M3/4/5N9 – Project 2
Due 20 Dec 2016

Use the MATLAB format `shorte` when reporting any numerical values.

1. **PageRank Algorithm**
   Google PageRank developed by Larry Page and Sergey Brin revolutionized web search by generating a ranked list of web pages based on the underlying connectivity of the web. The PageRank algorithm is based on an ideal random web surfer who, when reaching a page, goes to the next page by clicking on a link. The surfer has equal probability of clicking any link on the page and, when reaching a page with no links, has equal probability of moving to any other page by typing in its URL. In addition, the surfer may occasionally choose to type in a random URL instead of following the links on a page. The PageRank is the ranked order of the pages from the most to the least probable page the surfer will the viewing.

   Suppose the web is composed of N sites. The probability of the surfer being on each page at time k is given by the state vector $\boldsymbol{x}^{(k)}$, i.e. the probability of being on site j after k steps is the jth component of $\boldsymbol{x}^{(k)}$. Since $\boldsymbol{x}^{(k)}$ contains the probabilities of being on a site for all sites, $\sum_{j=1}^{N} x_j^{(k)} = 1$. With every change of page, the state vector is multiplied by the transition matrix,

   $$\boldsymbol{P} = \alpha \boldsymbol{G} + \alpha \boldsymbol{a}\mathbf{1}^T/N + (1-\alpha)\mathbf{1}\mathbf{1}^T/N \tag{1}$$

   to give the new probability, $\boldsymbol{x}^{(k+1)} = \boldsymbol{P}^T\boldsymbol{x}^{(k)}$ the surfer is on any given page. In the equation, $\boldsymbol{G}$ is the matrix containing the transition probabilities to move to the next page via links, and a site i has links if the $\sum_{j=1}^{N} G_{ij} = 1$. The entries of the vector $\boldsymbol{a}$ are 1 if the the page has no links and 0 otherwise, and the term containing $\boldsymbol{a}$ gives the transition probabilities to move to the next page for pages that have no links. Finally, the matrix $\mathbf{1}\mathbf{1}^T/N$, where $\mathbf{1}$ is the vector with each component equal to 1, are the transition probabilities to move to the next page using solely the URL. The value $\alpha \in [0, 1)$ controls the probability of using one transition approach over the other. Thus, $\sum_{j=1}^{N} P_{ij} = 1$ for each i. PageRank continues to update the state vector until $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2$ is within a prescribed tolerance. Since Google regularly ranks 4.3 billion pages (thus $\boldsymbol{P}$ is a $4.3\text{e}9 \times 4.3\text{e}9$ matrix) this computation must be performed as efficiently as possible.

   (a) Based on properties of $\boldsymbol{P}^T$, it can be shown that $\boldsymbol{x}^{(k)}$ converges to a unique $\boldsymbol{x}$ as $k \to \infty$. Explain what algorithm PageRank is actually using and explain why the final state vector $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{P}^T$. What is the corresponding eigenvalue?

   (b) The accompanying file `G.mat` contains the matrix $\boldsymbol{G}$ for a subset of the web (9964 pages) a number of years ago. Construct the matrix $\boldsymbol{P}^T$ and implement the PageRank algorithm to find the PageRank state if $\alpha = 0.85$, the tolerance set to 10^{-8}, and initial state vector $\boldsymbol{x}^{(0)} = \boldsymbol{e}_1$. Generate the list the top 50 sites by their index (you may use `sort` to help generate the list).

   (c) The parameter α and the URL transition matrix must be introduced to ensure convergence to a unique $\boldsymbol{x}$. In their original paper, Page and Brin settled on the value of $\alpha = 0.85$, but it's not immediately clear why they chose this value and not another. Perform numerical experiments to study how PageRank performs for different values of α. In particular, note how the final state vector $\boldsymbol{x}$ and the number of iterations needed to converge vary with α. Discuss your findings in your report and give some indication why the value of $\alpha = 0.85$ was originally chosen. Based on your results, what can you say about how the eigenvalues of $\boldsymbol{P}^T$ depend on α?

   (d) Since many pages don't link to any other pages, and those that do have links only link to a handful of pages, the matrix $\boldsymbol{G}$, is sparse (you can visualize the sparsity using `spy`) and matrix-vector multiplication involving $\boldsymbol{G}^T$ can be done rapidly. MATLAB will do this rapid matrix-vector multiplication

automatically if $\boldsymbol{G}^T$ is a sparse data structure. This can be achieved by setting `GT = sparse(G')`. Explain how to take advantage of this in the PageRank algorithm and implement it in MATLAB. Demonstrate the decrease in runtime.

2. **Principal Component Analysis - Eigenworm**
Dimensional reduction is an important part of analysing any large dataset. Consider an $m \times n$ matrix $\boldsymbol{X}$ where the rows contain data measured at n points at different times. One technique to analyse this data is principal component analysis (PCA) which finds the largest eigenvalues and eigenvectors of the covariance matrix $\boldsymbol{C} = \boldsymbol{X}^T\boldsymbol{X}$. Thus, the matrix $\boldsymbol{C}$ is symmetric by construction. In this context, the eigenvectors, $\boldsymbol{v}$, are referred to as the principal components and the eigenvalues, λ, provide the variance in the direction of the principal component. To reduce the dimension of the data, one can then simply compute the principal components, then project the data onto the p components with the highest variance.

