Project 2: Serial Arnoldi Algorithm

Scientific Supercomputing

Assigned: Nov 16, 2018
Due: Dec 5, 2018

1 Introduction

The objective of this project is to formulate and implement an algorithm to compute a set of eigenvalues for a sparse matrix A. For this, one can use the Arnoldi algorithm in conjunction with the shifted QR iteration. Write and debug a serial Arnoldi algorithm; in the project, for brevity, you will only be required to find a set of eigenvalues (i.e., no eigenvectors) of a sparse matrix A. You may use C, C++, or FORTRAN for this assignment.

2 Implementation

So that we read/write ONLY the non-zero elements in a matrix, your program should be structured such that it reads the matrix from a file that has the following format:


```
ni nj nentries
irow icol value
irow icol value
.
.
.
```

Your code should read this format and then store the matrix in compressed row storage (CRS) form in memory (note that it will likely require two passes through the file). Notice that with CRS, you can't use a standard matrix-vector or matrix-matrix multiplication routine. Although this forces you to create a new sparse matrix multiplication routine, you will naturally avoid the many useless multiplications by zero.

3 Reporting of Results

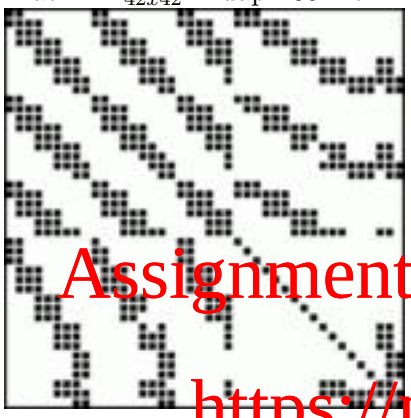
In this project, we will leave the topics of Arnoldi restarting and convergence largely unaddressed. In all items below, the number of Arnoldi iterations k will be specified. So, do not report the eigenvalues found for each run; plot all found eigenvalues each iteration against the iteration number as the result of your run (refer to the text, p. 192, for a sample plot). I will refer to this plot as an "eigenvalue iteration chart" below.

As an aside, it is important to note that, since we are not doing a restarted Arnoldi method, we are running "excessive" numbers of Arnoldi iterations. In real-world problems, we would use a restarted Arnoldi with $k < 20$ in most cases.

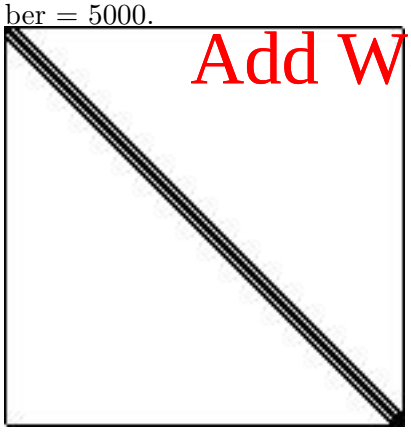
4 Test Matrices

All matrices are given as text files stored in the format described above. These matrices are in "matrices_p2.zip" (16MB file) on Canvas.

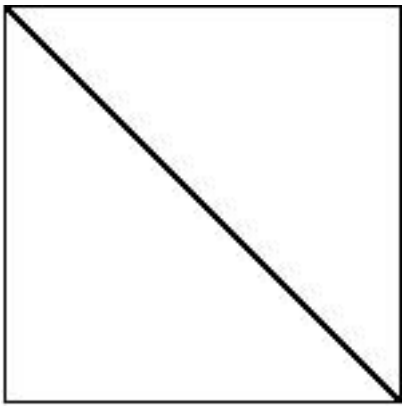
- Matrix A_{6x6}: dense6x6.mtx
- Matrix B_{42x42}: fidapm05.mtx



- Matrix $C_{17821x17821}$ e40r5000.mtx Driven cavity problem, 40x40 elements, Reynolds number = 5000.



- Matrix $D_{90448x90448}$ s3dkt3m2.mtx Finite element analysis of cylindrical shells, uniform 150x100 quad mesh



5 Verification

- For matrix A, perform the Arnoldi iteration with $k = 6$. Verify that your final eigenvalues are the same as that obtained via Octave/MATLAB. Include your eigenvalue iteration chart.
- For matrix B, perform the Arnoldi iteration with $k = 42$. Include your eigenvalue iteration chart.

6 Performance

Include your eigenvalue iteration charts for each of the following tasks. (Full page plots please; some of the below plots can become quite detailed). Also, include the timings of each run and discuss any differences that you see between the runs in each task. Additionally, please compare/contrast the results of the two tasks using matrix D.

- Run your algorithm with $k = 40$, $k = 60$, and $k = 80$ for matrix C.
- Using matrix D, run your algorithm with $k = 20$ and $k = 40$.
- Using matrix D, run your algorithm with Let $k = 10$, but run the Arnoldi iteration four times (m=4), using the result from the $(m - 1)$ pass as the initial guess for the m^{th} pass. (This simulates a restarted Arnoldi situation).

7 Administrative

Suggestion: use MATLAB or GNU octave (a MATLAB clone freely available on Linux platforms, among others) to help debug your code.

Submit at least the following for your project:

- presentation of the algorithm (section 2)
- source code
- discussions and results for the Verification and Performance sections.