Causal Dynamic Resonance

Claudia Lainscsek

This software reproduces the results in the paper "Causal Dynamic Resonance", submitted to PNAS and can be found on Github (`https://github.com/lclaudia/CDR`). The code is written in JULIA and works on Linux, Mac, and Windows. For the epilepsy part we chose to include a shorter data file due to size and computation time. The Rössler part is written as a sequential code (calling `julia`) and for the epilepsy part we use parallel capabilities of JULIA (calling `julia -p`n, where n are the parallel nodes to be used).

The underlying C codes are compiled using `cosmocc` from `https://github.com/jart/cosmopolitan`.

LINUX: If you get the error "run-detectors: unable to find an interpreter", you can fix that by running these commands:
(see `https://github.com/jart/cosmopolitan/blob/master/tool/cosmocc/README.md` for more details).

```
sudo wget -O /usr/bin/ape https://cosmo.zip/pub/cosmos/bin/ape-$(uname -m).elf
sudo chmod +x /usr/bin/ape
sudo sh -c "echo ':APE:M::MZqFpD::/usr/bin/ape:' >/proc/sys/fs/binfmt_misc/register"
sudo sh -c "echo ':APE-jart:M::jartsr::/usr/bin/ape:' >/proc/sys/fs/binfmt_misc/register"
```

WINDOWS: Microsoft might be seeing `run_DDA_ASCII.exe` as a virus and is deleting it. To fix this problem, turn the "Real-time protection" (temporarily) off to execute the codes.

1 Single Rössler system

Before introducing coupled Rössler systems the code to integrate a single system is presented. The equations for the Rössler system are

$$\begin{aligned} \dot{u}_1 &= -u_2 - u_3 \\ \dot{u}_2 &= u_1 + a\,u_2 \\ \dot{u}_3 &= b - c\,u_3 + u_1 u_3 \end{aligned} \tag{1}$$

with $a = 0.2$ and $c = 5.7$ and $\delta t = 0.05$. This system can be encoded as

system	equation #	variable		coefficients
$\dot{u}_1 = -u_2 - u_3$	0	0	2	-1
$\dot{u}_1 = -u_2 - u_3$	0	0	3	-1
$\dot{u}_2 = u_1 + a\,u_2$	1	0	1	1
$\dot{u}_2 = u_1 + a\,u_2$	1	0	2	a
$\dot{u}_3 = b - c\,u_3 + u_1\,u_3$	2	0	0	b
$\dot{u}_3 = b - c\,u_3 + u_1\,u_3$	2	0	3	$-c$
$\dot{u}_3 = b - c\,u_3 + u_1\,u_3$	2	1	3	1

Note, that the equation numbers are (0,1,2) for the three equations. This defines `DIM=3` (the number of equations). There are two "variable" columns which define the order of nonlinearity `ODEorder=2`. The numbers in the two columns are 1 for u_1, 2 for u_2, and 3 for u_3. A line with only zeros denotes a constant term. All other entries are filled with zeros.

This encoding can be used to numerically integrate the Rössler system. The plots are shown in Fig. 1.

```
include("DDAfunctions.jl");                                    # set of Julia functions

NrSyst=1;                                                      # 1 single system
ROS=[[0  0 2];                                                 # single Roessler system
     [0  0 3];
     [1  0 1];
     [1  0 2];
     [2  0 0];
     [2  0 3];
     [2  1 3]
    ];
 (MOD_nr,DIM,ODEorder,P) = make_MOD_nr(ROS,NrSyst);             # encoding of the Roessler system
                                                               # function defined in DDAfunctions.jl
a=.2; c=5.7;
dt=.05; X0=rand(DIM,1);                                        # choice of parameters
L=10000; TRANS=5000;                                           # integration length and transient

b=0.45;                                                        # chaotic attractor
MOD_par=[-1 -1 1 a b -c 1];                                    # parameters

X = integrate_ODE_general_BIG(MOD_nr,MOD_par,dt,
                              L,DIM,ODEorder,X0,TRANS);        # integrate system
                                                               # function defined in DDAfunctions.jl
plot(X[:,1],X[:,2],X[:,3],                                     # plot the attractor
     color=:blue,legend=false,
     xlabel=L"x",ylabel=L"y",zlabel=L"z")
plot!(size=(500,500))

display(current());
print("Make pdf file and continue? ");
readline()

savefig("Roessler_0.45.pdf")

b=1;                                                           # periodic attractor
MOD_par=[-1 -1 1 a b -c 1];                                    # parameters
X = integrate_ODE_general_BIG(MOD_nr,MOD_par,dt,
                              L,DIM,ODEorder,X0,TRANS);        # integrate system
                                                               # function defined in DDAfunctions.jl
plot(X[:,1],X[:,2],X[:,3],                                     # plot the attractor
     color=:blue,legend=false,
     xlabel=L"x",ylabel=L"y",zlabel=L"z")
plot!(size=(500,500))

display(current());
print("Make pdf file and continue? ");
readline()

savefig("Roessler_1.pdf")
```

2 Coupled Rössler systems

We couple Rössler systems using diffusive coupling and consider here seven (coupled) Rössler systems

$$\begin{aligned} \dot{u}_{1,n} &= -u_{2,n} - u_{3,n} + \sum_j \epsilon(u_{1,n} - u_{1,j}) \\ \dot{u}_{2,n} &= u_{1,n} + a_n\, u_{2,n} \\ \dot{u}_{3,n} &= b_n + c_n u_{3,n} + u_{1,n} u_{3,n} \end{aligned} \tag{2}$$

with $n = 1, 2, \ldots, 7$ and x_j is the u_1-component of another system j. The values for a_n, b_n, and c_n are listed in Tab. 1. ϵ is either 0 or 0.15 depending on which systems are coupled.

