Indian Institute of Science Bangalore
Department of Computational and Data Sciences (CDS)

DS284: Numerical Linear Algebra

Assignment 5 [Posted Nov 6, 2023]

Faculty Instructor: Dr. Phani Motamarri
TAs: Kartick Ramakrishnan, G Sundaresan,
Sayan Dutta

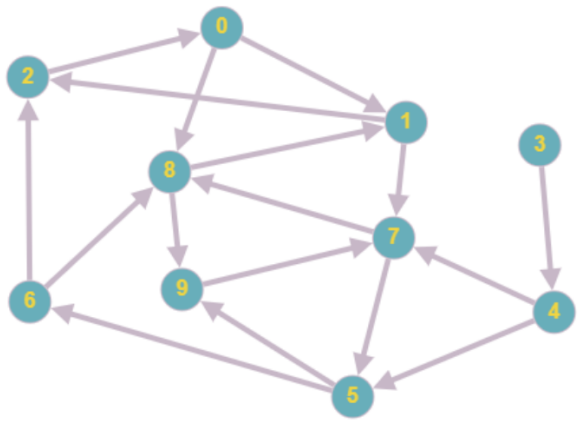
Problem 1

This question will help you to appreciate the usefulness of power iteration in a real-world application. In this question, you need to calculate the Page rank of each of the nodes of the graph given below by first constructing its Markov transition matrix and subsequently, calculating the most dominant eigenvector of this transition matrix i.e the eigenvector corresponding to the largest eigenvalue. Use the power iteration algorithm discussed in the class to compute the dominant eigenvector by implementing it on a computer.

Justify your result showing the following data/figures:

- Plot of the 2-norm of the residual of eigenvalue problem involving the dominant eigenvector with iteration number. (Use the unit vector generated at the end of each iteration in your power iteration algorithm to compute the eigenvalue problem residual at each iteration)
- Plot of the 2-norm of the difference between the vectors corresponding to the successive iterates of your power iteration with number of iterations.
- Compute the Rayleigh quotient at each iteration of your algorithm and plot the convergence with respect to iteration number.
- State the node numbers with least and highest page ranks.

Refer to the write-up below for description about building the Markov Transition matrix of a given graph.



PageRank Algorithm - The Mathematics of Google Search

We live in a computer era. Internet is part of our everyday lives and information is only a click away. Just open your favourite search engine, like Google, AltaVista, Yahoo, type in the key words, and the search engine will display the pages relevant for your search. But how does a search engine really work?

At first glance, it seems reasonable to imagine that what a search engine does is to keep an index of all web pages, and when a user types in a query search, the engine browses through its index and counts the occurrences of the key words in each web file. The winners are the pages with the highest number of occurrences of the key words. These get displayed back to the user.

This used to be the correct picture in the early 90s, when the first search engines used *text based ranking systems* to decide which pages are most relevant to a given query. There where however a number of problems with this approach. A search about a common term such as "Internet" was problematic. The first page displayed by one of the early search engines was written in Chinese, with repeated occurrences of the word "Internet" and containing no other information about the Internet. Moreover, suppose we wanted to find some information about Cornell. We type in the word "Cornell" and expect that "www.cornell.edu" would be the most relevant site to our query. However there may be millions of pages on the web using the world Cornell, and www.cornell.edu may not be the one that uses it most often. Suppose we decided to write a web site that contains the word "Cornell" a billion times and nothing else. Would it then make sense for our web site to be the first one displayed by a search engine? The answer is obviously no. However, if all a search engine does is to count occurrences of the words given in the query, this is exactly what might happen.

The usefulness of a search engine depends on the *relevance* of the result set it gives back. There may of course be millions of web pages that include a particular word or phrase; however some of them will be more relevant, popular, or authoritative than others. A user does not have the ability or patience to scan through all pages that contain the given query words. One expects the relevant pages to be displayed within the top 20-30 pages returned by the search engine.

Modern search engines employ methods of ranking the results to provide the "best" results first that are more elaborate than just plain *text ranking*. One of the most known and influential algorithms for computing the relevance of web pages is the Page Rank algorithm used by the Google search engine. It was invented by Larry Page and Sergey Brin while they were graduate students at Stanford, and it became a Google trademark in 1998. The idea that Page Rank brought up was that, the importance of any web page can be judged by looking at the pages that link to it. If we create a web page i and include a hyperlink to the web page j, this means that we consider j important and relevant for our topic. If there are a lot of pages that link to j, this means that the common belief is that page j is important. If on the other hand, j has only one backlink, but that comes from an authoritative site k, (like www.google.com, www.cnn.com, www.cornell.edu) we say that k transfers its authority to j; in other words, k asserts that j is important. Whether we talk about popularity or authority, we can iteratively assign a rank to each web page, based on the ranks of the pages that point to it.

