Math 801– ODE HW
(A. Assadi)

Systems of Ordinary Differential Equations from Multi-channel Measurements

This is a summary I wrote up for some my class to explore some hands-on exercise with modeling ODE systems from real-data. Some code are provided from past projects and require some modification. What I would like to see is a clean piece of code, well-documented by comments that say what each notation means and what the steps are. The code should use variables rather than numerical arrays and specific integers, and the specific values are assigned to these variables as needed. At the very end, test that the code runs and to debug it: use synthetic data that you generate from random numbers for the EXAMPLE I. If you have time, a short animation (video) sequence for EXAMPLE II could be also useful to check how fast your code runs for larger data sets like video processing. For video, it is not good to use MPEG format, because the compression ruins the quality of the single frames. Perhaps you could generate a simple video sequence with MATLAB that could serve as a test case.

MATLAB PROGRAMMING ASSIGNMENT

We have a multi-channel data acquisition system, such as a collection of electrodes on a rectangular grid. The dimensions of the grid are K and L, and the number of channels is N=K*L.

EXAMPLE I. For example, K=6 and L=15, so N=60 electrodes. To test your code, generate random numbers from a uniform distribution for each entry as a typical numerical data that is given.

EXAMPLE II. Another example is an imaging system, and the numbers could be K=480 and L=640 and N is 480*640=307200. To test your code, use a frame of video that is digitized as an image in gray-scale pixels.

The channels make measurements that we call them X_{ij}, and the array X is formed whose size is (K , L). Each measurement is done in a (possibly) short time-interval, say J. The duration of the experiment is a time interval that we call TIME, measured in milliseconds, divided into small time-intervals by M equal pieces of size J. The notation from starting till end is $t_0, t_1, \ldots t_k , \ldots, t_M$. For example, when we have J=2 milliseconds and M=1000, then each interval $t_{1000} - t_{999}, \ldots, t_2 - t_1, t_1 - t_0$ etc are of length 2 milliseconds each, so TIME=2000 milliseconds. We will denote the time-series of the measurement channels by $X_{ij}(t)$, and consider them as functions of t. The problem that we wish to write the code for is the following.

We assume that a we have a window W that is in the shape of a small matrix of size (m , n). In Example I, W could be from the smallest size (1,1) or it could be any intermediate size, (1,2), or (2,2), or … even the largest possible size (6,15). The values of W will vary as we need to use different sizes. We show this window as a function of time t, and the notation below does not show where the window W is positioned on the original grid, just to keep the notation simple. If we look at the sliding of the window W on the grid, and put the values of each time series $X_{ij}(t)$ into entries of a matrix that we call W(t), then we have a collection of matrices at each time t. Position the window on a temporarily fixed location such that its upper left corner has coordinates (x,y), and just call W(t) for the matrix of values of measurements in time t. We are interested in the time-difference operation DW(t)=W(t)-W(t-1) from $t=t_1, \ldots, t_M$. Consider a length of time T that starts from $t=t_S$ and ends in time $t=t_{S+T}$. The collection of matrices $\{DW(t) \mid t=t_S , \ldots, t_{S+T}\}$ is a data set that we can organize into columns of matrix A. Therefore, the matrix A has T columns and each column is a vector of length equal to the number of entries in W. In Example I, if we use window-size (3 , 3), the columns of A will be arrays of size (9,1). We shall do Principal Components Analysis (PCA) on the data given by the columns of A, which is very similar to the Singular Value Decomposition (SVD). To do PCA, we form the correlation matrix C=A*A' (A' is transpose of A), which is a symmetric matrix, hence has real eigenvalues that are all non-negative, and an orthonormal basis of the corresponding eigenvectors. We sort the eigenvalues according to the largest to the smallest, and call them the first principal value, the second principal value, etc. The

corresponding unit eigenvectors are called the first principal component, the second principal component, etc. We are interested in the first and second principal values and their sum, which we normalize by dividing by the trace of C, denote by E_1, E_2, and E_{12}. Therefore, E_1, E_2, and E_{12} indicate what percentages of the total of the principal values are concentrated in E_1, in E_2, and E_{12}. Corresponding to these, we have three *unit vectors* U_1, U_2, and U_{12} which are the first, second and the sum of the first two principal components that we normalize to be the unit vectors. Notice that now each of the scalars E_1, E_2, and E_{12} are functions of two time-like variables S (the starting point in time for the time window) and T (the time duration of the time window). These are of course all dependent on the position of the window on the grid, so they are also functions of the space-like variables (x , y).

We need the following codes and afterwards computations to be made.

PROBLEM 1. Fix the time variables (S,T). (1a) Graphs of the scalar functions E_1, E_2, and E_{12} as a function of the pairs (x,y) for each space variable that is given on the grid of measurement channels. For Example I, we have each (x,y) given as the indices of the array of size (K,L) that are positions of the electrodes. For Example II, (x,y) are the indices of positions of the pixels on the image from one video frame. These surfaces are graphed over the corresponding grid. (1b) As each variable S or T varies (keeping the other one fixed), we get snap-shots in time of these surfaces. Animate the time variation of the surfaces defined by E_1, E_2, and E_{12} over the grids. Each case will have a separate animation.

PROBLEM 2. Fix the time variables (S ,T). The unit vectors U_1, U_2, and U_{12} are also functions of (x,y) on the grid, and they can be used to define unit vector fields on the grid. We would like to scale these vector fields by their corresponding scalar functions E_1, E_2, and E_{12}. Let $Y_1=E_1*U_1$, $Y_2=E_2*U_2$, and $Y_{12}=E_{12}*U_{12}$. (2a) Graph the vector fields on the grid (each separately graphed). (2b) As each time variable S or T varies, we get a vector field. Animate the time variation of these vector fields defined by each of the Y_1, Y_2, and Y_{12}.

PROBLEM 3. Fix the time variable T, so every function is a function of time variable S (the place that the time window of fixed duration T is situated and slides along the time-axis). We get a vector field in the 3-dimensional space (x,y,t) that is a discrete sampling of some smooth vector field of a system of ODE. The sampling has been uniform in space (the (x,y) points on the grid) and in time. (3a) Solve this ODE system using the Runge-Kutta method. That is, find the flow line (integral curves) for each initial value (x,y). (3b) We wish to graph a few of these curves in the 3-dimensional space (x,y,t). Write a function that takes the initial values (x,y) and graphs the flow line.