UNIVERSITY OF MANITOBA

ECE 7650-T18: Advanced Matrix Algorithms **Fall 2016**

Assignment 3/4: Permutations and Basic Iterative Methods

November 18, 2016

The purpose of this assignment is to implement techniques for constructing symmetric permutations and basic iterative solutions to sparse matrix equations.

Important: The assignment is due December 5, 2016 by 4:30 pm (hard copy) or 11:59 pm (electronic submission).

The assignment should be performed **individually** but feel free to interact with your classmates. **Code should be well documented**.

Introduction

This assignment is focused on implementation adjacency graph traversal reordering permutations, applying them, and solving sparse systems using basic iterative methods.

For problems involving code feel free to use any language you wish. Analysis of your program should include comments related to the limitations of your implementation.

Questions

For testing your programs you are given a driver program that will construct a sparse matrix equation for solving Laplace's equation on a plate - the same system we have used frequently in class. If you are using a language other than MATLAB (and if you have been doing so for previous assignments I strongly encourage you to keep it up) you will need to save the matrix/right-hand-side in appropriate vectors and load them into your external program. You will also need a sparse LU decomposition routine. Of course you could write your algorithms in another language and then dump the permutations to be used in MATLAB.

Q1. 30 marks. Implement the BFS and RCM ordering algorithms. Your approach should not actually construct an object to represent the graph, but should simply use the sparse matrix structure to identify nodes in the graph and connectivity. If using MATLAB you can assume that accessing the connectivity of a sparse matrix graph is equivalent to accessing A(i,:) which, provided the matrix A is sparse, will return only the non-zero entries (with appropriate row/column values) that can be looped over. If using another language you will want to ensure that you have similar access, straightforward if, for example, you are using CSR format.

Q2. 10 marks. Use your permutations obtained in Q1 to apply symmetric permutations to the system matrix and compute the solution using an LU decomposition. Plot the time required as a function of matrix size for each of the following: 1) the original unpermuted matrix, 2) the BFS symmetric permutation, 3) the RCM symmetric permutation and 4) the built-in MATLAB permutation. If you are not using MATLAB you get a "pass" on this last comparison. Make sure to provide some validation that your program works (there is a plotting routine in the provided MATLAB code). You will have to decide on appropriate stopping conditions.

Q3. 40 marks. Implement both a Block Jacobi and block Block Gauss-Seidel iterative technique that supports arbitrary block sizes (no overlap). Using the same matrix equation as in Q2 and without permutation, validate your algorithms. Provide, on a single plot, the convergence for different block sizes for both algorithms run on a fixed, large matrix size. Be sure to specify what you are using to define the error at each iteration. On a separate plot, provide the convergence of both algorithms when the original system is permuted using BFS and RCM for a fixed block size of appreciable size (compared to the matrix dimensions). Comment on the results: does permuting the original system help? Why or why not? Would you expect that it would help if the block-size was 1?

Q*. 0 marks. Can anyone figure out the permutation strategy used by MATLAB to perform the general permuted LU decomposition?

Hand In

Your code, output and the answers to the questions above including any requested plots/ figures in a report.