We have 7 three-dimensional systems and therefore 21 variables. In the code we want to number them $x_1, x_2, \ldots x_{21}$ and therefore need to make the following change in variables: $u_{1,n} \to x_{3n-2}$, $u_{2,n} \to x_{3n-1}$,

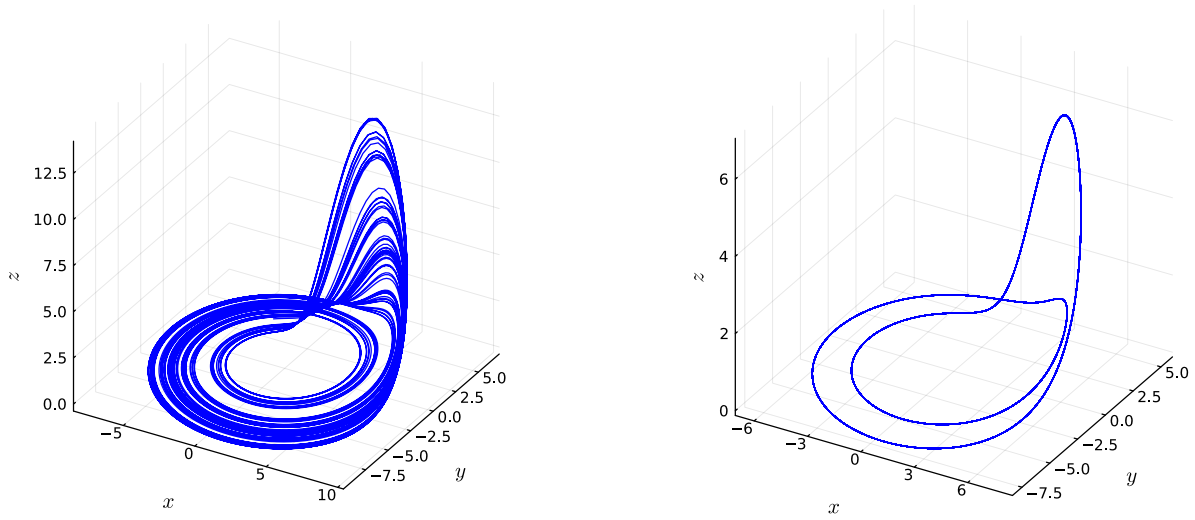

Figure 1: Rössler attractor with $b = 0.45$ (left) and $b = 0.1$ (right)

$u_{3,n} \to x_{3n}$. In general, for a N-dimensional system we would have $u_{k,n} \to x_{Nn-(N-k)}$ This change of variables changes system (2) to

$$\begin{aligned} \dot{x}_{3n-2} &= -x_{3n-1} - x_{3n} + \sum_j \epsilon(x_{3n-2} - x_{3j-2}) \\ \dot{x}_{3n-1} &= x_{3n-2} + a_n\, x_{3n-1} \\ \dot{x}_{3n} &= b_n + c_n x_{3n} + x_{3n-2} x_{3n} \end{aligned} \tag{3}$$

In the code we first encode the 7 systems without the coupling part:

```
include("DDAfunctions.jl");                                    # set of Julia functions

NrSyst=7;                                                      # 7 coupled systems
ROS=[[0  0 2];                                                 # single Roessler system
     [0  0 3];
     [1  0 1];
     [1  0 2];
     [2  0 0];
     [2  0 3];
     [2  1 3]
    ];
(MOD_nr,DIM,ODEorder,P) = make_MOD_nr(ROS,NrSyst);             # encoding of the 7 Roessler systems
                                                               # function defined in DDAfunctions.jl
```

```
a123=0.21;                                                     # model parameters
a456=0.20;
a7  =0.18;
b1 = 0.2150;
b2 = 0.2020;
b3 = 0.2041;
b4 = 0.4050;
b5 = 0.3991;
b6 = 0.4100;
```

Table 1: Parameters of the seven Rössler systems

#	a_n	b_n	c_n
1	0.21	0.21505	-4.5
2	0.21	0.20201	-4.5
3	0.21	0.20411	-4.5
4	0.20	0.40503	-4.5
5	0.20	0.39905	-4.5
6	0.20	0.41000	-4.5
7	0.18	0.50000	-6.8

```
b7 = 0.5000;
c =5.7;
c7=6.8;
MOD_par=[
         -1 -1 1 a123  b1 -c  1
         -1 -1 1 a123  b2 -c  1
         -1 -1 1 a123  b3 -c  1
         -1 -1 1 a456  b4 -c  1
         -1 -1 1 a456  b5 -c  1
         -1 -1 1 a456  b6 -c  1
         -1 -1 1 a7    b7 -c7 1
        ];
MOD_par=reshape(MOD_par',size(ROS,1)*NrSyst)';
```

