10.5834;
z1=-1.7767;
w=lorz_n(x1,y1,z1,dt,nts0);
x0=w(nts0,1);
y0=w(nts0,2);
z0=w(nts0,3);

for i=1:NN

   disp(['for the time-loop, consider i=' num2str(i) ', length nt=' num2str(nt(i))])

   % spin up the lorenz model by running the model over nts timesteps from
   % a perturbed equilibrated state
   x1=x0+varn*randn(1,1);
   y1=y0+varn*randn(1,1);
   z1=-z0+varn*randn(1,1);
   w =lorz_n(x1,y1,z1,dt,nts);
   x1=w(nts,1);
   y1=w(nts,2);
   z1=w(nts,3);

   wk = lorz_n(x1,y1,z1,dt,nt(i));

   v_nt(:,i)=var(wk,1);

end
```

```
for i=1:NN

   disp(['for the ensemble-loop, consider i=' num2str(i)])
   wk=zeros(3,en(i));

   for in=1:en(i)

      if(fix(in/10000)*10000==in)
         disp(['        consider in=' num2str(in)])
      end

      % spin up the lorenz model by running the model over nts timesteps
      x1=x0+varn*randn(1,1);
      y1=y0+varn*randn(1,1);
      z1=z0+varn*randn(1,1);
      w =lorz_n(x1,y1,z1,dt,nts);
      x1=w(nts,1);
      y1=w(nts,2);
      z1=w(nts,3);

      wk(:,in) = lorz_n(x1,y1,z1,dt,1);
   end

   for icomp=1:3
      v_en(icomp,i)=var(wk(icomp,:),1);
   end
end

save(ofname,'v_nt','v_en','nt','en','x0','y0','z0')
plot_var


##########################################################################
###################    plot.var.m  (for Fig.3)                    ###################
##########################################################################


% plot the result of get_var
%

%clear
load('var.mat');
NN=length(nt);
fs=18;

% plot time-mean variance
figure
   semilogx(nt,v_nt(1,:),'g',nt,v_nt(2,:),'r',nt,v_nt(3,:),'b',
en,v_en(1,:),'g--',en,v_en(2,:),'r--',en,v_en(3,:),'b--','LineWidth',2)
   xlabel('2N/m')
   ylabel('var')
   xlim([0 nt(NN)])
   ylim([0 155])
   xticks([1e+1 1e+2 1e+3 1e+4 1e+5 1e+6])
   yticks([20 40 60 80 100 120 140])
   %text(1e+6,5,['component ' num2str(icp)],'FontSize',16)
   legend('comp. 1','comp. 2','comp. 3', 'FontSize',fs+1,'location','NorthEast')
   grid on
   ax = gca;
```

```
    ax.FontSize = fs;


##########################################################################
##################    get.c.m (for Fig.4)                    ##################
##########################################################################

% For the Lorenz model:
% Calculating auro-correlation function c of component x from a single
% equilibrium solution using time avearge <>:
% c(ilag)=<x'(1)*x'(ilag)>,
% where <> is the average over all possible time points, ' is the anoamly relative to <>, i.e.
% for each ensember member: x'(1)=x(1)-<x(1)>, x'(ilag)=x(ilag)-<x(ilag)>
%
% Equilibrium solutions are obtained by
% - starting from a  state (x1,y1,z1), which is a slightly-perturbed
%   equilibrium state (x0,y0,z0) obtained by integrating the Lorenz model over a long
%   time period.
% - integrate this slightly-perturbed state for another nts time steps
% - integrate the Lorenz model for another nt time steps
%
% jsvs, May 2022
%

clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

N=[1e+3 5e+3 5e+4 5e+5];
nlen=length(N);
lsave=true;

dl=500;          % for controling the loop ilag

varn=0.1;        % variance of the random number used to generate an initial condition
nts=5e+3;        % number of time steps for spin up the Lorenz model
dt=0.01;         % time stepping of the Lorenz model,  dt=0.01 is used in the paper

c=zeros(2*N(nlen)+1,3,nlen);

for ilen=1:nlen

    % spin up the lorenz model
    x1=x0+varn*randn(1,1);
    y1=y0+varn*randn(1,1);
    z1=z0+varn*randn(1,1);
    w = lorz_n(x1,y1,z1,dt,nts);
    x1=w(nts,1);
    y1=w(nts,2);
    z1=w(nts,3);

    % generating  a solution of length 2N+1 with N=N1, and calculating c
    % from the solution

    nt=2*N(ilen)+1;        % length of time series used to calculate r
    x = lorz_n(x1,y1,z1,dt,nt);
    x=x-mean(x);

    for icp=1:3
        %disp(['consider N=' num2str(N(ilen)) ',  icp=' num2str(icp)])
```

```
      for ilag=1:2*N(ilen)+1
         if(fix(ilag/dl)*dl==ilag)
            disp(['consider ilen=' num2str(ilen) ',  icp=' num2str(icp) ...
               ', lag=' num2str(ilag)])
         end

         il=ilag-1;              % il starts from lag=0
         for s=(-N(ilen)+il):N(ilen)
            r=  N(ilen)+1+s;          % changing indexing -N,...0...N to 1,...,2N+1
            c(ilag,icp,ilen)=c(ilag,icp,ilen)+x(r,icp)*x(r-il,icp);
         end
         c(ilag,icp,ilen)=c(ilag,icp,ilen)/(2*N(ilen)+1-il);
      end
   end

end

ofname='c_x.mat';
save(ofname,'c','N')
plot_c

##########################################################################
###################    plot.c.m (for Fig.4)                    ###################
##########################################################################

% plot autocorrelation function c of x obtained from get_c
% jsvs,  May, 2022

load('c_x.mat','c','N');

mN=2*N;
mN1=mN+1;
yl='c_{\tau,N}';
txt=["2N=2x10^3", "2N=1x10^4", "2N=1x10^5", "2N=1x10^6"];

figure('Position', [0 0 800 800])

fs=14;
ymax=60;
slag=1000;

ipt=0;

for i=1:4
   % plot c for the first 200+1 lags
   ipt=ipt+1;
   subplot(4,2,ipt);
   lag=0:slag;
   zz=lag*0;
   plot(lag,c(1:slag+1,1,i),'g',lag,c(1:slag+1,2,i),'r',lag,c(1:slag+1,3,i),'b',...
      lag,zz,'k')
   ylim([-ymax ymax])
   yticks([-60 -40 -20 0 20 40 60])
   ylabel(yl)
   if ipt==1
      legend('comp. 1','comp. 2','comp. 3')
   end
   xlabel('lag \tau')
   xticks([200 400 600 800 1000])
```