For this problem, we will use data collected from the crawling worm, *C. elegans* (believe it or not, three nobel prizes have been awarded for research involving this organism). PCA is now a regular technique (see attached reference *Dimensionality and Dynamics in the Behavior of C. elegans* by Stephens, Johnson-Kerner, Bailek, and Ryu) used to analyse its motion and often is used to help link behaviour with genetic variations between mutants. The data matrix $\boldsymbol{\Theta}$ in the associated file `theta.mat` gives the worm's bending angle at $n = 49$ points along its length measured at $m = 26996$ points in time. PCA will pick out a natural basis (the principal components) for the worm shapes based on those that yield the highest variance in the data. The hope is that very few principal components are needed to reproduce all of the observed shapes allowing for easier interpretation of the experimental data.

(a) Compute $\boldsymbol{C}$ and modify the code `house` to turn $\boldsymbol{C}$ into a tridiagonal matrix $\boldsymbol{T}$ via similarity transformations. You will receive *most* marks for using the general algorithm introduced in lecture, but for *full* marks you must take full advantage of symmetry and sparsity in your code. Provide the non-zero entries of $\boldsymbol{T}$ in your report.

(b) Use the QR algorithm to find the eigenvalues of $\boldsymbol{C}$ and provide the values of the 10 most dominant eigenvalues in your report. Use a tolerance 10^{-14} to determine when an off-diagonal component is zero and an eigenvalue has been found. Full marks will be awarded only if your code uses shifts and takes advantage of the structure of $\boldsymbol{T}$. Based on the magnitude of the values, do you think that PCA will help us reduce the dimension of the data?

(c) Find the principal components (eigenvectors) associated with the 10 most dominant eigenvalues. There are different ways of doing this and the choice is up to you, however, for whichever route you choose, you may only use elementary MATLAB operations in you code. Also, if you need to solve a linear system you cannot use backslash, and must solve the system using LU decomposition for full marks. Make a single plot (line plots of the entries vs index) of the principal components associated with the 4 most dominant eigenvalues and provide their numerical values in the report.

(d) Compute the projection, $\boldsymbol{\theta}_p$, of the first row of $\boldsymbol{\Theta}$ onto the space spanned by the first p principal components for $p = 1$ to 10. Provide a table of the projection coefficients for the case $p = 10$. Make a single plot showing the projected data for $p = 2, 4, 6$ and 10, as well as the original data. Make a table the error $\|\boldsymbol{\theta} - \boldsymbol{\theta}_p\|_2$ where $\boldsymbol{\theta}$ is the first row of $\boldsymbol{\Theta}$, for $p = 1$ to 10. Discuss your results and provide some insight into how to choose p.

3. **Mastery**

A study of the various direct methods for solving linear systems via matrix factorization could be viewed as a tour of MATLAB's backslash. In lectures, we only explored in detail the first (QR) and last (LU) stops of backslash as they are the most general methods that apply to virtually all linear systems of interest. The stops backslash takes along the way between these two points explores if the matrix has a particular structure or certain properties allowing either LU to be performed more rapidly or another closely related algorithm to be used instead. These algorithms are covered in Chapter 4 of *Matrix Computations* by Golub and Van Loan (attached).

Take again the linear system $-\boldsymbol{D}_2\boldsymbol{u}_N = \boldsymbol{f}_N$ from Project 1 and consider the permutation matrix

$$\boldsymbol{P} = [\boldsymbol{e}_1\ \boldsymbol{e}_N\ \boldsymbol{e}_2\ \boldsymbol{e}_{N-1}\ \boldsymbol{e}_3 \dots] \tag{2}$$

where $\boldsymbol{e}_i$ is the ith column of identity. Since $\boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}$, we have that

$$-(\boldsymbol{P}^T\boldsymbol{D}_2\boldsymbol{P})(\boldsymbol{P}^T\boldsymbol{u}_N) = (\boldsymbol{P}^T\boldsymbol{f}_N). \tag{3}$$

(a) For this problem, you are backslash. Using Chapter 4, select the best algorithm to solve Eq. (3) for $\boldsymbol{u}_N$. In your report, discuss and justify your choice.

(b) Implement the algorithm, and solve the linear system for u_N for $N = 16$, 32, 64 and 128, again taking $f_{N,i} = \sin(x_i)$. Explain why this equation can be solved even though we know $\boldsymbol{D}_2$ is singular. As in project 1, provide the error, E_N, to ensure your code is working properly and use `tic` and `toc` to obtain runtimes.

(c) Compare your choice of algorithm with those used in Project 1. Is it as fast or even faster than the FFT approach? Discuss your comparison and provide an explanation of the advantages of your chosen algorithm. Also, describe a situation where your chosen algorithm would certainly fail to be the best one!