2.1 First network example

For the first example the network of the seven Rössler systems in (2) reads as

$$
\begin{aligned}
&\text{(i)}\left\{\begin{array}{ll}
\text{R1}\left\{\begin{array}{lcl} \dot{x}_1 &=& -y_1 - z_1 + 0.15(x_1 - x_7) \\ \dot{y}_1 &=& x_1 + a_1\, y_1 \\ \dot{z}_1 &=& b_1 + c_1 z_1 + x_1 z_1 \end{array}\right. \\
\text{R2}\left\{\begin{array}{lcl} \dot{x}_2 &=& -y_2 - z_2 \\ \dot{y}_2 &=& x_2 + a_2\, y_2 \\ \dot{z}_2 &=& b_2 + c_2 z_2 + x_2 z_2 \end{array}\right. \\
\text{R3}\left\{\begin{array}{lcl} \dot{x}_3 &=& -y_3 - z_3 \\ \dot{y}_3 &=& x_3 + a_3\, y_3 \\ \dot{z}_3 &=& b_3 + c_3 z_3 + x_3 z_3 \end{array}\right.
\end{array}\right. \qquad
\text{(ii)}\left\{\begin{array}{ll}
\text{R4}\left\{\begin{array}{lcl} \dot{x}_4 &=& -y_4 - z_4 + 0.15(x_4 - x_7) \\ \dot{y}_4 &=& x_4 + a_4\, y_4 \\ \dot{z}_4 &=& b_4 + c_4 z_4 + x_4 z_4 \end{array}\right. \\
\text{R5}\left\{\begin{array}{lcl} \dot{x}_5 &=& -y_5 - z_5 + 0.15(x_5 - x_7) \\ \dot{y}_5 &=& x_5 + a_5\, y_5 \\ \dot{z}_5 &=& b_5 + c_5 z_5 + x_5 z_5 \end{array}\right. \\
\text{R6}\left\{\begin{array}{lcl} \dot{x}_6 &=& -y_6 - z_6 \\ \dot{y}_6 &=& x_6 + a_6\, y_6 \\ \dot{z}_6 &=& b_6 + c_6 z_6 + x_6 z_6 \end{array}\right.
\end{array}\right. \\
&\qquad\qquad \text{(iii)}\left\{\begin{array}{ll}
\text{R7}\left\{\begin{array}{lcl} \dot{x}_7 &=& -y_7 - z_7 + 0.15(x_7 - x_3) \\ \dot{y}_7 &=& x_7 + a_7\, y_7 \\ \dot{z}_7 &=& b_7 + c_7 z_7 + x_7 z_7 \end{array}\right.
\end{array}\right.
\end{aligned}
\tag{4}
$$

with the values for a_n, b_n, and c_n ($n = (1, 2, \ldots, 7)$) as listed in (1). (i), (ii), and (iii) denote the different dynamical regimes. The adjacency matrix and network motif of ground truth in (4) is shown in (2). The three different colors in the nodes of the motif denote the three dynamical regimes (i), (ii), (iii).

The encoding for the couplings in Fig. 2 is done in the following way:

	from			to	
i	Eq. #	variable	j	Eq. #	variable
3	0	0 1	7	0	0 1
7	0	0 1	1	0	0 1
7	0	0 1	4	0	0 1
7	0	0 1	5	0	0 1

```
FromTo=[
        [3 0  0 1   7 0  0 1];
        [7 0  0 1   1 0  0 1];
        [7 0  0 1   4 0  0 1];
        [7 0  0 1   5 0  0 1]
        ];
PF = "__FirstExample";                                          # parts of file names

II=make_MOD_nr_Coupling(FromTo,DIM,P);                          # MOD_nr part for coupling
epsilon=0.15;                                                   # coupling strength
MOD_par_add=repeat([epsilon -epsilon],size(FromTo,1),1)'[:]';   # MOD_par for coupling part
```

We need to adjust the integration length according to the DDA parameters. For data of length `L`, the maximal delay `TM`, the number of data points for numerical integration `dm`, a window length `WL`, and a window shift `WS` the window number we loose `dm + TM` data points at the beginning of the time series and `dm` data points at the end. The number of windows `WN` of the DDA output is then `WN = 1 + floor((L-WL-TM-2*dm)/WS)`: For anticipated 100 windows we then can compute the data length.

```
TAU=[32 9]; TM=maximum(TAU); dm=4;                              # DDA parameters
WL=4000;WS=2000;                                                # window length and window shift for DDA
WN=100;                                                         # assign window number
LL=WS*(WN-1)+WL+TM+2*dm;                                        # ajust integration length
```

The seven Rössler systems are integrated with a step size of 0.05 and down-sampled by a factor of two.

```
TRANS=20000;                                                    # transient
dt=0.05;                                                        # integration step size
X0=rand(DIM*NrSyst,1);                                          # initial conditions
DATA_DIR="DATA"; dir_exist(DATA_DIR);                           # DATA folder
noise="NoNoise";NOISE="NoNoise";                                # parts of file names
FN=@sprintf("%s%sCD_DDA_data_%s__WL%d_WS%d_WN%d%s.ascii",
            DATA_DIR,SL,noise,WL,WS,WN,PF);                     # noise free data file
CH_list=1:DIM:DIM*NrSyst;                                       # only x
DELTA=2;                                                        # every second data point
if !isfile(FN)
   integrate_ODE_general_BIG([MOD_nr II],[MOD_par MOD_par_add],  # encoding of the coupled systems
                             dt,                                # step size of num. integration
```

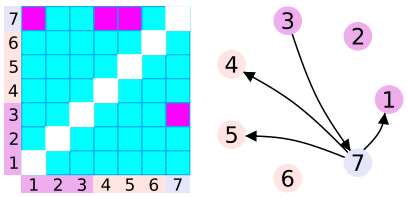

Figure 2: Adjacency matrix and network motif of ground truth. The connections are indicated as magenta boxes.

```
                            LL,                                         # length
                            DIM*NrSyst,ODEorder,X0,                     # parameters
                            FN,
                            CH_list,DELTA,
                            TRANS);
end
```

We add noise to the data and run DDA

```
SNRadd_list= 20:-1:0;

MakeDataNoise(PF,noise,SNRadd_list);                                    # add noise


DDA_DIR="DDA"; dir_exist(DDA_DIR);                                      # DDA folder

nr_delays=2;
DDAmodel=[[0 0 1];
          [0 0 2];
          [1 1 1]];
(MODEL, L_AF, DDAorder)=make_MODEL(DDAmodel);                           # DDA model

NrCH=NrSyst; CH=collect(1:NrCH);
LIST=collect(combinations(CH,2));
LL1=vcat(LIST...)';
LIST=reduce(hcat,LIST)';                                                # pairwise combinations

RunDDA(PF,NOISE,SNRadd_list);                                           # run DDA
```