```
    text(200,50,txt(i),'FontSize',fs+1)
    grid on
    ax=gca;
    ax.FontSize = fs;

    % plot c1 for all lags
    ipt=ipt+1;
    subplot(4,2,ipt);
    lag=0:mN(i);
    zz=lag*0;
    plot(lag,c(1:mN1(i),1,i),'g',lag,c(1:mN1(i),2,i),'r',lag,c(1:mN1(i),3,i),'b',lag,zz,'k')
    ylim([-ymax ymax])
    yticks([-60 -40 -20 0 20 40 60])
    ylabel(yl)
    xlabel('lag \tau')
    grid on
    ax=gca;
    ax.FontSize = fs;

end


################################################################################
###################    get_gamma.m (for Fig.5)                  ###################
################################################################################

% For the Lorenz model:
% calculate autocorrelation function r of x at lag 0,1,...,nlag from
% 3 ensembles, each having en equilibrium solutions (r=gamma in the paper)
%
% Equilibrium solutions are obtained by
% - starting from a  state (x1,y1,z1), which is a slightly-perturbed
%   equilibrium state (x0,y0,z0) obtained by integrating the Lorenz model over a long
%   time period.
% - integrate this slightly-perturbed state for another nts time steps
% - integrate the Lorenz model for another nt time steps
%
% r(ilag)=<x'(1)*x'(ilag)>,
% where <> is ensemble mean, ' is the anoamly relative to <>, i.e.
% for each ensember member: x'(1)=x(1)-<x(1)>, x'(ilag)=x(ilag)-<x(ilag)>
% We have hence:
% r(lílag)=<x'(1)*x'(ilag)> = <x(1)*x(ilag)>-<x(1)>*<x(ilag)>
%
% jsvs, May 2022
%

clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

en=[1e+2 1e+4 1e+5 1e+6];          % ensemble size
nlen=length(en);
nlag=1e+5;        % maximum value of lag considered. nlag=1e+5 is used for dt=0.01

varn=0.1;        % variance of the random number used to generate an ensemble
nts=5e+3;

dt=0.01;         % time stepping of the Lorenz model,  dt=0.01 is used in the paper

ofname='gamma.mat';
```

```
r=zeros(nlag+1,3,nlen);
mx=zeros(nlag+1,3);  % ensemble mean of x at different times

rng(2)

dr=200;        % for tracking  the in-loop

for i=1:nlen

   for in=1:en(i)

      if(fix(in/dr)*dr == in)
         disp(['consider ensemble size =' num2str(en(i)) ', in=' num2str(in)])
      end

      % spin up the lorenz model
      x1=x0+varn*randn(1,1);
      y1=y0+varn*randn(1,1);
      z1=z0+varn*randn(1,1);
      wk = lorz_n(x1,y1,z1,dt,nts);
      x1=wk(nts,1);
      y1=wk(nts,2);
      z1=wk(nts,3);
      wkx = lorz_n(x1,y1,z1,dt,nlag+1);

      for ilag=1:nlag+1
         for ic=1:3
            r(ilag,ic,i)=r(ilag,ic,i)+wkx(1,ic)*wkx(ilag,ic);
            mx(ilag,ic)=mx(ilag,ic)+wkx(ilag,ic);
         end
      end

   end
   mx=mx/en(i);

   for ilag=1:nlag+1
      for ic=1:3
         r(ilag,ic,i)=r(ilag,ic,i)/en(i)-mx(1,ic)*mx(ilag,ic);
      end
   end

end

save(ofname,'r','en','nlag')

plot_gamma


##########################################################################
###################    plot_gamma.m (for Fig.5)                  ###################
##########################################################################


% plot autocorrelation function r  obtained from get_gamma
% Mai, 2022

load('gamma.mat');
txt='m=';
yl='\gamma_{\tau,m}';
```

```
txt=["m=1x10^2", "m=1x10^4", "m=1x10^5", "m=1x10^6"];

figure('Position', [0 0 800 800])

fs=13;
ymax=60;
slag=1000;

ipt=0;

for i=1:4

    % plot gamma for the first 200+1 lags
    ipt=ipt+1;
    subplot(4,2,ipt);
    lag=0:slag;
    zz=lag*0;
    plot(lag,r(1:slag+1,1,i),'g',lag,r(1:slag+1,2,i),'r',lag,r(1:slag+1,3,i),'b',...
        lag,zz,'k')
    ylim([-ymax ymax])
    yticks([-60 -40 -20 0 20 40 60])
    ylabel(yl)
    if ipt==1
        legend('comp. 1','comp. 2','comp. 3')
    end
    xlabel('lag \tau')
    xticks([200 400 600 800 1000])
    text(200,50,txt(i),'FontSize',fs+1)
    grid on
    ax=gca;
    ax.FontSize = fs;

    % plot c1 for all lags
    ipt=ipt+1;
    subplot(4,2,ipt);
    lag=0:nlag;
    zz=lag*0;
    plot(lag,r(1:nlag+1,1,i),'g',lag,r(1:nlag+1,2,i),'r',lag,r(1:nlag+1,3,i),'b',lag,zz,'k')
    ylim([-ymax ymax])
    yticks([-60 -40 -20 0 20 40 60])
    ylabel(yl)
    xlabel('lag \tau')
    grid on
    ax=gca;
    ax.FontSize = fs;

end



##########################################################################
###################    get_gamma_f.m (for Fig.6)                  ###################
##########################################################################


% For the Lorenz model:
% calculate autocorrelation function r of f at lag 0,1,...,nlag from
% one ensemble consiting en equilibrium solutions
% (r=gamma in the paper)
%
```

```
% Equilibrium solutions are obtained by
% - starting from a  state (x1,y1,z1), which is a slightly-perturbed
%   equilibrium state (x0,y0,z0) obtained by integrating the Lorenz model over a long
%   time period.
% - integrate this slightly-perturbed state for another nts time steps
% - integrate the Lorenz model for another nt time steps
%
% r(ilag)=<x'(1)*x'(ilag)>,
% where <> is ensemble mean, ' is the anoamly relative to <>, i.e.
% for each ensember member: x'(1)=x(1)-<x(1)>, x'(ilag)=x(ilag)-<x(ilag)>
% We have hence:
% r(lílag)=<x'(1)*x'(ilag)> = <x(1)*x(ilag)>-<x(1)>*<x(ilag)>
%
% jsvs, May 2022
%

clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

en=1e+6;          % ensemble size
nlag=5e+3;        % maximum value of lag considered. nlag=1e+5 is used for dt=0.01

varn=0.1;         % variance of the random number used to generate an ensemble
nts=5e+3;

dt=0.01;          % time stepping of the Lorenz model,  dt=0.01 is used in the paper

ofname='gamma_f.mat';

r=zeros(nlag+1,3);
mf=zeros(nlag+1,3);  % ensemble mean of x at different times

rng(2)

dr=1000;          % for tracking  the in-loop


for in=1:en
   if(fix(in/dr)*dr == in)
      disp(['consider ensemble size =' num2str(en) ', in=' num2str(in)])
   end

