DOCUMENTATION

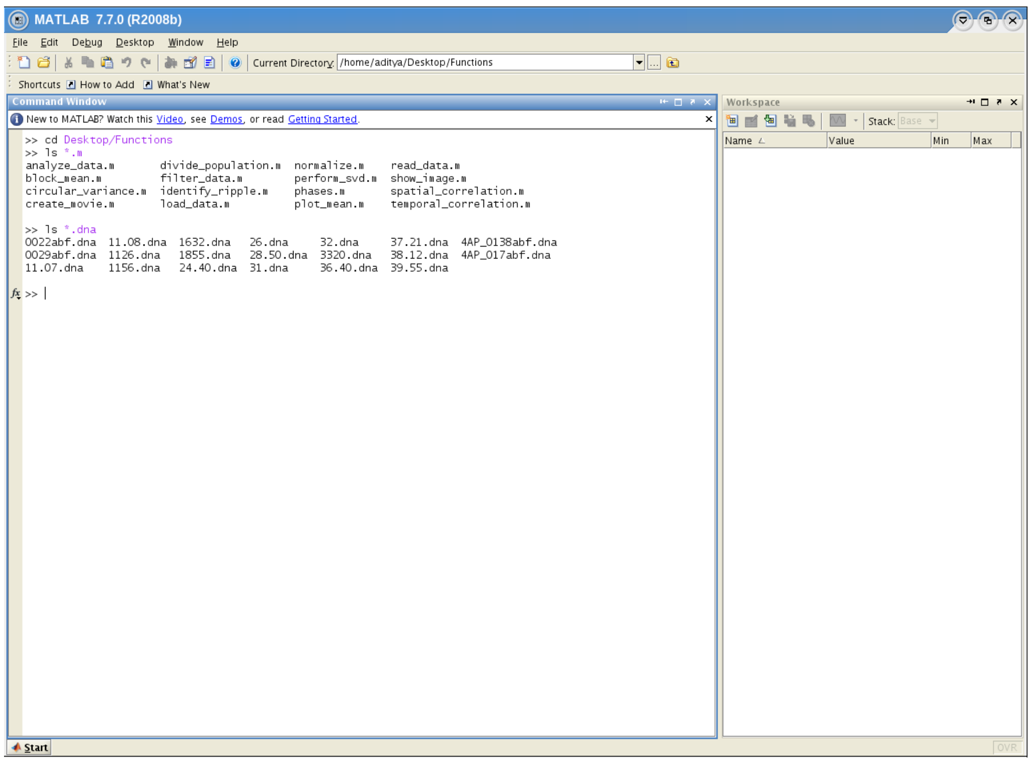
Contents

1 Exporting Data from Brain Vision

Data from the Brain Vision software can be exported as ASCII files (created as ".dna" files).

2 Loading the Exported Data and creating a Data matrix

Once the data has been exported as a .dna file , it can be loaded into MATLAB for processing . First of all , it is necessary to change the MATLAB directory to the directory where the files are located. This is done by using the *cd* command :



On my computer, all the files (data and MATLAB programs) are stored in the folder *Functions* on the *Desktop*. Writing *ls* lists all the files in the current directory . Writing *ls *.dna* lists all the files with an extension *.dna* . MATLAB programs have an extension of *.m*. So typing *ls *.m* will list all the MATLAB programs in the current directory .

Now , to load the data exported from the Brain Vision Software into MATLAB , I wrote a program called *load_data.m*. The syntax for this program is :

$$C = load\_data(filename);$$

The output C is the data from the $.dna$ file . As an example , the file '11.07.dna' can be loaded by the following command :

$$C = load\_data('11.07.dna');$$

Note that this might take some time since these files are quite large.