Then we plot the results:

```
(C,E)=makeCE(PF,NOISE,SNRadd_list);                                     # read outputs

c=reshape(C[:,:,1],NrSyst^2); IDX=reverse(sortperm(c[:])); IDX=IDX[1:NrSyst^2-NrSyst];
c=reshape(C,NrSyst^2,length(SNRadd_list)+1); c=c[IDX,:];
e=reshape(E,NrSyst^2,length(SNRadd_list)+1); e=e[IDX,:];

S=[repeat(1:NrSyst,NrSyst,1) repeat(1:NrSyst,1,NrSyst)'[:]][IDX,:];

p1=Plots.palette(cgrad(:cool,scale=:log,rev=true),42);
p2=Plots.palette(cgrad(:cool,scale=:log),42);

A=reshape(C[:,:,1],NrSyst^2); idx=reverse(sortperm(A[:])); A=A[idx[1:NrSyst^2-NrSyst]];
A=A .- A[end]; A=A ./ A[1] .* 1000; A=Int.(floor.(A)); A[A .== 0] .= 1;
p1=Plots.palette(cgrad(:cool,scale=:log),1000); p1=p1[A];

pp=plot(size=(1000,550),margins=10*Plots.px,layout=@layout[a{0.8w} [b;c]]);

scatter!(pp,subplot=1,c[:,1:end]',e[:,1:end]',msw=0,palette=p1,markersize=5,
            xscale=:log10,yscale=:log10,label="")
plot!(pp,subplot=1,c',e',msw=0,palette=p1,linewidth=0.2,xscale=:log10,yscale=:log10,label="",grid=false)

plot!(pp,subplot=1,xlabel=L"$\mathcal{C}$");
plot!(pp,subplot=1,ylabel=L"$\mathcal{E}$");

for k=1:size(FromTo,1)+3
    txt=join([string(S[k,1]) " \u21FE " string(S[k,2])])
    annotate!(pp,subplot=1,c[k,1],e[k,1],Plots.text(txt,18,p1[k]));
end
display(pp)

cc=C[:,:,1]; cc=cc .- minimum(cc); cc=cc ./ maximum(cc);
cc[diagind(cc)] .= NaN;

heatmap!(pp,subplot=2,cc,aspect_ratio=:equal,c=p2,colorbar=false,grid=false,axis=([], false))
```

```
display(pp)


GR.setarrowsize(0.5);

MS = [1,1,1,2,2,2,3];
colors = [colorant"plum2", colorant"mistyrose1", colorant"lavender"];

A=C[:,:,1];
A[A .== 0] .= 1;
A = A .- minimum(A);
A[A .== maximum(A)] .= 0;
A = A ./ maximum(A);

A[A .< 0.33] .= 0;

graphplot!(pp,subplot=3,A,
              method=:circular,nodeshape=:circle,
              names=1:7,
              markersize=0.15,
              fontsize=20,
              linewidth=3,
              linealpha=1,
              markercolor = colors[MS],
              nodestrokecolor=colors[MS],
              arrow=arrow(:closed,10),
              )

display(pp)
```

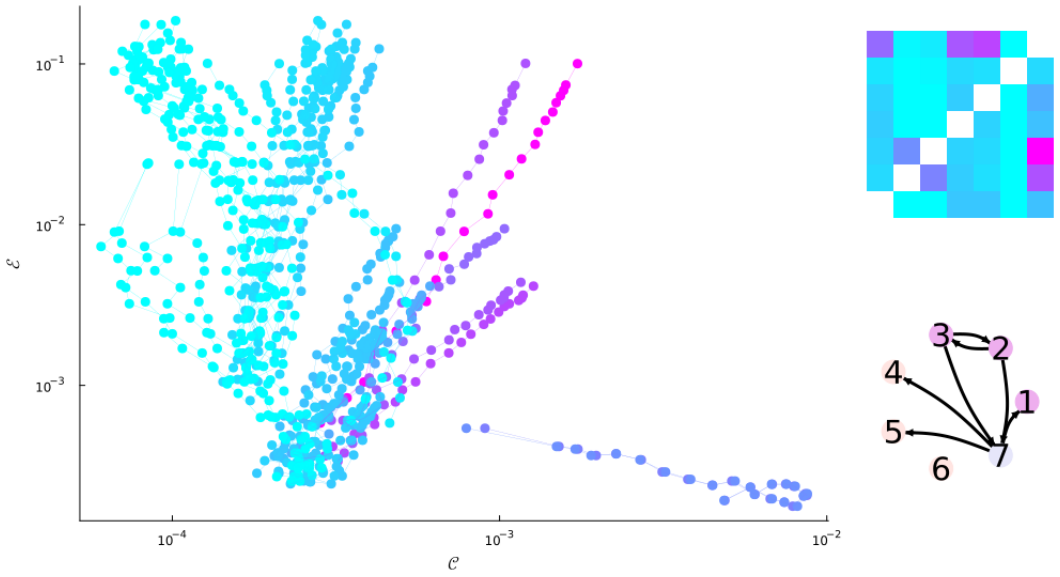

Figure 3: $\mathcal{E}$ over $\mathcal{C}$; numerical adjacency matrix and network motif. The colormap is on a `log10` scale from cyan (low) to magenta (high). The colors in the adjacency matrix are the same as those in the $\mathcal{C}$-$\mathcal{E}$ plot on the left (see code above).