   % spin up the lorenz model
   x1=x0+varn*randn(1,1);
   y1=y0+varn*randn(1,1);
   z1=z0+varn*randn(1,1);
   wk = lorz_n(x1,y1,z1,dt,nts);
   x1=wk(nts,1);
   y1=wk(nts,2);
   z1=wk(nts,3);
   [wkx,wkf] = lorz_f_n_shift(x1,y1,z1,dt,nlag+1);

   for ilag=1:nlag+1
      for ic=1:3
         r(ilag,ic)=r(ilag,ic)+wkf(1,ic)*wkf(ilag,ic);
         mf(ilag,ic)=mf(ilag,ic)+wkf(ilag,ic);
      end
   end
```

```
end
mf=mf/en;

for ilag=1:nlag+1
   for ic=1:3
      r(ilag,ic)=r(ilag,ic)/en-mf(1,ic)*mf(ilag,ic);
   end
end

save(ofname,'r')



##########################################################################
###################    get_gamma_xf.m (for Fig.6)                  ###################
##########################################################################


% For the Lorenz model:
% estimate cross-correlation functions, rxf=gamma^{xf}, at lag -nlag,...-1,0,1,...,nlag
% from an ensemble with en equilibrium solutions
%
% The ensemble of size en is obtained in the same way as in get_gamma.m, i.e.
% - starting from a state (x1,y1,z1),  a slightly perturned equilibrated
%   state (x0,y0,z0)
% - integrate this state for another nts time steps
% - integrate for each member the Lorenz model for another nt time steps
%
% calculate the cross-correlation function rxf:
% rxf(lílag)=<x'(1)*f'(ilag)>,
% where <> is ensemble mean, ' is the anoamly relative to <>, i.e.
% for each ensember member: x'(1)=x(1)-<x(1)>, f'(ilag)=f(ilag)-<f(ilag)>
% We have hence:
% rxf(lílag)=<x'(1)*f'(ilag)> = <x(1)*f(ilag)>-<x(1)>*<f(ilag)>
%
% jsvs, May 2022
%

clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

ofname='gamma_xf';

en=1e+6;          % ensemble size, use  en=1000000 in paper
nlag=5e+3;        % maximum value of lag considered. nlag must smaller than nt
nt=2*nlag+1;       % length of time series used to calculate r

nts=5e+3;
varn=0.1;       % variance of the random number used to generate an ensemble

dt=0.01;        % time stepping of the Lorenz model, use dt=0.01 in paper

rxf=zeros(nt,3);
mx0=zeros(3,1);  % ensemble mean of x
mf0=zeros(nt,3);  % ensemble mean of f at different times

rng(2)

dr=1000;
```

```
for in=1:en

   if(fix(in/dr)*dr == in)
      disp(['consider in=' num2str(in)])
   end

   % spin up the lorenz model
   x1=x0+varn*randn(1,1);
   y1=y0+varn*randn(1,1);
   z1=z0+varn*randn(1,1);
   wk = lorz_n(x1,y1,z1,dt,nts);
   x1=wk(nts,1);
   y1=wk(nts,2);
   z1=wk(nts,3);
   [wkx,wkf] = lorz_f_n_shift(x1,y1,z1,dt,nt);

   for ic=1:3
      mx0(ic)=mx0(ic)+wkx(nlag+1,ic);
      for ilag=nlag:-1:1         % for negative lags
         i=nlag-ilag+1;
         rxf(i,ic)=rxf(i,ic)+wkx(nlag+1,ic)*wkf(nlag+1-ilag,ic);
         mf0(i,ic)=mf0(i,ic)+wkf(nlag+1-ilag,ic);
      end
      for ilag=0:nlag       % for zero and pos. lags
         i=nlag+ilag+1;
         rxf(i,ic)=rxf(i,ic)+wkx(nlag+1,ic)*wkf(nlag+1+ilag,ic);
         mf0(i,ic)=mf0(i,ic)+wkf(nlag+1+ilag,ic);
      end
   end

end

mx0=mx0/en;
mf0=mf0/en;
rxf=rxf/en;

for ic=1:3
   for ilag=nlag:-1:1
      i=nlag-ilag+1;
      rxf(i,ic)=rxf(i,ic)-mx0(ic)*mf0(nlag+1-ilag,ic);
   end
   for ilag=0:nlag
      i=nlag+ilag+1;
      rxf(i,ic)=rxf(i,ic)-mx0(ic)*mf0(nlag+1+ilag,ic);
   end
end

disp(['mx: ' num2str(mx0(1)) ', ' num2str(mx0(2)) ', ' num2str(mx0(3)) ', ' ])
disp(['mf: ' num2str(mf0(nlag,1)) ', ' num2str(mf0(nlag,2)) ', ' num2str(mf0(nlag,3)) ', ' ])

save(ofname,'rxf')
plot_gamma_xf


##########################################################################
###################    plot_gamma_xf.m (for Fig.6)                 ###################
##########################################################################


% Plot auto-covariance function of f (r) and cross-covariance function
```

```
% between x anf f (rxf)
% Mai, 2022

clear
load('gamma_f.mat','r');
load('gamma_xf.mat','rxf');
nlag_f=length(r(:,1))-1;
nlag_xf=(length(rxf(:,1))-1)/2;

fs=14;
ymax_f=2500;
ymax_xf=420;
slag=1000;

figure('Position', [0 0 800 400])

% plot auro-covariance function of f at short lags
subplot(2,2,1)
lag=0:slag;
zz=lag*0;
plot(lag,r(1:slag+1,1),'g',lag,r(1:slag+1,2),'r',lag,r(1:slag+1,3),'b',...
       lag,zz,'k')
ylim([-ymax_f ymax_f])
yticks([-2000 -1000 0 1000 2000])
xlim([0 slag])
xticks([200 400 600 800 1000])
ylabel('\gamma_{\tau,m}^f')
xlabel('lag \tau')
grid on
ax=gca;
ax.FontSize = fs;