3 Reading in the data and creating a data matrix

Once the data has been loaded , the variable C should appear in the list of variables in the workspace In order to read data from a variable time window the command is :

$$X\_original = read\_data(a1, a2, s, m, n, C);$$

where a1=x coordinate of the upper left corner of the space window , a2=y coordinate of the upper left corner of the space window , s = size of the window, m=initial time , n=final time , C= loaded data from the previous step . For example , suppose we wanted to read in the entire image for 700 frames . We would then write:

$$X\_original = read\_data(1, 1, 99, 1, 700, C);$$

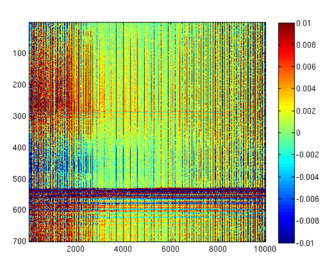
X_original is the data matrix and contains the original dF/F values.

4 Viewing the data matrix

The data matrix can be viewed by typing :

$$show\_image(X\_original)$$

The resultant output looks like :



There appears to be a minor ripple at around frame 280 and then a major one at about frame 520 .

5 Performing a Singular Value Decomposition

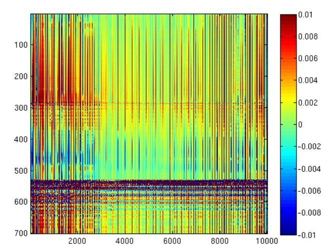
A singular value decomposition and subsequently retaining the principal components can be done by running the following command:

$$Xrecon = perform\_svd(X, M);$$

where X is a data matrix and M is the number of principal components to retain.As an example, we can do :

$$Xrecon = perform\_svd(X\_original, 10);$$

The output can be viewed by calling $show\_image(Xrecon)$ and looks like :



This is a denoising process .

6 Filtering the data

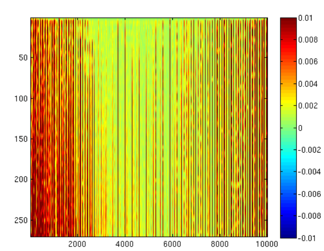
Low pass filtering can be performed by calling :

$$Xfilt = filter\_data(X, Fpass);$$

where Fpass is the cut-off frequency. Only frequencies below Fpass are allowed to pass through.For example, we can low pass filter the data with a cut-off frequency of 100Hz.This should allow slowly varying trends to become visible.In order to prevent the wave from affecting how the filtered data looks , we consider data only upto time 270.

$$X2 = read\_data(1, 1, 99, 1, 270, C);$$
$$Xfilt = filter\_data(X2, 100);$$

The result looks like :

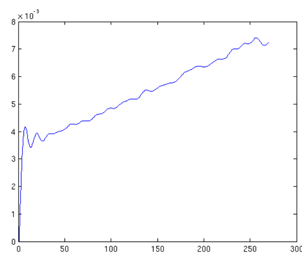


7 Plotting the mean activity

The mean activity can be plotted as a function of time by using:

$$R = plot\_mean(X, a, b);$$

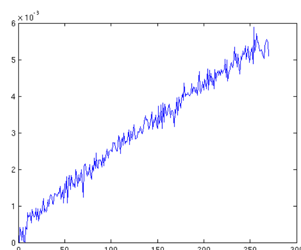
where X is the data matrix . a is the initial time and b is the final time . As an example :if we plot the mean activity of Xfilt as computed in the previous step we see the following :



This shows a steady increase in activity as the time of the ripple approaches. If we plot the mean activity of the original data by using the following command :

$$R = plot\_mean(X\_original, 1, 270);$$

we see :



which appears to be a noisy version of the previous graph.