To this aim, we begin by picturing the Web net as a directed graph, with nodes represented by web pages and edges represented by the links between them.

Suppose for instance, that we have a small Internet consisting of just 4 web sites www.page1.com, www.page2.com, www.page3.com, www.page4.com, referencing each other in the manner suggested by the Fig. 1:

We "translate" the picture into a directed graph with 4 nodes, one for each web site. When web site i references j, we add a directed edge between node i and node j in the graph. For the purpose of computing their page rank, we ignore any navigational links such as back, next buttons, as we only care about the connections between different web sites. For instance, Page1 links to all of the other pages, so node 1 in the graph will have outgoing edges to all of the other nodes.

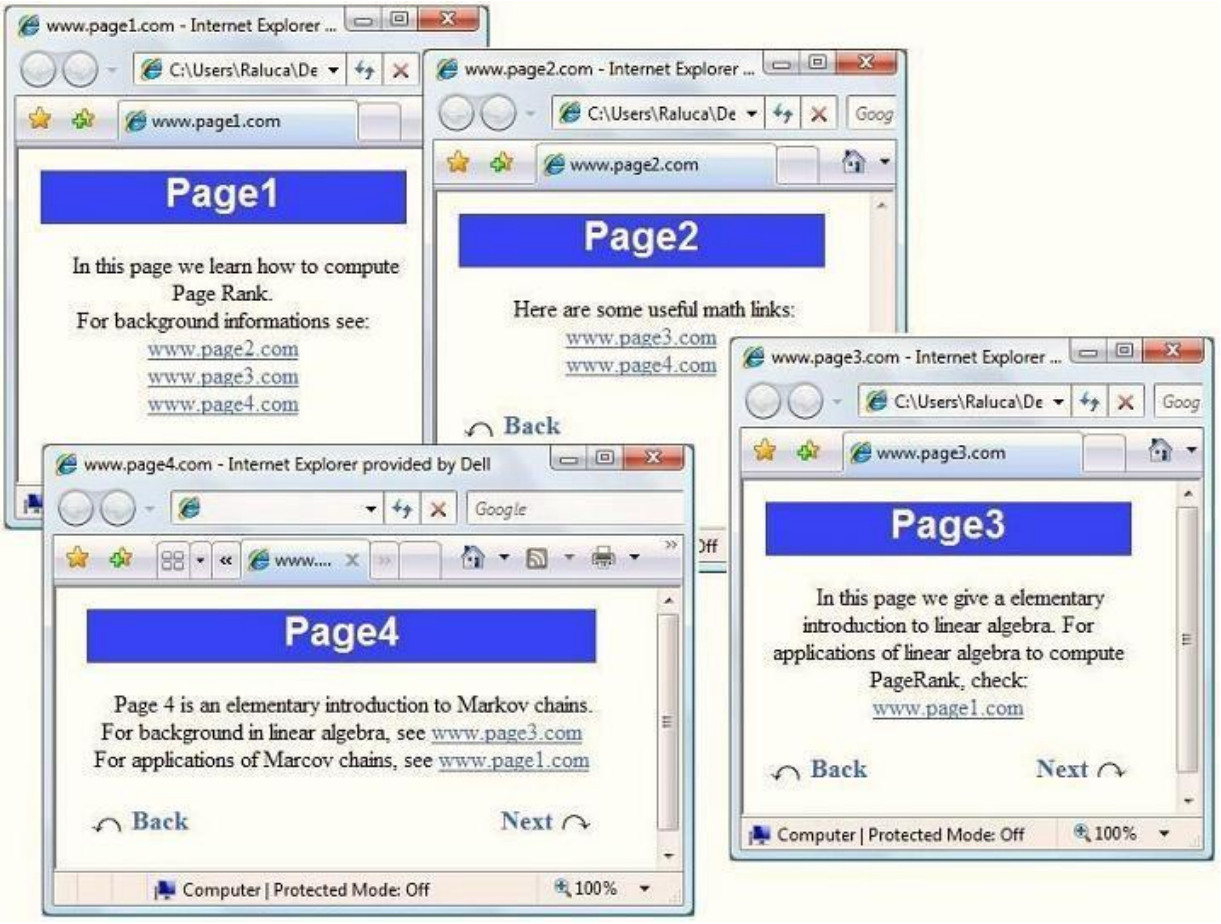
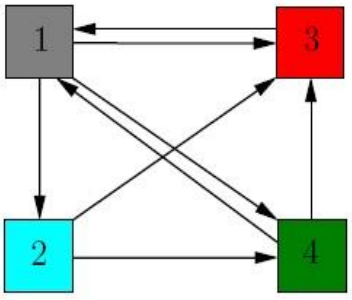
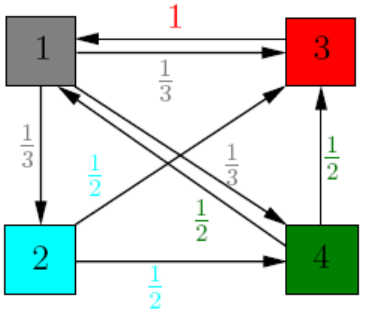