% plot auro-covariance function of f at all lags
subplot(2,2,2);
lag=0:nlag_f;
zz=lag*0;
plot(lag,r(1:nlag_f+1,1),'g',lag,r(1:nlag_f+1,2),'r',lag,r(1:nlag_f+1,3),'b',lag,zz,'k')
ylim([-ymax_f ymax_f])
yticks([-2000 -1000 0 1000 2000])
xlim([0 nlag_f])
xticks([0 1000 2000 3000 4000 5000])
ylabel('\gamma_{\tau,m}^f')
xlabel('lag \tau')
legend('comp. 1','comp. 2','comp. 3')
grid on
ax=gca;
ax.FontSize = fs;

% plot cross-covariance function (x,f) at short lags
subplot(2,2,3)
lag=-nlag_xf:nlag_xf;          % lags running from -nlag to nlag
lm=  nlag_xf+1;             % center point corresponding to lag zero
lL=200;                  % plot the center piece of the cross-correlation
lf=lm-lL;
ll=lm+lL;
zz=lag(lf:ll)*0;
plot(lag(lf:ll), rxf(lf:ll,1), 'g', ...
    lag(lf:ll), rxf(lf:ll,2), 'r', lag(lf:ll), rxf(lf:ll,3), 'b', ...
    lag(lf:ll), zz, 'k')
grid on
```

```
ylim([-420 420])
yticks([-400 -200 0 200 400])
xlim([-lL lL])
xticks([-160 -120 -80 -40 0 40 80 120 160])
xlabel('lag \tau')
ylabel('\gamma_\tau^{xf}')
ax = gca;
ax.FontSize = fs;

% plot cross correlation at all lags
subplot(2,2,4)
lf=lm-nlag_xf;
ll=lm+nlag_xf;
zz=lag(lf:ll)*0;
plot(lag(lf:ll), rxf(lf:ll,1), 'g', ...
    lag(lf:ll), rxf(lf:ll,2), 'r', lag(lf:ll), rxf(lf:ll,3), 'b', ...
    lag(lf:ll), zz, 'k')
grid on
xlabel('lag \tau')
ylabel('\gamma_\tau^{fx}')
xlim([-nlag_xf nlag_xf])
xticks([-4000 -2000 0 2000 4000])
ylim([-420 420])
yticks([-400 -200 0 200 400])
%ylim([-0.1 0.1])
%legend('comp. 1','comp. 2','comp. 3')
grid on
ax = gca;
ax.FontSize = fs;

##########################################################################
####################    get_spxf_shift.m (for Fig.7)                ######################
##########################################################################


% From one ensemble of Lorenz equilibrium solutions, get:
%
% mspxf:  ensemble averaged cross spectrum between x and f
% mspxx:  ensemble averaged spectrum of x
% mspff:  ensemble averaged spectrum of f
% coxf:   co-spectrum = real part of mspxf
% cxf/pxf:    ensemble averaged coherence/phase spectrum
%
% The Lorenz solution is obtained by
% - starting from a state (x1,y1,z1),  which is a perturbed equilibrated state
% - integrate this state for another nts time steps, where nts is randomly
%   chosen
% - integrate the Lorenz model for another nt time steps
%
% en: ensemble size
%
% jsvs, Mai 2022
%

clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

check_var=true;   % if check_var: compare the total variance with the integral of the spectrum
```

```
en=1000;          % ensemble size
den=20;
nt=1e+7+1;       % length of time series considered for spectra
nts=5e+3;
varn=0.1;        % variance of the random number used to generate
                 % an ensemble

ns=fix(nt/2);
dt=0.01;         % time stepping of the Lorenz model,
                 % dt=0.01 is used in the paper

ofname='sp_xf_shift.mat';

wx=zeros(nt,1);  % delta x
wf=zeros(nt,1);  % forcing of delta x

ftx=zeros(nt,1);  % Fourier transform of x
ftf=zeros(nt,1);  % Fourier transform of f

spxf=zeros(ns,1);
spxx=zeros(ns,1);
spff=zeros(ns,1);

mspxf=zeros(ns,3);  % mean spectrum of x
mspxx=zeros(ns,3);  % mean spectrum of x
mspff=zeros(ns,3);  % mean spectrum of f
cxf=zeros(ns,3);    % estimated coherence spectrum
pxf=zeros(ns,3);    % estimated phase spectrum
coxf=zeros(ns,3);  % co-spectrum between f and x

rng('default');

% consider each member of an ensemble of size en
for in=1:en

   if(fix(in/den)*den==in)
      disp(['consider in=' num2str(in)])
   end

   % spin up the lorenz model
   x1=x0+varn*randn(1,1);
   y1=y0+varn*randn(1,1);
   z1=z0+varn*randn(1,1);
   wk=lorz_n(x1,y1,z1,dt,nts);
   x1=wk(nts,1);
   y1=wk(nts,2);
   z1=wk(nts,3);

   [wkx,wkf] = lorz_f_n_shift(x1,y1,z1,dt,nt);
   %remove the time mean and time standard deviation

   wkx=wkx-mean(wkx);
   wkf=wkf-mean(wkf);

   for icomp=1:3

      wx=wkx(:,icomp);
      wf=wkf(:,icomp);

      ftx=fft(wx);
```

```
        ftf=fft(wf);

        for i=1:ns
            spxf(i)=2*ftx(i+1)*conj(ftf(i+1))/nt;
            spxx(i)=2*ftx(i+1)*conj(ftx(i+1))/nt;
            spff(i)=2*ftf(i+1)*conj(ftf(i+1))/nt;
        end

        mspxf(:,icomp)=mspxf(:,icomp)+spxf;
        mspxx(:,icomp)=mspxx(:,icomp)+spxx;
        mspff(:,icomp)=mspff(:,icomp)+spff;

    end

end

mspxf=mspxf/en;
mspxx=mspxx/en;
mspff=mspff/en;
coxf=real(mspxf);


if check_var
    vx=var(wkx);
    vf=var(wkf);
    sum_spxx=sum(mspxx)/nt;
    sum_spff=sum(mspff)/nt;
    disp('compare total variance with the integral of spectrum:')
    disp(['var(x)=' num2str(vx(1)) ',' num2str(vx(2)) ',' num2str(vx(3))])
    disp(['sum(spxx)=' num2str(sum_spxx(1)) ',' num2str(sum_spxx(2)) ',' num2str(sum_spxx(3))])
    disp(['var(f)=' num2str(vf(1)) ',' num2str(vf(2)) ',' num2str(vf(3))])
    disp(['sum(spff)=' num2str(sum_spff(1)) ',' num2str(sum_spff(2)) ',' num2str(sum_spff(3))])
end

cxf=(abs(mspxf).^2)./(mspxx.*mspff);
for icomp=1:3
    for i=1:ns
        pxf(i,icomp)=atan2(imag(mspxf(i,icomp)),real(mspxf(i,icomp)))*180/pi;
    end
end

f=(1:ns)'/(nt-1);

save(ofname,'mspxf','mspxx','mspff','coxf','cxf','pxf', 'f', 'nt','-v7.3')

plot_sp_1
plot_spxf




##########################################################################
###################    get_white_lf_var.m (for Fig.7)                ###################
##########################################################################