We also plot the adjacency matrices for added noise:

```
pp1=plot(size=(1000,200),margins=10*Plots.px,layout=(1,5));
k=1;cc=C[:,:,1]; M1=minimum(cc); cc=cc .- M1; M2=maximum(cc); cc=cc ./ M2;
heatmap!(pp1,subplot=Int((k - 1)/5 + 1),cc,
             aspect_ratio=:equal,c=p2,colorbar=false,
             grid=false,axis=([], false),clims=(0,1),
             title="no noise")

for k=6:5:22
    cc=C[:,:,k]; cc=cc .- M1; cc=cc ./ M2; cc[diagind(cc)] .= NaN;
    heatmap!(pp1,subplot=Int((k - 1)/5 + 1),cc,
             aspect_ratio=:equal,c=p2,colorbar=false,
             grid=false,axis=([], false),clims=(0,1),
             title=@sprintf("SNR = %d dB",SNRadd_list[k]))
end
display(pp1)
```

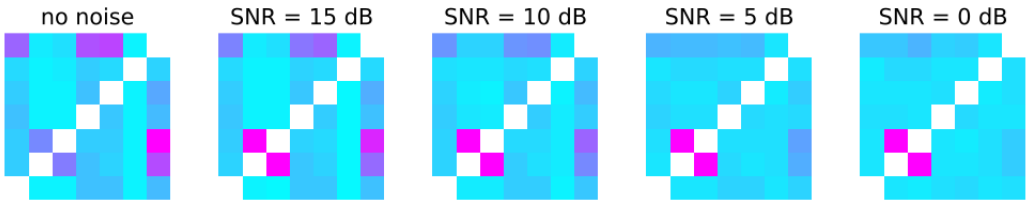

Figure 4: Adjacency matrices for added white noise.

2.2 Unconnected nodes

To change the network we have to change the following parameters in the code

```
FromTo=[];
PF = "__Empty";
```

and get the following results:

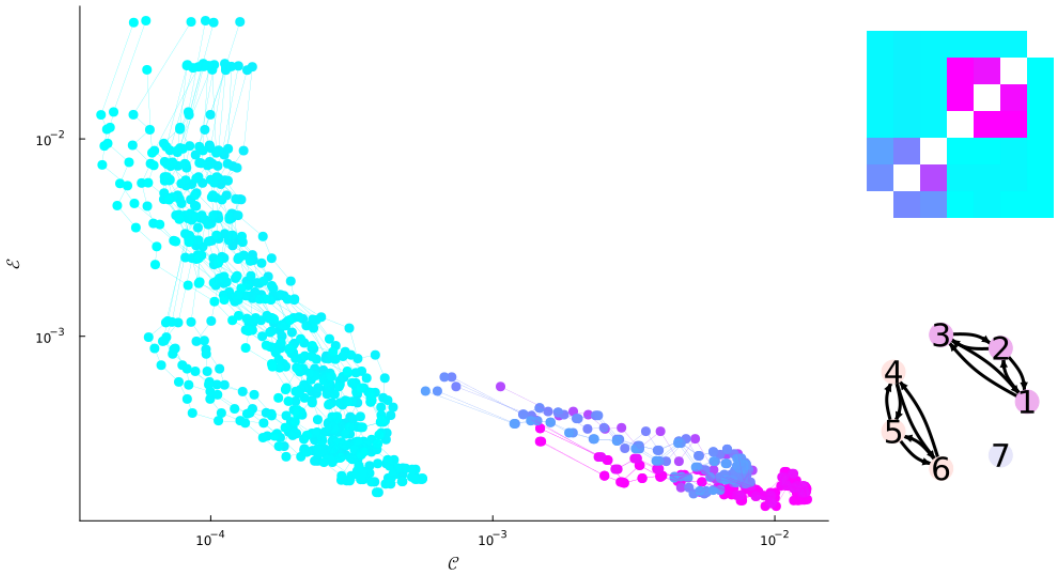

Figure 5: $\mathcal{E}$ over $\mathcal{C}$; numerical adjacency matrix and network motif. The colormap is on a `log10` scale from cyan (low) to magenta (high). The colors in the adjacency matrix are the same as those in the $\mathcal{C}$-$\mathcal{E}$ plot on the left (see code).

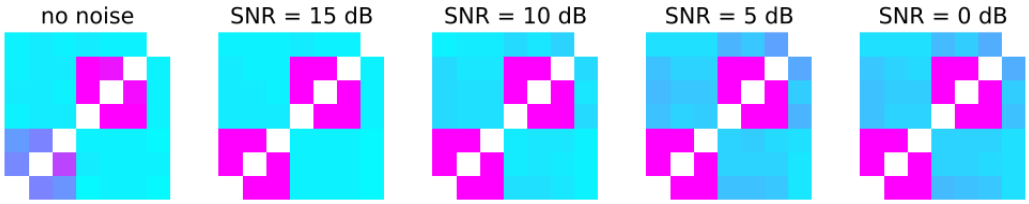

Figure 6: Adjacency matrices for added white noise.

