CS 377P Spring 2018
Assignment 2: Algorithms
Due: 11:59 PM, February 19th, 2018

February 13, 2018

You should do this assignment in teams of two.
Submissions can be at most 1 day late with 10% penalty.

Some of the problems in this assignment ask you to write code in MATLAB, which is available on the public machines in the CS department. You can also use Octave, which is free software and is similar to MATLAB, or Mathematica.

1. (Iterative solution of linear systems, MATLAB code, 15 points) Consider the linear system

   $$3x - 4y = -1$$
   $$x + 2y = 3$$

   (a) (5 points) Write down the recurrence relation that corresponds to solving this system using the Jacobi method, starting with the initial approximation $(x_1 = 0, y_1 = 0)$. Use the first equation to refine the approximation for x and the second equation to refine the approximation for y. Express this recurrence as a computation involving matrices and vectors.

   (b) (5 points) Compute the first 25 approximations (x_i, y_i) and plot a 3D plot (x, y, i) in which the z-axis is the iteration number i. Give an intuitive explanation of this 3D plot.

   (c) (5 points) Repeat these two parts for the Gauss-Seidel method. You can find a description of the Gauss-Seidel method online.

2. (ODE's, no code to submit, 20 points)) Consider the ordinary differential equation

   $\frac{dy}{dx} = y * sin(x)$

   with initial condition $y(0) = 1$ in the interval $0 < x < 2$. Use finite-differences to find an approximate solution to this differential equation, using a step size $h = 0.1$, as specified below.

(a) (10 points) Recall that the forward-Euler approximation of the derivative is the following:
$\frac{dy}{dx}|_{i*h} \approx \frac{y_f((i+1)*h)-y_f(i*h)}{h}$
What is the recurrence for computing the Forward-Euler approximation to the solution of this ode? Draw a graph of this solution.

(b) (10 points) Recall that the centered-difference approximation of the derivative is the following:
$\frac{dy}{dx}|_{i*h} \approx \frac{y_c((i+1)*h)-y_c((i-1)*h)}{2h}$.
Write down the corresponding recurrence formula and draw a graph of this solution.

Hint: the differential equation tells you the derivative at 0 is 0. Use this fact to argue that $y_c(h) = 1$, and then use the recurrence equation to compute the rest of the values.

3. (PDE's, MATLAB code, 30 points) In this problem, we will solve the one-dimensional diffusion equation, which models how heat spreads through a material of uniform conductivity and similar problems. The diffusion equation is usually written as follows

$\frac{\delta f}{\delta t} = D\frac{\delta^2 f}{\delta x^2}$

The solution $f(x,t)$ depends on both x and t. Assume that we have a rod of length 10 meters, and that the two ends of the rod are kept at a fixed temperature of 0^oC, so $f(0,t) = 0.0$ and $f(10,t) = 0.0$. Assume that initially the temperature in the interior of the rod is $f(x,0) = e^{-4(x-5)^2}$.

The approximate solution $\hat{f}$ can be thought of a two-dimensional array in which $\hat{f}(i,j) \approx f(i\Delta x, j\Delta t)$. Intuitively, $\hat{f}(i,j)$ is the computed solution after j time steps at a spatial position i spatial steps away from the origin.

Compute the array $\hat{f}$ using the following discretization scheme, which discretizes time using Forward-Euler and discretizes space using centered-differences.

$\Delta x = 0.25$
$\Delta t = 0.025$
$D = 1$
$0 \leq x \leq 10$
$0 \leq t \leq 10$

$\frac{\delta f}{\delta t}|_{(i*\Delta x, j*\Delta t)} \approx \frac{\hat{f}(i,j+1)-\hat{f}(i,j)}{\Delta t}$ (Forward-Euler)
$\frac{\delta^2 f}{\delta x^2}|_{(i*\Delta x, j*\Delta t)} \approx \frac{\hat{f}(i+1,j)-2*\hat{f}(i,j)+\hat{f}(i-1,j)}{(\Delta x)^2}$ (Centered differences)

(a) Draw the stencil for this discretization scheme.

(b) Use the recurrence equation to find an approximate solution to this problem. Plot the temperature distribution along the rod for different time steps on the same graph (so the x-axis is the distance from the

origin in the rod and the y-axis is the temperature). Does this graph jive with your intuition?

(c) Increase the time step to 0.050 and repeat the previous steps. Explain your observations briefly.

4. (Page-rank, C++ code, 35 points) In this problem, you will write two implementations of the page-rank algorithm described in class and compute the page-rank values for the small example graph from Wikipedia shown on Page 7 of the online lecture slides as well as for a power-law graph called rmat15.

   (a) (5 points) Write a C++ routine that reads in a DIMACS graph from a file and constructs its CSR representation in memory. It should also construct the CSR representation for the transpose of the graph. The DIMACS format is specified at the end of this assignment. For unweighted graphs, you can assume that the edge weights in the file will be 0.

   (b) (15 points) Implement the pull-style page-rank algorithm discussed in class. It should operate on a transposed graph stored in CSR format in memory. Recall that in a pull-style algorithm, the operator writes to a field of the active node and can only read from fields of other nodes in its neighborhood. Compute the page-rank values for the nodes in the Wikipedia graph and the rmat15 graph.

   (c) (15 points) Modify your implementation to produce a push-style page-rank algorithm. It should operate on a graph stored in CSR format in memory. Recall that in a push-style algorithm, the operator may write to fields of any node in its neighborhood. Compute the page-rank values for the nodes in the Wikipedia graph and the rmat15 graph.

   Some additional specifications:

   (a) Convergence: terminate the page-rank iterations when no node changes its page-rank value by more than 10^{-4} between successive iterations.

   (b) After the page-rank iteration is done, scale the page-rank values of all nodes so that their sum is one.

   (c) One of the graphs that is provided to you is the Wikipedia graph used in lecture to illustrate page-rank. Verify the correctness of your implementation using this graph before running page-rank on the bigger graph we have provided.

What to turn in: Create one .tar.gz file with your answers, code and graphs for all problems, and submit to Canvas. Include a README.txt in the tarball to explain how to run your code and what the output will be. No credit will be given for problems in which you are asked to write code unless you include your code and we can run it.

DIMACS format:

One popular format for representing directed graphs as text files is the DIMACS format (undirected graphs are represented as a directed graph by representing each undirected edge as two directed edges). Files are assumed to be well-formed and internally consistent so it is not necessary to do any error checking. A line in a file must be one of the following.

- Comments. Comment lines give human-readable information about the file and are ignored by programs. Comment lines can appear anywhere in the file. Each comment line begins with a lower-case character c.

  c This is an example of a comment line.

- Problem line. There is one problem line per input file. The problem line must appear before any node or edge descriptor lines. The problem line has the following format.

  p FORMAT NODES EDGES

  The lower-case character p signifies that this is the problem line. The FORMAT field should contain a mnemonic for the problem such as sssp. The NODES field contains an integer value specifying n, the number of nodes in the graph. The EDGES field contains an integer value specifying m, the number of edges in the graph.These two fields tell you how much storage to allocate for the CSR representation of the graph.

- Edge Descriptors. There is one edge descriptor line for each edge the graph, each with the following format. Each edge (s,d,w) from node s to node d with weight w appears exactly once in the input file.

  a s d w

  The lower-case character "a" signifies that this is an edge descriptor line. The "a" stands for arc, in case you are wondering. For graphs with unweighted edges, the weight field in each line will be 0.