% using the output of get_spxf to
%
% 1) identify the frequency om at which the specrum of x, mspxx, deviates
%    about p*100% from spx0,
```

```
%    spx0 = mean(mspxx(1:nw,:)), with nw being defined below
%
%    The frequency at which this happens is om=f(i0), where f is the
%    frequency outputted by get_spxf
%
%    Since mspxx reveals still some variations, the spectrum
%    is first running averaged with a spectrum window having length
%    2*nw+1.
%
% 2) get the variance of x in the frequency range [0,om], vx and vx0.
%    vx is estimated using running averaged ratio, vx0 is estimated using
%    raw ratio
%
% jsvs,  June, 2022
%

clear
load('sp_xf_shift.mat');

nw=20;       % half of the running average window
spx0=mean(mspxx(1:2*nw+1,:));

p=0.05;         % thredhold for defining omega_m=omm,  using ratio=sph/spf

ns=length(mspxx(:,1));
nt=2*ns+1;
rmx=zeros(ns,3);          % running averaged spx

iomr=zeros(3,1);    % the index of the frequency f at which r=p*mspxx(1,:)
vx=zeros(3,1);      % using running averaged data

for icomp=1:3
   disp(['consider comp=' num2str(icomp)])
   for i=nw+1:ns-nw
      rmx(i,icomp)=mean(mspxx(i-nw:i+nw,icomp));
   end
end

for icomp=1:3
   for i=nw+1:ns-nw
      if (rmx(i,icomp)<spx0(icomp)*(1-p)) || (rmx(i,icomp)>spx0(icomp)*(1+p))
         iomr(icomp)=i;
         break
      end
   end
   vx(icomp)=sum(mspxx(1:iomr(icomp)))/nt;
end

disp(['variance of the white low-frequency plateau' num2str(vx(1)) ', ' num2str(vx(2)) ', '
num2str(vx(3))])
vtot=sum(mspxx)/nt;
vx=(vx'./vtot)*100;
disp(['percentages =' num2str(vx(1)) ', ' num2str(vx(2)) ', ' num2str(vx(3))])
disp(['frequency omega_*=' num2str(f(iomr(1))) ', ' num2str(f(iomr(2))) ', ' num2str(f(iomr(3)))])

save(,omega.mat','iomr','vx')
```

```
##############################################################################
###################    plot_spxf.m (for Fig.7)                  ##################
##############################################################################


% plot spectra of f and x

load('sp_xf_shift.mat');
load('omega.mat','iomr')

wf=mspff;
f2=f.*f*10e8;
fk2=1.e-10/(f.*f);

% removed the phase jump from -180 to 180
ns=length(pxf(:,1));
for icomp=1:3
   for i=1:ns
      if abs(imag(mspxf(i,icomp))) <0.1
         %if real(mspxf(i,icomp))>0
            %pxf(i,icomp)=0;
         %else
            pxf(i,icomp)=-180;
         %end
      elseif abs(real(mspxf(i,icomp))) <0.1
         if imag(mspxf(i,icomp)) > 0
            pxf(i,icomp)=90;
         else
            pxf(i,icomp)=-90;
         end
      else
         pxf(i,icomp)=atan2(imag(mspxf(i,icomp)),real(mspxf(i,icomp)))*180/pi;
      end
   end
end
%for ic=1:3
%    for i=1:ns
%       if(pxf(i,ic)>179) && (pxf(i,ic)<180.1)
%          pxf(i,ic)=-pxf(i,ic);
%       end
%    end
%end

%f2=f.*f*10e8;
xlim1=1/(nt-1);
fs=14;
cc=['g' 'r' 'b'];

figure('Position', [0 0 900 500])

subplot(3,2,[1 3])
loglog(f,mspxx(:,3),'b',f,mspxx(:,2),'r',f,mspxx(:,1),'g',f,fk2,'k')
for icomp=1:3
   xline(f(iomr(icomp)),cc(icomp));
end
xlim([xlim1 1])
%xline(1.42e-2,'k');
%xline(2e-2,'k');
ylim([1e-6 1e+5])
yticks([1e-6 1e-4 1e-2 1e+0 1e+2 1e+4])
```

```
xticks([1e-7 1e-5 1e-3 1e-1])
ylabel('spectrum x')
text(8e-7,4e+5,'a)','FontSize',fs+2)
legend('comp. 3','comp. 2','comp. 1', 'FontSize',fs+1,'location','SouthWest')
grid on
ax = gca;
ax.FontSize = fs;

subplot(3,2,[2 4])
loglog(f,wf(:,1),'g',f,wf(:,2),'r',f,wf(:,3),'b',f,f2,'k')
hold on
%xline(1.42e-2,'k');
%xline(2e-2,'k');
xlim([xlim1 1])
xticks([1e-7 1e-5 1e-3 1e-1])
ylim([1e-4 1e+7])
yticks([1e-3 1e-1 1e+1 1e+3 1e+5 1e+7])
ylabel('spectrum f')
text(8e-7,4e+7,'b)','FontSize',fs+2)
text(8e-7,1e-2,'\omega^2','FontSize',fs+2)
grid on
ax = gca;
ax.FontSize = fs;

subplot(3,2,5)
semilogx(f,cxf(:,1),'g',f,cxf(:,2),'r',f,cxf(:,3),'b')
ylabel('coherence (x,f)')
xlabel('frequency')
yticks([0 0.2 0.4 0.6 0.8 1])
xticks([1e-7 1e-5 1e-3 1e-1])
ylim([0 1.1])
xlim([xlim1 1])
text(5e-7,1.25,'c)','FontSize',fs+2)
grid on
ax = gca;
ax.FontSize = fs;
%legend('comp. 1','comp. 2','comp. 3','FontSize',16,'location','SouthWest')

subplot(3,2,6)
semilogx(f,pxf(:,1),'g',f,pxf(:,2),'r',f,pxf(:,3),'b')
xlim([xlim1 1])
ylim([-160 50])
yticks([-135 -90 -45 0 45])
xticks([1e-7 1e-5 1e-3 1e-1])
ylabel('phase (x,f)')
xlabel('frequency')
%legend('comp. 1','comp. 2','comp. 3','FontSize',16,'location','SouthEast')
text(5e-7,75,'d)','FontSize',fs+2)
grid on
ax = gca;
ax.FontSize = fs;
```