3 Epilepsy

We here use a 4 minute data segment around the second seizure we show in the paper.

```
# julia -p10

@everywhere include("DDAfunctions.jl");

using JLD2

#import Pkg; Pkg.add("Distributed")
using Distributed

FN_part="S05__00_04";
NOISE_list=["NoNoise";"add15dB"];

FN_NoNoise = @sprintf("%s__%s.ascii",FN_part,NOISE_list[1]);
FN_15dB    = @sprintf("%s__%s.ascii",FN_part,NOISE_list[2]);

if !isfile(FN_15dB)
   X=readdlm(FN_NoNoise);

   NrCH=size(X,2);

   SNR=15;
   Y=fill(NaN,size(X,1),NrCH);
   for n_ch=1:NrCH
       Y[:,n_ch]=add_noise(X[:,n_ch],SNR);
   end

   fid=open(FN_15dB,"w");
   for k1=1:size(Y,1)
       for k2=1:NrCH
           @printf(fid,"%13.10f ",Y[k1,k2]);
       end
       @printf(fid,"\n");
   end
   close(fid);

   X = nothing;
   Y = nothing; GC.gc();
end

NrCH=78;
CH=1:NrCH;
ONSET_CH = sort([45:52; 85:92; 61:68; 101:108] .- 30);

TAU=[7 10]; TM = maximum(TAU); dm=4;
WL=500; WS=50;

nr_delays=2;
DDAmodel=[[0 0 0 1];
          [0 0 0 2];
          [1 1 1 1]];
(MODEL, L_AF, DDAorder)=make_MODEL(DDAmodel);                          # DDA model

DDA_DIR="DDA_Epilepsy"; dir_exist(DDA_DIR);

LIST=reduce(hcat,collect(combinations(CH,2)))';
DELTA=20;
N=Int64(ceil(size(LIST,1)/DELTA));

for n_FN=1:length(NOISE_list)

    FN_DATA = @sprintf("%s__%s.ascii",FN_part,NOISE_list[n_FN]);
    FN_DDA  = @sprintf("%s%s%s__%s.DDA",DDA_DIR,SL,FN_part,NOISE_list[n_FN]);

    FN_ALL = @sprintf("%s%s%s__%s.jld2",DDA_DIR,SL,FN_part,NOISE_list[n_FN]);
    if !isfile(FN_ALL)
```

```
@printf("%s\n",FN_ALL);

if !isfile(join([FN_DDA,"_ST"]))
   if Sys.iswindows()
      if !isfile("run_DDA_ASCII.exe")
         run(`cp run_DDA_ASCII run_DDA_ASCII.exe`);
      end

      CMD=".\\run_DDA_ASCII.exe";
   else
      CMD="./run_DDA_ASCII";
   end

   CMD = "$CMD -ASCII";
   CMD = "$CMD -MODEL $(join(MODEL," "))"
   CMD = "$CMD -TAU $(join(TAU," "))"
   CMD = "$CMD -dm $dm -order $DDAorder -nr_tau $nr_delays"
   CMD = "$CMD -DATA_FN $FN_DATA -OUT_FN $FN_DDA"
   CMD = "$CMD -WL $WL -WS $WS";
   CMD = "$CMD -SELECT 1 0 0 0";                                  # ST-DDA

   if Sys.iswindows()
      run(Cmd(string.(split(CMD, " "))));
   else
      run(`sh -c $CMD`);
   end

   rm(@sprintf("%s.info",FN_DDA));
end

@sync @distributed for n_N=1:N
    FN_DDAn=@sprintf("%s%s%s__%s__%03d.DDA",DDA_DIR,SL,FN_part,NOISE_list[n_FN],n_N);

    n=collect(1:DELTA) .+ (n_N-1)*DELTA; n=n[n.<=size(LIST,1)];
    LL1=LIST[n,:]; LL1=vcat(LL1'...)';

    if !isfile(join([FN_DDAn,"_CT"]))
       if Sys.iswindows()
          if !isfile("run_DDA_ASCII.exe")
             run(`cp run_DDA_ASCII run_DDA_ASCII.exe`);
          end

          CMD=".\\run_DDA_ASCII.exe";
       else
          CMD="./run_DDA_ASCII";
       end

       CMD = "$CMD -ASCII";
       CMD = "$CMD -MODEL $(join(MODEL," "))"
       CMD = "$CMD -TAU $(join(TAU," "))"
       CMD = "$CMD -dm $dm -order $DDAorder -nr_tau $nr_delays"
       CMD = "$CMD -DATA_FN $FN_DATA -OUT_FN $FN_DDAn"
       CMD = "$CMD -WL $WL -WS $WS";
       CMD = "$CMD -SELECT 0 1 0 0"                               # CT-DDA
       CMD = "$CMD -CT_CH_list $(join(LL1," "))";                 # all pairwise channels
       CMD = "$CMD -WL_CT 2 -WS_CT 2";

       if Sys.iswindows()
          run(Cmd(string.(split(CMD, " "))));
       else
          run(`sh -c $CMD`);
       end

       rm(@sprintf("%s.info",FN_DDAn));
    end
end

@sync @distributed for n_N=1:N
    FN_DDAn=@sprintf("%s%s%s__%s__%03d.DDA",DDA_DIR,SL,FN_part,NOISE_list[n_FN],n_N);

    n=collect(1:DELTA) .+ (n_N-1)*DELTA; n=n[n.<=size(LIST,1)];
```

```
        LL1=LIST[n,:]; LL1=vcat(LL1'...)';

        if !isfile(join([FN_DDAn,"_CD_DDA_ST"]))
           if Sys.iswindows()
              if !isfile("run_DDA_ASCII.exe")
                 run(`cp run_DDA_ASCII run_DDA_ASCII.exe`);
              end

              CMD=".\\run_DDA_ASCII.exe";
           else
              CMD="./run_DDA_ASCII";
           end

           CMD = "$CMD -ASCII";
           CMD = "$CMD -MODEL $(join(MODEL," "))"
           CMD = "$CMD -TAU $(join(TAU," "))"
           CMD = "$CMD -dm $dm -order $DDAorder -nr_tau $nr_delays"
           CMD = "$CMD -DATA_FN $FN_DATA -OUT_FN $FN_DDAn"
           CMD = "$CMD -WL $WL -WS $WS";
           CMD = "$CMD -SELECT 0 0 1 0";                          # CD-DDA
           CMD = "$CMD -PAIRS $(join(LL1," "))";                  # all pairwise channels

           if Sys.iswindows()
              run(Cmd(string.(split(CMD, " "))));
           else
              run(`sh -c $CMD`);
           end

           rm(@sprintf("%s.info",FN_DDAn));
        end
    end

    ############  stitch together outputs

    ST=readdlm(join([FN_DDA,"_ST"]));
    T=ST[:,1:2]; ST=ST[:,3:end];
    WN=size(T,1);

    rhoS=ST[:,L_AF:L_AF:end];
    ST = nothing; GC.gc();

    E=fill(NaN,WN,NrCH,NrCH);
    for n_N=1:N
