Project 4

Eigenvectors and Google’s PageRank

1. Using the eigen-decomposition \( A = U \Lambda U^{-1} \), build a matrix \( A \) that has eigenvalues \( \lambda_1 = 2 \) and \( \lambda_2 = 1 \) and associated eigenvectors \( \vec{v}_1 = (2, 1)^T \) and \( \vec{v}_2 = (1, -1)^T \).

2. Fit the polynomial \( y = \theta_0 + \theta_1 x + \theta_2 x^2 \) to the data points \((x, y) = (1, 4), (2, 6), (3, 15), (4, 19)\) by setting up a system of equations \( A \vec{\theta} = \vec{b} \) where \( A \) is \( 4 \times 3 \) and \( \vec{b} \) is \( 4 \times 1 \) and \( \vec{\theta} = (\theta_0, \theta_1, \theta_2)^T \). Find the total least squares solution using the singular value decomposition (SVD) \([U, S, V] = \text{svd}([A \ b])\); in MATLAB.

Make a plot in MATLAB showing the four data points and the linear least squares polynomial fit from Project 3 (last week) and the total least squares polynomial. Explain the difference between the linear least squares and total least squares fit in your plot.

3. Notice that if \([A \ b] = USV^T\) then \([A \ b]^T [A \ b] = VSU^T USV^T = VS^2 V^T\). Compute \( S \) and \( V \) from Step 1 by finding the eigenvectors and eigenvalues of \([A \ b]^T [A \ b]\) using Algorithm 4.6 Orthogonal Iteration on page 181.

4. Download “surfer2.m” and type \([U, G] = \text{surfer2}('http://math.gmu.edu',100)\) to build a graph which represents the links between 100 websites starting from the GMU homepage (or any other appropriate webpage of your choice). Watch the GUI output and hit the ‘skip’ button if the algorithm gets stuck on a website. The 100 websites selected will be stored in \( U \) and the graph of hyperlink-connectivity will be stored in \( G \) which is a \( 100 \times 100 \) sparse matrix of zeros and ones. Convert \( G \) to a full matrix with double precision values using \( G = \text{full}(	ext{double}(G)) \);

5. We are going to convert \( G \) into a probability matrix, in order to insure that we can get to any website in the network form any other website, add a small constant to every entry, \( G = G + 1/100 \). Now, convert \( G \) into a (left) stochastic matrix \( P \) by dividing each column by the column sum \( P_{ij} = \frac{G_{ij}}{\sum_{k=1}^{100} G_{kj}} \). We can think of \( P_{ij} \) as the probability of a random surfer moving from the \( j \)-th page to the \( i \)-th page in a single step.

6. We are interested in the so-called ‘invariant measure’ of this random walk, meaning how long does the random surfer spend at each website on average. For a randomly chosen initial vector \( \vec{v} \) we are looking for \( \vec{v}^\infty = \lim_{k \to \infty} P^k \vec{v} \); notice that \( P \vec{v}^\infty = \vec{v}^\infty \) so \( \vec{v}^\infty \) is an eigenvector with eigenvalue \( \lambda = 1 \). In fact, \( \lambda = 1 \) is the largest eigenvalue of \( P \). Use Normalized Power Iteration (Algorithm 4.2, page 175) to find \( \vec{v}^\infty \). Compute the Rayleigh quotient to check that the eigenvalue is \( \lambda = 1 \).

7. The invariant measure, \( \vec{v}^\infty \) represents the simplified PageRank introduced by Google. Use the command \([\text{pagerank}, \text{pageindex}] = \text{sort}(\vec{v}^\infty, 'descend')\) to sort the pages from largest PageRank to smallest PageRank (insert your chosen variable name for \( \vec{v}^\infty \)). Output the top 10 pages, \( U(\text{pageindex}(1:10)) \) and the bottom 10 pages \( U(\text{pageindex}(\text{end}-9:\text{end})) \). How many of the top 10 sites have you visited? How many of the bottom 10?

Due: Beginning of Class, Sept. 29, 2015