```
##########################################################################
###################    plot_spf_linear.m (for Fig.8)                ###################
##########################################################################

% plot spectra of f and x
```

```
load('sp_xf_shift.mat');

wx=abs(mspxf);
wf=mspff;

xlim1=1/(nt-1);
fs=20;
cc=['g' 'r' 'b'];

figure('Position', [0 0 1000 400])
subplot(1,2,1)
semilogx(f,wf(:,1),'g',f,wf(:,2),'r',f,wf(:,3),'b')
hold on
%xline(1.42e-2,'k','LineWidth',1);
%xline(2e-2,'k','LineWidth',1);
ylabel('spectrum f')
ylim([0,1e+6])
xlim([1e-7,1e+0])
xticks([1e-6 1e-4 1e-2 1e0])
text(0.08e-7,1e+6,'a)','FontSize',fs+2)
legend('comp. 1','comp. 2','comp. 3', 'FontSize',fs+1,'location','NorthWest')
grid on
ax = gca;
ax.FontSize = fs;

subplot(1,2,2)
semilogx(f,wx(:,1),'g',f,wx(:,2),'r',f,wx(:,3),'b')
hold on
ylim([0,1e+5])
%xline(1.42e-2,'k');
%xline(2e-2,'k');
xlim([1e-7,1e+0])
xticks([1e-6 1e-4 1e-2 1e0])
text(0.08e-7,1e+5,'b)','FontSize',fs+2)
ylabel('amplitude (x,f)')
grid on
%legend('comp. 1','comp. 2','comp. 3', 'FontSize',fs+1,'location','NorthWest')
ax = gca;
ax.FontSize = fs;

##########################################################################
###################    get_signal.m (for Fig.9)                  ###################
##########################################################################


% Calculate the spectrum of an impulse-like signal x of length 2N+1
% x is a function of time that is not zero for s in [-n0, n0] and zero
% outside the interval [-n0,n0].
%
% The values inside [-n0,n0] correspond to an AR1-process with coefficient
% a and initial condition x0
%
% The spectrum is an average over en sample spectra,  each being obtained
% from one realisation of the signal
%
% jsvs, July 2022
%
```

```
en=100;
n=5e+6;
nn0=[5 50];
in0=length(nn0);
ofname=['sp_signal.mat'];

ns=n;

wsp=zeros(ns,1);
msp=zeros(ns,in0);

rng('default');
a=0.9;
%x0=0.5;

for i0=1:in0
   n0=nn0(i0);
   x=zeros(2*n+1,1);

   for in=1:en
      x0=randn(1,1);

      disp(['consider i0=' num2str(i0) ', in=' num2str(in)])
      x(n-n0:n+n0)=ar1(x0,a,2*n0+1);

      ftx=fft(x);

      for is=1:ns
         wsp(is)=2*ftx(is+1)*conj(ftx(is+1))/(2*n+1);
      end

      msp(:,i0)=msp(:,i0)+wsp;
   end

end

msp=msp/en;


f=(1:ns)'/(2*ns);

save(ofname,'msp','f','ns','en','nn0','-v7.3')

plot_signal


##########################################################################
###################    ar1.m (for Fig.9)                    ###################
##########################################################################

function [ x ] = ar1(x0,alpha,nt)
% generate a time series of length nt from an  AR1-process with parameter alpha

x=zeros(1,nt);
x(1)=x0;
for i=2:nt
   x(i)=x(i-1)*alpha+randn(1,1);
end
end
```

```
##########################################################################
###################    plot_signal.m (for Fig.9)                  ###################
##########################################################################


clear
load('sp_signal.mat','msp','f','ns');
nn0=[5 50];
in0=length(nn0);
p=0.01;
iomr=zeros(in0,1);

for i0=1:in0
   for i=1:ns
      if abs(msp(i,i0)-msp(1,i0)) > p*msp(1,i0)
         iomr(i0)=i;
         break
      end
   end
end

fs=22;
xlim1=1/(2*ns);
f2=1e-11./(f.*f);

figure('Position', [0 0 800 700])
loglog(f,msp(:,1),'r',f,msp(:,2),'b','LineWidth',1.5)
xline(f(iomr(1)),'r');
xline(f(iomr(2)),'b');
xlim([xlim1 1])
xticks([1e-6 1e-4 1e-2 1e+0])
ylim([5e-7 5e-3])
yticks([1e-6 1e-5 1e-4 1e-3 1e-2])
legend(['n=' num2str(2*nn0(1)+1)],['n='
num2str(2*nn0(2)+1)],'FontSize',fs+2,'location','SouthWest')
grid on
ax = gca;
ax.FontSize = fs;


##########################################################################
###################    get_sp_m.m (for Fig.B1)                  ###################
##########################################################################


% For n=3 ensembles of Lorenz equilibrium solutions, get for each ensemble
% the ensemble averaged spectrum of x, mspxx.
%
% The Lorenz solution is obtained by
% - starting from a state (x1,y1,z1), a perturbed equilibrated state
% - integrate this state for another nts time steps
% - integrate the Lorenz model for another nt time steps
%
% Three sample sizes, 1, en1=10 and en2=100,  are considered to
% check the convergence of ensemble averaged spectra with increasing
% ensemble size
%
% jsvs, May 2022
%
```

```
clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

en=[1 10 100];    % ensemble size
nen=length(en);


nt=1e+7+1;       % length of time series considered for spectra
nts=5e+3;
varn=0.1;        % variance of the random number used to generate
                 % an ensemble

ns=fix(nt/2);
dt=0.01;         % time stepping of the Lorenz model,
                 % dt=0.01 is used in the paper

ofname='sp_x_m.mat';

wkx=zeros(nt,1);
spxx=zeros(ns,1);
mspxx=zeros(ns,3,nen);  % mean spectrum of x

rng('default');

% consider 2 ensembles
for ien=1:nen

   % consider each member of an ensemble of size en
   for in=1:en(ien)

      disp(['consider ien=' num2str(ien) ', in=' num2str(in)])