        @printf("%3d ",n_N)
        FN_DDAn=@sprintf("%s%s%s__%s__%03d.DDA",DDA_DIR,SL,FN_part,NOISE_list[n_FN],n_N);

        n=collect(1:DELTA) .+ (n_N-1)*DELTA; n=n[n.<=size(LIST,1)];
        LL1=LIST[n,:] .- CH[1] .+ 1;

        CT=readdlm(join([FN_DDAn,"_CT"]));
        CT=CT[:,3:end];
        CT=CT[:,L_AF:L_AF:end];

        for l=1:size(LL1,1)
            ch1=LL1[l,1];ch2=LL1[l,2];
            E[:,ch1,ch2] = abs.( dropdims(mean(rhoS[:,[ch1,ch2]],dims=2),dims=2) ./ CT[:,l] .- 1 );
            E[:,ch2,ch1] = E[:,ch1,ch2];
        end
        CT = nothing; GC.gc();
    end
    @printf("\n");

    C=fill(NaN,WN,NrCH,NrCH);
    for n_N=1:N
        @printf("%3d ",n_N)
        FN_DDAn=@sprintf("%s%s%s__%s__%03d.DDA",DDA_DIR,SL,FN_part,NOISE_list[n_FN],n_N);

        n=collect(1:DELTA) .+ (n_N-1)*DELTA; n=n[n.<=size(LIST,1)];
        LL1=LIST[n,:] .- CH[1] .+ 1;

        CD=readdlm(join([FN_DDAn,"_CD_DDA_ST"]));
```

```
            CD=CD[:,3:end];
            CD=reshape(CD,WN,2,size(LL1,1));

            for l=1:size(LL1,1)
                ch1=LL1[l,1];ch2=LL1[l,2];
                C[:,ch1,ch2] = CD[:,2,l];
                C[:,ch2,ch1] = CD[:,1,l];
            end
            CD = nothing; GC.gc();
        end
        @printf("\n\n");

        @save FN_ALL C E rhoS T WN
        E = nothing; C = nothing; GC.gc();
    end

end

###########  plot results

n_FN=1;
FN_ALL = @sprintf("%s%s%s__%s.jld2",DDA_DIR,SL,FN_part,NOISE_list[n_FN]);
@load FN_ALL C E T WN ;
C1 = C .* 1;
E1 = E .* 1;
E = nothing; C = nothing; GC.gc();

n_FN=2;
FN_ALL = @sprintf("%s%s%s__%s.jld2",DDA_DIR,SL,FN_part,NOISE_list[n_FN]);
@load FN_ALL C E T WN ;
C2 = C .* 1;
E2 = E .* 1;
E = nothing; C = nothing; GC.gc();

eLABEL=["LFP"; "LCG"; "LAT"; "LMT"; "LHP"; "LOC"; "LTH"; "LSU";
        "RFP"; "RCG"; "RAT"; "RMT"; "RHP"; "ROC"; "RTH"; "RSU"]
e_list = [
        [34:38; 15:22; 25:30],      #LFP
        [31:33; 23; 24],            #LCG
        [3:5; 10:14],               #LAT
        Int[],
        [1:2; 7:9],                 #LHP
        Int[],
        Int[],
        Int[],
        [55:62; 71:78; 63; 65:70],  #RFP
        Int[],
        [40:44; 50:52],             #RAT
        Int[],
        [39; 47:49],                #RHP
        Int[],
        Int[],
        [64]                        #RSU
        ];
e_NotZero = findall(x -> x == 1, length.(e_list)' .!= 0 );
e_NotZero = [i[2] for i in e_NotZero];

SR=500;
t=(T[:,1] .+ 1 .+ TM .+ dm) ./ SR ./ 60;

SEQ=1:length(e_NotZero);

CHs=vcat(e_list[e_NotZero][SEQ][:]...);
L_e_list=map(x -> length(e_list[e_NotZero][x]),SEQ);

IND=setdiff(1:length(CHs)^2,diagind(C1[1,CHs,CHs]));
c1=reshape(C1[:,CHs,CHs],WN,length(CHs)^2)[:,IND];
c2=reshape(C2[:,CHs,CHs],WN,length(CHs)^2)[:,IND];
e1=reshape(E1[:,CHs,CHs],WN,length(CHs)^2)[:,IND];
e2=reshape(E2[:,CHs,CHs],WN,length(CHs)^2)[:,IND];
```

```
c2[c2 .< 0.01] .= NaN;
c1[c1 .< 0.01] .= NaN;

BETA=asin.((c2 .- c1) ./ sqrt.( (c1 .- c2).^2 + (e1 .- e2).^2) );

SG = plot(size=(1000,1000),layout=(2,2));

heatmap!(SG,subplot=1,
         t,1:length(IND),c1',
         c=:jet,
         xtickfont=font(12), ytickfont=font(12),
         colorbar = true,clims=(0.1,0.06)
         )
hline!(SG,subplot=1,[cumsum(L_e_list[:]) .* (length(CHs)-1) .+ 0.5],legend=false,c=:black,linewidth=2)
Y = vcat(0,cumsum(L_e_list[:]) .* (length(CHs)-1));
Y = (Y .- [0;diff(Y)./2])[2:end];
heatmap!(SG,subplot=1,yticks=(Y,eLABEL[e_NotZero][SEQ]),xlabel="time [min]",clims=(0.01,0.06))
display(SG)

heatmap!(SG,subplot=2,
         t,1:length(IND),c2',
         c=:jet,
         xtickfont=font(12), ytickfont=font(12),
         colorbar = true,clims=(0.1,0.06)
         )
hline!(SG,subplot=2,[cumsum(L_e_list[:]) .* (length(CHs)-1) .+ 0.5],legend=false,c=:black,linewidth=2)
heatmap!(SG,subplot=2,yticks=(Y,eLABEL[e_NotZero][SEQ]),xlabel="time [min]",clims=(0.01,0.06))
display(SG)

ALPHA = BETA .* 1;
ALPHA[ALPHA .>= 0] .= 0;
ALPHA[ALPHA .< 0]  .= 1;
ALPHA[ALPHA .== 0] .= NaN;
heatmap!(SG,subplot=3,
         t,1:length(IND),(c1 .* ALPHA)',
         c=:jet,
         xtickfont=font(12), ytickfont=font(12),
         colorbar = true,clims=(0.1,0.06)
         )
hline!(SG,subplot=3,[cumsum(L_e_list[:]) .* (length(CHs)-1) .+ 0.5],legend=false,c=:black,linewidth=2)
heatmap!(SG,subplot=3,yticks=(Y,eLABEL[e_NotZero][SEQ]),xlabel="time [min]",clims=(0.01,0.06))
display(SG)

ALPHA = BETA .* 1;
ALPHA[ALPHA .<= 0] .= 0;
ALPHA[ALPHA .> 0]  .= 1;
ALPHA[ALPHA .== 0] .= NaN;
heatmap!(SG,subplot=4,
         t,1:length(IND),(c1 .* ALPHA)',
         c=:jet,
         xtickfont=font(12), ytickfont=font(12),
         colorbar = true,clims=(0.1,0.06)
         )
hline!(SG,subplot=4,[cumsum(L_e_list[:]) .* (length(CHs)-1) .+ 0.5],legend=false,c=:black,linewidth=2)
heatmap!(SG,subplot=4,yticks=(Y,eLABEL[e_NotZero][SEQ]),xlabel="time [min]",clims=(0.01,0.06))
display(SG)
```