Fig. 1

Page3 has only one link, to Page 1, therefore node 3 will have one outgoing edge to node 1. After analyzing each web page, we get the following graph:



In our model, each page should transfer evenly its importance to the pages that it links to. Node 1 has 3 outgoing edges, so it will pass on 1/3 of its importance to each of the other 3 nodes. Node 3 has only one outgoing edge, so it will pass on all of its importance to node 1. In general, if a node has *k* outgoing edges, it will pass on 1/k of its importance to each of the nodes that it links to. Let us better visualize the process by assigning weights to each edge.



Let us denote by A the transition matrix of the graph, $A =$ $\begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix}$

Linear algebra point of view:

Let us denote by x_1, x_2, x_3, and x_4 the importance of the four pages. Analyzing the situation at each node we get the system:

$$\begin{cases} x_1 = 1 \cdot x_3 + \frac{1}{2} \cdot x_4 \\ x_2 = \frac{1}{3} \cdot x_1 \\ x_3 = \frac{1}{3} \cdot x_1 + \frac{1}{2} \cdot x_2 + \frac{1}{2} \cdot x_4 \\ x_4 = \frac{1}{3} \cdot x_1 + \frac{1}{2} \cdot x_2 \end{cases}$$

This is equivalent to asking for the solutions of the equations $A \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$.

We know that the eigenvectors corresponding to the eigenvalue 1 are of the form $c \cdot \begin{bmatrix} 12 \\ 4 \\ 9 \\ 6 \end{bmatrix}$.

Since PageRank should reflect only the relative importance of the nodes, and since the eigenvectors are just scalar multiples of each other, we can choose any of them to be our PageRank vector. Choose v^* to be the unique eigenvector with the sum of all entries equal to 1. (We will sometimes refer to it as the probabilistic eigenvector corresponding to the eigenvalue 1). The eigenvector $\frac{1}{31} \cdot \begin{bmatrix} 12 \\ 4 \\ 9 \\ 6 \end{bmatrix} \sim \begin{bmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{bmatrix}$

is our PageRank vector.

Problem 2

Assert if the following statements are True or False. Give a detailed reasoning for your assertion.

(a) An eigenvalue solver can be designed to compute eigenvalues and eigenvectors of a given matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ in a finite number of steps, using exact arithmetic.

(b) An eigenvalue solver designed to compute all eigenvalues and eigenvectors of a symmetric dense matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ requires at most $O(m^3)$ work, if it is not initially reduced to tri-diagonal form in Phase 1.

(c) Power iteration produces a sequence of vectors $\mathbf{v}^{(i)}$ that converges to the eigenvector corresponding to the largest eigenvalue of $\mathbf{A} \in \mathbb{R}^{m \times m}$ starting with any initial guess vector $\mathbf{v}^{(0)} \neq \mathbf{0}$.

(d) Let $\mathbf{F} \in \mathbb{R}^{m \times m}$ denote the Householder reflector that introduces zeros below the diagonal entry of symmetric matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ in the 1st column when pre-multiplied with $\mathbf{A}$. Then eigenvalues of $\mathbf{FAF}^T$ and $\mathbf{A}$ are the same.

Problem 3

Assert if the following statements are True or False. Give a detailed reasoning for your assertion. Marks will be awarded only for your reasoning.

(a) Pure QR algorithm is equivalent to the Simultaneous iteration applied to an initial guess of vectors which are columns of a square full rank matrix.

(b) Pure QR algorithm used to solve the eigenvalues and eigenvectors of a matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ generates a sequence of matrices $\mathbf{Q}^{(k)}$ which converges to the eigenvector matrix of $\mathbf{A}$ as $k \to \infty$.