      % spin up the lorenz model by running the model over nts timesteps
      x1=x0+varn*randn(1,1);
      y1=y0+varn*randn(1,1);
      z1=z0+varn*randn(1,1);
      wk=lorz_n(x1,y1,z1,dt,nts);
      x1=wk(nts,1);
      y1=wk(nts,2);
      z1=wk(nts,3);

      wkx = lorz_n(x1,y1,z1,dt,nt);
      %remove the time mean
      wkx=wkx-mean(wkx);

      for icomp=1:3

         ftx=fft(wkx(:,icomp));

         for i=1:ns
            spxx(i)=2*ftx(i+1)*conj(ftx(i+1))/nt;
         end
         mspxx(:,icomp,ien)=mspxx(:,icomp,ien)+spxx;

      end

   end
```

```
end

for ien=1:nen
   mspxx(:,:,ien)=mspxx(:,:,ien)/en(ien);
end

f=(1:ns)'/(nt-1);

save(ofname,'mspxx', 'f', 'nt','en','-v7.3')

plot_sp_m

##########################################################################
###################    plot_sp_m.m (for Fig.B1)                  ###################
##########################################################################


% plot spectra derived from ensembles of different sizes
%
%

%clear

load('sp_x_m.mat');
wa=mspxx;
load('sp_xf_shift.mat');
wb=mspxx;
fs=f;

ns=length(f);
xlim1=1/1e+7;

tt1=['size=' num2str(en(1))];
tt2=['size=' num2str(en(2))];
tt3=['size=' num2str(en(3))];
tt4='size=1000';

for icomp=1:1   % consider only the first Lorenz component

   figure('Position', [0 0 600 500])
   loglog(f,wa(:,icomp,1),'c',f,wa(:,icomp,2),'r',f,wa(:,icomp,3),'w',f,wb(:,icomp),'k','LineWidth',1)
   xlim([xlim1 1])
   ylim([1e-5 1e+5])
   yticks([1e-4 1e-2 1e-0 1e+2 1e+4])
   xlabel('frequency')
   ylabel('spectrum')
   grid on
   ax = gca;
   ax.FontSize = 18;

end


##########################################################################
###################    get_sp_n.m (for Fig.B2)                   ###################
##########################################################################



% Calculate the spectrum of x using ensembles containing en=1000
```

```
% equilibrium solutions, with the length of the solution differs from
% ensemble to ensemble and equals
% 1e+3+1, 1e+4+1, 1e+5+1, 1e+6+1, respectively
%
% The equilibrium solutions are obtained by
% - starting from a state (x1,y1,z1),  a perturbed equilibrated state
% - integrate this state for another nts time steps, where nts is randomly
%   chosen
% - integrate the Lorenz model for another nt time steps
%
% jsvs, May 2022
%

clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

en=1000;
nt=[1e+3+1 1e+4+1 1e+5+1 1e+6+1];    % ensemble size
NT=length(nt);
ns=fix(nt./2);

nts=5e+3;

varn=0.1;        % variance of the random number used to generate an ensemble
dt=0.01;         % time stepping of the Lorenz model,
                 % dt=0.01 is used in the paper

ofname='sp_x_n.mat';

ns0=max(ns);
msp=zeros(ns0,3,NT);

rng('default');

for i=1:NT

    wsp=zeros(nt(i),1);

    disp(['consider i=' num2str(i) ', length nt=' num2str(nt(i))])

    for in=1:en

        if(fix(in/200)*200==in)
            disp(['        consider in=' num2str(in)])
        end

        % spin up the lorenz model by running the model over nts timesteps
        x1=x0+varn*randn(1,1);
        y1=y0+varn*randn(1,1);
        z1=z0+varn*randn(1,1);
        wk=lorz_n(x1,y1,z1,dt,nts);
        x1=wk(nts,1);
        y1=wk(nts,2);
        z1=wk(nts,3);

        wkx = lorz_n(x1,y1,z1,dt,nt(i));
        %remove the time mean
        wkx=wkx-mean(wkx);
```

```
        for icomp=1:3

            ftx=fft(wkx(:,icomp));

            for is=1:ns(i)
                wsp(is)=2*ftx(is+1)*conj(ftx(is+1))/nt(i);
            end

            msp(1:ns(i),icomp,i)=msp(1:ns(i),icomp,i)+wsp(1:ns(i));

        end

    end

end

msp=msp/en;

save(ofname,'msp','nt','en','-v7.3')

plot_sp_n
################################################################################
###################    plot_sp_n.m (for Fig.B2)                  ###################
################################################################################

% plot spectra obtained from get_msp_n


%clear
load('sp_x_n.mat');
icomp=1;

NT=length(nt);
ns=fix(nt./2);
xlim1=1/(nt(NT)-1);
fs=14;
txt=["2N=1x10^3", "2N=1x10^4", "2N=1x10^5", "2N=1x10^6"];

figure('Position', [0 0 800 700])

isp=0;
for i=1:NT
    isp=isp+1;
    subplot(2,2,isp)
    f=(1:ns(i))'/(nt(i)-1);
    wsp=nonzeros(msp(1:ns(i),icomp,i));
    loglog(f,wsp,'r','LineWidth',1.5)
    xlim([xlim1 1])
    xticks([1e-6 1e-4 1e-2 1e+0])
    ylim([1.5e-5 1e+5])
    yticks([1e-4 1e-2 1e-0 1e+2 1e+4])
    xlabel('frequency')
    ylabel('E_m(S_{\omega,N})')
    text(2e-6,1e-4,txt(isp),'FontSize',fs+2)
    grid on
    ax = gca;
    ax.FontSize = fs;
end
```

```
##########################################################################
##################    get_sp_n_plus1.m (for Fig.C1)                ##################
##########################################################################


%  Get from one solution x_s two pieces of length nt1 and nt2=nt1+2.
%  The first piece start from s=1, the second from s=nt1+1.
%  Calculate the sample spectra from the two pieces and calculate the difference
%
% The Lorenz solution is obtained by
% - starting from a state (x1,y1,z1),  a perturbed equilibrated state
% - integrate this state for another nts time steps
% - integrate the Lorenz model for another nt time steps
%
% jsvs, May 2022
%

clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

dn=2;

nt1=[1e+5+1 1e+6+1 1e+7+1 1e+8+1];    % length of the first piece
nt2=nt1+dn;    % length of the second piece = nt1+2