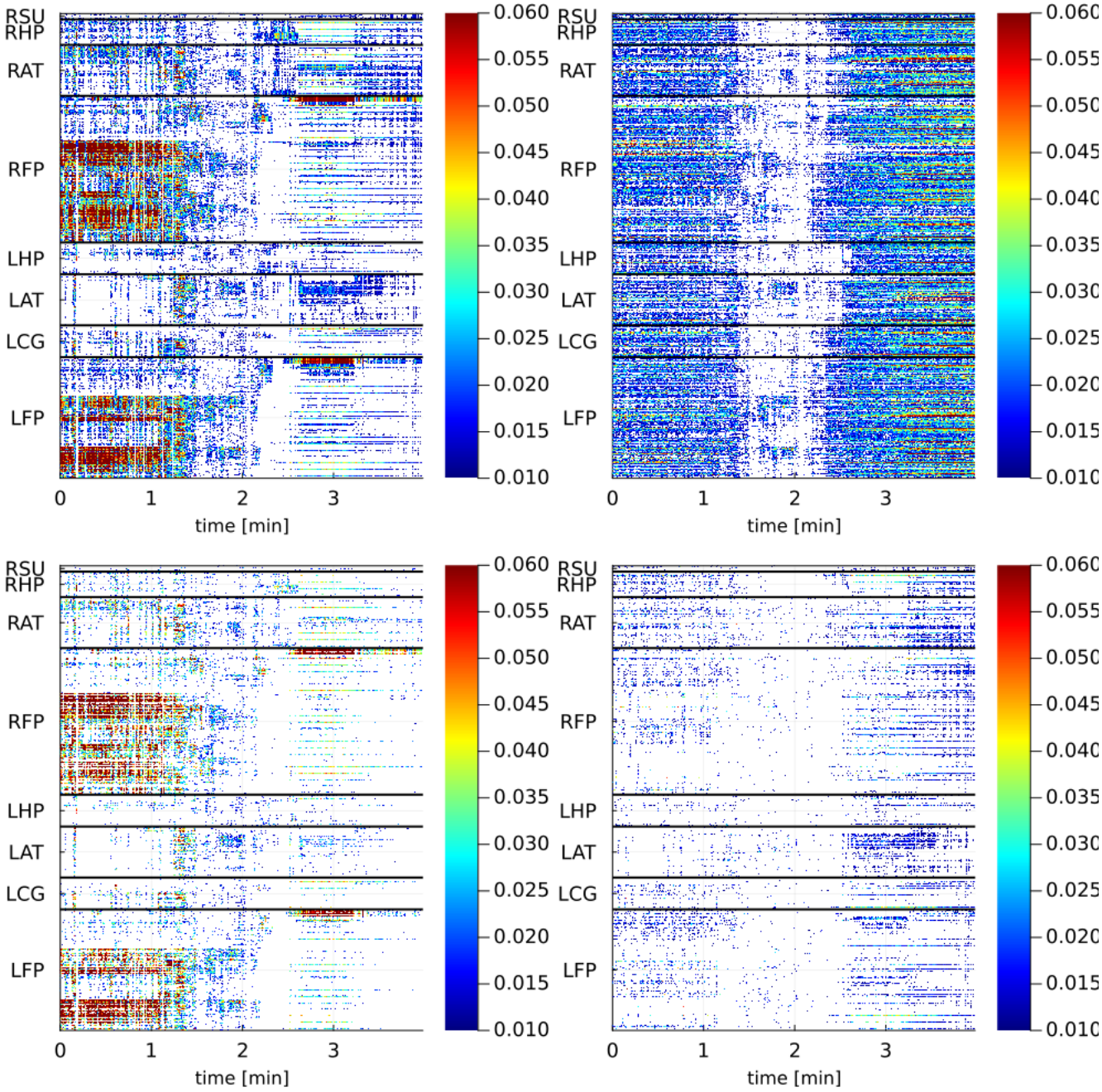

Figure 7: Figure 8 in the paper, but for only 4 minutes of data.