(c) Pure QR algorithm used to solve the eigenvalues and eigenvectors of a matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ generates QR factorization to be $\mathbf{A}^k = \mathbf{Q}^{(k)}\mathbf{R}^{(k)}$ generated at the k^{th} iteration of the algorithm.

(d) Computational complexity for finding the eigenvectors using Pure QR algorithm and the Simultaneous iteration is same.

Problem 4

We are usually confronted with large sparse matrix eigenvalue problems arising from the discretization of a partial differential equation (eigenproblem), where the unknown eigenfunctions are approximated in a finite-dimensional subspace as a linear combination of localized basis functions spanning the subspace. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times m}$ be two such large symmetric positive definite sparse matrices and the problem of finding eigenvector, eigenvalue pairs $(\lambda_i, \mathbf{u}_i)$ satisfying the equation

$$\mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{B}\mathbf{u}_i \tag{1}$$

is called a *generalized* eigenvalue problem (Note: The standard eigenvalue problem you are familiar with is a special case having $\mathbf{B} = \mathbf{I}$).

(a) Rewrite the generalized eigenvalue problem in equation (1) as a standard eigenvalue problem $\mathbf{H}\mathbf{v}_i = \lambda_i \mathbf{v}_i$ and in doing so, ensure that $\mathbf{H}$ is a symmetric matrix. Express $\mathbf{u}_i$ in terms of $\mathbf{v}_i$. (Hint: Use the property of $\mathbf{B}$)

(b) Devise an iterative algorithm to find the eigenvalue λ_i closest to 2.0 (assume 2.0 is not an eigenvalue of equation (1)) and a corresponding eigenvector. Will your algorithm always converge, or is there any condition that needs to be satisfied for convergence?

(c) Identify the computationally dominant step in your algorithm and describe your method of choice for performing this step.

(d) How will you modify the algorithms proposed in parts (b) and (c) if you are told that the eigenvalue closest to 2.0 is, in fact, the lowest eigenvalue? What do you gain in doing so?

Problem 5

If $\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 2 & 3 & 3 \end{bmatrix}$, answer the following questions:

(a) Write the characteristic equation associated with the above matrix $\mathbf{A}$ and subsequently compute its eigenvalues

(b) Set up the Arnoldi iteration with the starting vector $\mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ to find the orthonormal basis vectors $\{\mathbf{q}_1, \mathbf{q}_2\}$ spanning the two dimensional Krylov subspace $\mathcal{K}_2 =< \mathbf{b}, \mathbf{A}\mathbf{b} >$

(c) Find the orthogonal projection $\mathbf{H}$ of $\mathbf{A}$ onto $\mathcal{K}_2$ represented in the basis $\{\mathbf{q}_1, \mathbf{q}_2\}$ obtained in (b) above, and then compute the eigenvalues of this $\mathbf{H}$ (also called Ritz values). Find the absolute error between the smallest Ritz value and the smallest eigenvalue of $\mathbf{A}$ and similarly compute the absolute error between the largest Ritz value and largest eigenvalue of $\mathbf{A}$.

(d) Consider the system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 2 & 3 & 3 \end{bmatrix}$ as given above and $\mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ as given in (b). Find the exact solution $\mathbf{x}^*$ which solves $\mathbf{A}\mathbf{x} = \mathbf{b}$ using forward substitution. Subsequently find the vector $\hat{\mathbf{x}} \in \mathcal{K}_2$ that minimizes the norm $\|\mathbf{A}\mathbf{c} - \mathbf{b}\|_2$ over all possible vectors $\mathbf{c} \in \mathcal{K}_2$, where $\mathcal{K}_2$ is the Krylov subspace constructed in (b). Finally, find the norm of error between exact solution $\mathbf{x}^*$ and $\hat{\mathbf{x}}$.

Problem 6

Consider the eigenvalue problem corresponding to a large sparse and diagonally dominant symmetric matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$. Let us say, we are interested in solving $\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{x}_i$ for $i = 1 \cdots n$ smallest eigenvalue and eigenvector pairs of $\mathbf{A}$ ($n << m$). Answering the following questions will make you deduce an iterative algorithm for solving this eigenvalue problem different from the methods discussed in the class.

