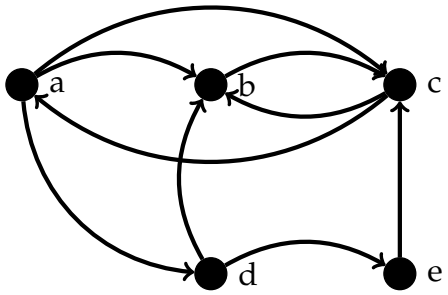
Assignment #9

Due Monday 15, 2021 at the start of class

Please read Lectures 20, 21, 22, 23, and 24 in the textbook *Numerical Linear Algebra* by Trefethen and Bau. Then do the following exercises.

P22. Consider the graph of five web pages a,b,c,d,e shown below, where each page links to some of the other pages as shown. The goal of this problem is to compute the Google PageRank™ of the pages. In fact, this problem is about the essential idea which created Google, only about 25 years ago.



(a) Start by asking Google to document itself. Go to Google Scholar and search for "pagerank citation ranking". The first link, the one with highest PageRank, will be a technical preprint by Page, Brin, Motwani, and Winograd; it has more than 15000 citations. Download the 17 page PDF and read it. It describes a simpler time.

The main ideas are that *a web is a directed graph*, as in the example shown above, that *the limiting probability that a certain random web surfer visits any particular page is its PageRank*, and finally that *a search engine should report results in the PageRank order*.

We can regard PageRank as simply an eigenvector of a certain matrix A, a transition probability matrix for a Markov chain (defined below). To build A we will start with the adjacency (connectivity) matrix G of the web. Here is how to start with a web as a directed graph and construct G, A, and the eigenvector of PageRank values:[1]

> Let W be a connected, directed graph of n webpages.[2] Index the pages as $1, 2, \ldots, n$. Let G be the $n \times n$ connectivity matrix of W, that is,
>
> $$g_{ij} = \begin{cases} 1, & \text{there is a hyperlink (directed edge) to page } i \text{ from page } j, \\ 0, & \text{otherwise.} \end{cases}$$

[1] The story here is distilled from Cleve Moler's note http://www.mathworks.com/company/newsletters/articles/the-world-s-largest-matrix-computation.html. Unfortunately, his note contains a very confusing typo, as well as old/bad typesetting.

[2] Here "connected" for a directed graph means that there is one vertex where, by following enough links, you can reach any other vertex. The node a in the above graph suffices.

The number of nonzeros in G is the total number of hyperlinks in W. Let c_j be the column sums of G:

$$c_j = \sum_{i=1}^{n} g_{ij}.$$

The value c_j gives the *outdegree* of the jth page. Let p be the fraction of the time that the random walk follows some link, so $1-p$ is the fraction of time that an arbitrary page is chosen. Define $\delta = (1-p)/n$. Now let A be the $n \times n$ matrix whose entries are

$$a_{ij} = \frac{p g_{ij}}{c_j} + \delta.$$

The matrix A is the transition probability matrix of the random-surfer Markov chain. An old theorem says that the largest eigenvalue of A is equal to one and the corresponding eigenvector x, which satisfies

$$Ax = x,$$

is unique up to a scaling factor, and has positive entries. Choose the scaling so that

$$1 = \sum_{i=1}^{n} x_i.$$

The entries of x are the PageRanks of the webpages in W.

For a realistic web, G is huge and sparse because most pairs of webpages are not connected by a single link. Google suggests $p = 0.85$. Matrix A is not sparse because most entries are equal to the small constant $\delta > 0$. The "old" Perron-Frobenius theorem says that if all entries in a matrix are positive then all the entries of the eigenvector associated to the largest eigenvalue can be chosen to be positive.

There is nothing to turn in for part **(a)**. The job was to read and think.

(b) Following my description of the process, compute G for the web shown above.

(c) Continuing from my description, build A from G, using $p = 0.85$ as suggested. Finally, use MATLAB's `eig` to compute the PageRanks. Which page gets the highest PageRank? Explain why in intuitive terms.

(d) Draw a new web W with 10 pages. (*That is, draw a connected, directed graph with 10 vertices. I suggest you have only 1 to 4 out-going links for each page.*) Recompute G, A, and the PageRanks.

(Extra Credit) (*Worth a few expensive points. This part shows how PageRank is actually computed. The matrix A is never formed and no eigenvalue algorithm is used.*) Build a random walk program which traverses a web as would a Google "bot". Build-up an estimate of PageRank based on the fraction of the time spent at a given page. It is up to you how to represent a web in some data structure; it could be the matrix G, or not. Check that your surfer converges to the results computed in parts **(c)** and **(d)**.

Exercise 20.1.

Exercise 20.3. Do part (a) only.

Exercise 21.1.

Exercise 21.4. Do part (a) only.