NT=length(nt1);
ns1=fix(nt1./2);
ns2=fix(nt2./2);

nts=5e+3;
varn=0.1;       % variance of the random number used to generate an ensemble
dt=0.01;        % time stepping of the Lorenz model,
              % dt=0.01 is used in the paper

ofname='sp_x_n_plus1.mat';
%ofname='sp_x_n_plus1_different_solution.mat';

ns0=max(ns1);
sp1=zeros(ns0,3,NT);
ns0=max(ns2);
sp2=zeros(ns0,3,NT);

rng('default');

for i=1:NT

   wk1=zeros(nt1(i),1);
   wk2=zeros(nt2(i),1);

   disp(['consider i=' num2str(i) ', length nt1=' num2str(nt1(i)) ', length nt2=' num2str(nt2(i))])

   % spin up the lorenz model by running the model over nts timesteps
   x1=x0+varn*randn(1,1);
   y1=y0+varn*randn(1,1);
   z1=z0+varn*randn(1,1);
   wk=lorz_n(x1,y1,z1,dt,nts);
   x1=wk(nts,1);
   y1=wk(nts,2);
   z1=wk(nts,3);
```

```
    wk1 = lorz_n(x1,y1,z1,dt,nt1(i));

    x1=wk1(nt1(i),1);
    y1=wk1(nt1(i),2);
    z1=wk1(nt1(i),3);
    % get the second piece from a different solution
    %x1=x0+varn*randn(1,1);
    %y1=y0+varn*randn(1,1);
    %z1=z0+varn*randn(1,1);
    %wk=lorz_n(x1,y1,z1,dt,nts);
    %x1=wk(nts,1);
    %y1=wk(nts,2);
    %z1=wk(nts,3);
    wk2 = lorz_n(x1,y1,z1,dt,nt2(i));

    wk1=wk1-mean(wk1);
    wk2=wk2-mean(wk2);

    for icomp=1:3

        ftx=fft(wk1(:,icomp));
        for is=1:ns1(i)
            sp1(is,icomp,i)=2*ftx(is+1)*conj(ftx(is+1))/nt1(i);
        end
        ftx=fft(wk2(:,icomp));
        for is=1:ns2(i)
            sp2(is,icomp,i)=2*ftx(is+1)*conj(ftx(is+1))/nt2(i);
        end

    end

end

save(ofname,'sp1','sp2','nt1','nt2', '-v7.3')

plot_sp_n_plus1


##########################################################################
##################    plot_sp_n_plus1.m (for Fig.C1)                ##################
##########################################################################

% plot the sample spectra obtained from get_sp_n_plus1


clear
load('sp_x_n_plus1.mat');
%load('sp_x_n_plus1_different_solution.mat');
icomp=1;

txt=["2N=1x10^5", "2N=1x10^6", "2N=1x10^7", "2N=1x10^8"];

NT=length(nt1);
ns1=fix(nt1./2);
ns2=fix(nt2./2);
xlim1=1/(nt1(NT)-1);
fs=14;

figure('Position', [0 0 800 700])
```

```
isp=0;
for i=1:NT
   isp=isp+1;
   subplot(2,2,isp)
   f1=(1:ns1(i))'/(nt1(i)-1);
   f2=(1:ns2(i))'/(nt2(i)-1);
   wsp2=nonzeros(sp2(1:ns1(i),icomp,i));
   wsp1=nonzeros(sp1(1:ns1(i),icomp,i));
   wsp=wsp2-wsp1;
   semilogx(f1,wsp,'r')
   hold on
   xlim([xlim1 1])
   xticks([1e-8 1e-6 1e-4 1e-2 1e+0])
   ylim([-1.5e+5 1.5e+5])
   xlabel('frequency')
   ylabel('S_{N+1} - S_{N}')
   text(2e-8,-1e5,txt(isp),'FontSize',fs+2)
   grid on
   ax = gca;
   ax.FontSize = fs;
   hold off
end
```

```
##########################################################################
###################    get_fc.m (for Fig.C2)                 ###################
##########################################################################
```

```
% Given an ensemble of size en containing equilibrium solutions of length
% T (T being odd), and a  frequency omega=j/(T-1):
% calculate the Fourier coefficients at frequency j/(T-1) from each ensemble member
%
% jsvs, May, 2022
%

clear
load('var.mat');
disp([' initial condition used for spin-up = (' num2str(x0) ',' num2str(y0) ',' num2str(z0) ')'])

en=100;          % ensemble size
T=1e+7+1;        % length of time series considered for calculating the
                 % Fourier coefficient
omega=0.04;    %5/(T-1);

nts=5e+3;
varn=0.1;        % variance of the random number used to generate
                 % an ensemble
dt=0.01;         % time stepping of the Lorenz model

ofname=['fc_' num2str(omega) '.mat'];

Fx_r=zeros(en,3);
Fx_i=zeros(en,3);

cs=zeros(T,1);
sn=zeros(T,1);
```

```
for t=1:T
   cs(t)=cos(2*pi*omega*t)/T;
   sn(t)=-sin(2*pi*omega*t)/T;
end

rng('default');

for in=1:en

   disp(['consider in=' num2str(in)])

   % spin up the lorenz model
   x1=x0+varn*randn(1,1);
   y1=y0+varn*randn(1,1);
   z1=z0+varn*randn(1,1);
   wk = lorz_n(x1,y1,z1,dt,nts);
   x1=wk(nts,1);
   y1=wk(nts,2);
   z1=wk(nts,3);

   wkx = lorz_n(x1,y1,z1,dt,T);
   wkx=wkx-mean(wkx);

   for t=1:T
      for ic=1:3
         Fx_r(in,ic)=Fx_r(in,ic)+wkx(t,ic)*cs(t);
         Fx_i(in,ic)=Fx_i(in,ic)-wkx(t,ic)*sn(t);
      end
   end

end

save(ofname,'Fx_r','Fx_i')
plot_fc



##########################################################################
###################    plot_fc.m (for Fig.C2)                    ###################
##########################################################################



% plot output of get_fc
%

ofname=['fc_' num2str(omega) '.mat'];
load(ofname,'Fx_r','Fx_i');
xmin1=min(Fx_r(:,1));
ymax1=max(Fx_i(:,1));

tt1=['comp. 1, \omega=' num2str(omega)];

figure
scatter(Fx_r(:,1),Fx_i(:,1),50,'o')
hold on
xline(0);
yline(0);
xlabel('real')
ylabel('imag')
```

```
xlim([-0.01 0.01])
ylim([-0.01 0.01])
%xlim([-0.05 0.05])
%ylim([-0.05 0.05])
%xticks([-0.04 -0.02 0 0.02 0.04])
%yticks([-0.04 -0.02 0 0.02 0.04])
grid on
ax = gca;
ax.FontSize = 18;
hold off
```