(a) Let us begin the iteration $k = 0$ with a trial guess of orthogonal vectors spanning the n-dimensional subspace $\mathbb{V}^n_{(0)} = \{\widetilde{\mathbf{x}}^{(0)}_1, \widetilde{\mathbf{x}}^{(0)}_2, \ldots, \widetilde{\mathbf{x}}^{(0)}_n\}$. Assume each of these vectors act as an approximation to the corresponding exact eigenvectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ of the matrix $\mathbf{A}$. Let $\mathbf{t}_i = \mathbf{x}_i - \widetilde{\mathbf{x}}^{(0)}_i$ for $i = 1, 2, \ldots, n$ denote the correction vectors to the approximate eigenvectors. Show that $\mathbf{t}_i$ can be computed by solving the corrector equation $(\mathbf{A} - \epsilon_i \mathbf{I})\, \mathbf{t}_i = (\epsilon_i \mathbf{I} - \mathbf{A})\, \widetilde{\mathbf{x}}^{(0)}_i$, where ϵ_i is the exact eigenvalue of $\mathbf{A}$. Assuming that you know the exact eigenvalue ϵ_i, can this corrector equation be solved for $\mathbf{t}_i$?

(b) *A priori* one does not know the value of ϵ_i. In this regard what is the best approximation that can be used for ϵ_i and why? (Hint:- Think how the space $\mathbb{V}^n_{(0)}$ is constructed). If ϵ_i is approximated to be $\widetilde{\epsilon}^{(0)}_i$, then the corrector equation becomes $(\mathbf{A} - \widetilde{\epsilon}^{(0)}_i \mathbf{I})\mathbf{t}_i = (\widetilde{\epsilon}^{(0)}_i \mathbf{I} - \mathbf{A})\widetilde{\mathbf{x}}^{(0)}_i$. To solve for $\mathbf{t}_i$ efficiently, one approximates $\mathbf{A}$ in L.H.S of the above corrector equation to the matrix $\mathbf{D} = diag(\mathbf{A})$. To this end, write an expression for solution to the corrector equation (let $\widetilde{\mathbf{t}}_i$ denote this solution vector, an approximation to the correction vector $\mathbf{t}_i$).

(c) Now we construct a $2n$ dimensional space $\mathbb{V}^{2n}_{(0)} = \{\widetilde{\mathbf{x}}^{(0)}_1, \widetilde{\mathbf{t}}^{(0)}_1, \widetilde{\mathbf{x}}^{(0)}_2, \widetilde{\mathbf{t}}^{(0)}_2, \ldots, \widetilde{\mathbf{x}}^{(0)}_n, \widetilde{\mathbf{t}}^{(0)}_n\}$. Argue why is $\mathbb{V}^{2n}_{(k)}$ a better subspace than $\mathbb{V}^n_{(k)}$ to look for eigenvectors of $\mathbf{A}$ at any given iteration k.

(d) We now look for approximate eigenvector, eigenvalue pair of $\mathbf{A}$ in the space $\mathbb{V}^{2n}_{(0)}$. Let us denote $\widetilde{\mathbf{x}}^{(1)}_i \in \mathbb{V}^{2n}_{(0)}$ for $i = 1, 2, \ldots, n$ to be the eigenvector approximations we seek to find in $\mathbb{V}^{2n}_{(0)}$. Define the residual vector $\mathbf{r}_i = \mathbf{A}\widetilde{\mathbf{x}}^{(1)}_i - \widetilde{\epsilon}^{(1)}_i \widetilde{\mathbf{x}}^{(1)}_i$ where $\widetilde{\epsilon}^{(1)}_i$ is the best approximation to the eigenvalue corresponding to the eigenvector approximation $\widetilde{\mathbf{x}}^{(1)}_i$. This approximate eigenvector, eigenvalue pair $(\widetilde{\mathbf{x}}^{(1)}_i, \widetilde{\epsilon}^{(1)}_i)$ is obtained by imposing the Galerkin condition that states that $\mathbf{r}_i$ is orthogonal to the space $\mathbb{V}^{2n}_{(0)}$. Mathematically deduce the consequences of the imposition of this Galerkin condition and subsequently elaborate how should one go about finding $(\widetilde{\mathbf{x}}^{(1)}_i, \widetilde{\epsilon}^{(1)}_i)$ after imposing the Galerkin condition. Finally, $(\widetilde{\mathbf{x}}^{(1)}_i, \widetilde{\epsilon}^{(1)}_i)$ forms the eigenvector, eigenvalue approximations for $k = 1$ iteration.

Note: Once $(\widetilde{\mathbf{x}}^{(1)}_i, \widetilde{\epsilon}^{(1)}_i)$ is obtained from $\mathbb{V}^{2n}$, the trial subspace gets updated to $\{\widetilde{\mathbf{x}}^{(1)}_i, \widetilde{\mathbf{t}}^{(1)}_i\}$ to seek $\{\widetilde{\mathbf{x}}^{(2)}_i, \widetilde{\mathbf{t}}^{(2)}_i\}$ and iterations are continued till convergence is reached.