8 Spatial correlation Maps

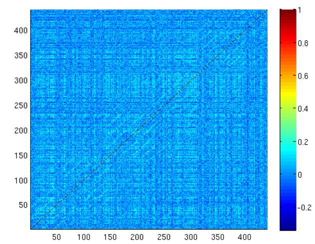
The spatial correlation map can be computed by using the command :

$$R = spatial\_correlation(X);$$

where X is the data matrix. To look at the spatial correlation between pixels before the ripple (from frame 1 to frame 270) , we first select a subset of the image. Note that , calculating the spatial correlation between *all* the pixels is computationally very intensive and in general my computer runs out of memory .

$$X\_subset = read\_data(50, 50, 20, 1, 270, C);$$
$$R1 = spatial\_correlation(X\_subset)$$

The spatial correlation map then looks like :



9 Temporal Correlation Maps

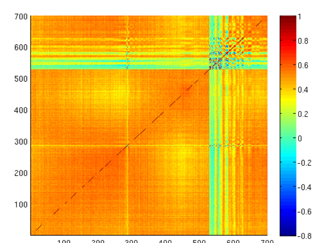
The temporal correlation map can be computed by using :

$$R = temporal\_correlation(X);$$

where X is the data matrix. To look at the temporal correlations and to see the phenomena after the wave , we read in a subset of the image but look at the data over a long period of time till frame 700 .

$$X\_subset = read\_data(50, 50, 20, 1, 700, C);$$
$$R2 = temporal\_correlation(X\_subset)$$

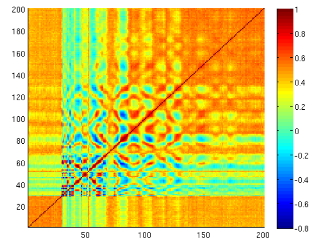
The result looks like :



After the first minor ripple , there donot appear to any distincitive patterns whereas , after the 2nd major ripple ,some interesting patterns can be observed.To look at only this section of the data , we read in the images from time frame 500 to 700 and then look at the temporal correlations.

$$X\_subset = read\_data(50, 50, 20, 500, 700, C);$$
$$R2 = temporal\_correlation(X\_subset)$$

This looks like:



10 Phase Analysis

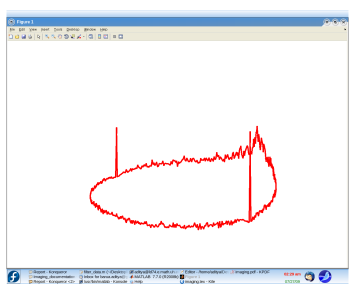
The phases of the pixels in the image can be evaluated by using the command :

$$[P, M] = phases(X);$$

where X is the data matrix. This will display a movie of the phase histogram as well as save the movie as a structure in the variable M.

$$[P, M] = phases(X\_original);$$

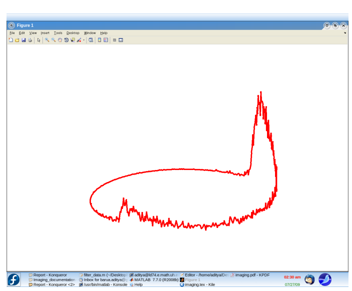
Looking at the Phase histogram of the original data , produces 2 constant peaks superimposed on the histogram which makes it difficult to see how the histogram is actually behaving:



So , we perform a Singular Value Decomposition and then look at the histogram.

$$[P, M] = phases(Xrecon);$$

to observe the bi-modal distribution as well as the phase precession.



11 Circular variance

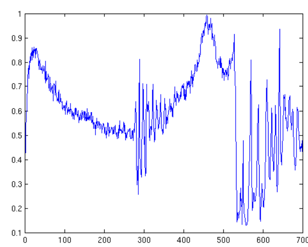
The circular variance is an indicator of how widely distributed the phases are . The circular variance as a function of time can be computed once the phase analysis has been performed. Having obtained the matrix P which is essentially the same as the data matrix with each pixel replaced by its correposnding phase, the circular variance is computed by calling :

$$R = circular\_variance(X);$$

where X is the data matrix.

$$R\_circ = circular\_variance(X\_original);$$

which produces:



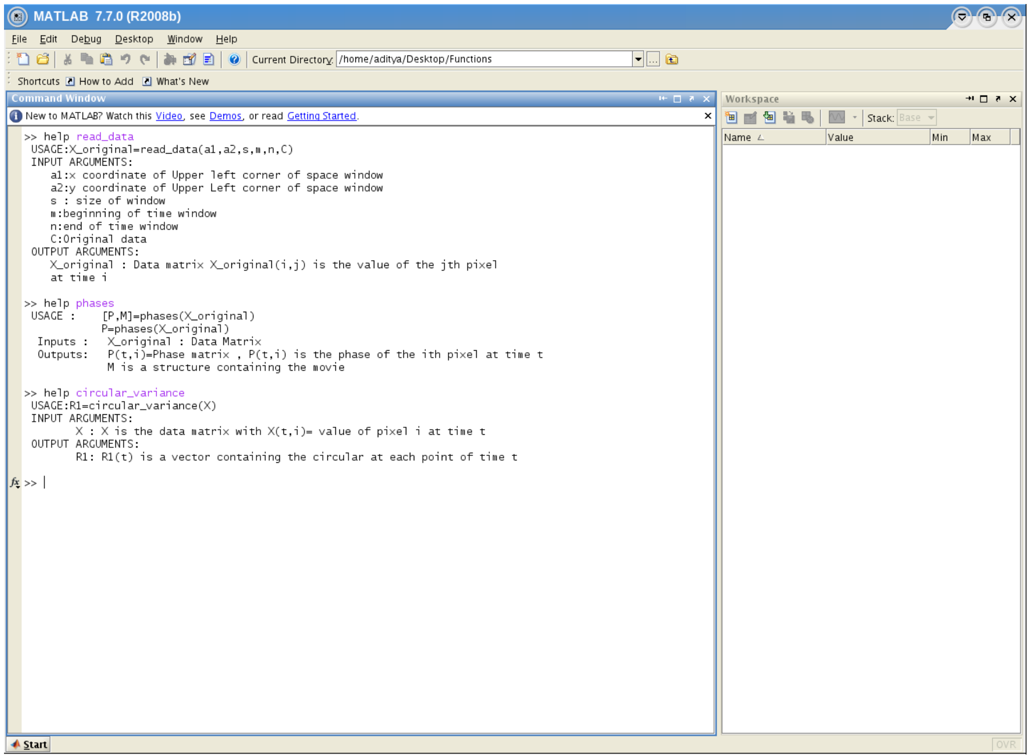
There appears to be a dip in the circular variance before the first minor ripple. However , before the second major ripple the circular variance increases and then falls again which seems to indicate phase synchronization prior to the ripple.

12 Help for the functions

Typing

$$\text{help } function\ name;$$

in the MATLAB command window will display the syntax for all the functions described above. For example :


MATLAB 7.7.0 (R2008b)
File Edit Debug Desktop Window Help
Current Directory: /home/aditya/Desktop/Functions
Shortcuts How to Add What's New
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> help read_data
USAGE:X_original=read_data(a1,a2,s,m,n,C)
INPUT ARGUMENTS:
a1:x coordinate of Upper left corner of space window
a2:y coordinate of Upper Left corner of space window
s : size of window
m:beginning of time window
n:end of time window
C:Original data
OUTPUT ARGUMENTS:
X_original : Data matrix X_original(i,j) is the value of the jth pixel
at time i
>> help phases
USAGE : [P,M]=phases(X_original)
P=phases(X_original)
Inputs : X_original : Data Matrix
Outputs: P(t,i)=Phase matrix , P(t,i) is the phase of the ith pixel at time t
M is a structure containing the movie
>> help circular_variance
USAGE:R1=circular_variance(X)
INPUT ARGUMENTS:
X : X is the data matrix with X(t,i)= value of pixel i at time t
OUTPUT ARGUMENTS:
R1: R1(t) is a vector containing the circular at each point of time t
>>
Workspace
Stack: Base
Name
Value
Min
Max
Start
OVR