ECS231 Problem Set #3 Due 5pm, May 13, 2019

In this homework assignment, we develop a number of algorithms for solving large scale linear system of equations and computing few eigenvalues and eigenvectors of large scale matrices. The assignment is based the assumption of the use of MATLAB. If you do not have MATLAB access and want to use it, please let me know. Otherwise, feel free to use your favored programming language.

Part I. Iterative subspace projection methods for large scale linear systems

1. Purpose: Iterative subspace projection methods are most widely used methods for solving large sparse linear systems $Ax = b$. In this part, you implement the Steepest Descent (SD), Conjugate Gradient (CG), Minimal Residual (RM) and Restarted GMRES methods, and study the convergence behaviors of these methods for a variant of problems.

2. Write functions for each methods, called `mysd.m`, `mycg.m`, `mymr.m` and `mygmres.m`, respectively. You may use MATLAB's functions "`cg`" and "`gmres`" for testing and comparison.

   If you use C or FORTRAN, you can start with CG and GMRES in "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods"

3. Run your implementations on the following test matrices, available from "The University of Florida Sparse Matrix Collection"

   - Symmetric positive definite matrices: `bcsstk15` and `nos3`
   - Nonsymmetric matrices: `west0479` and `mahindas`
   - Additional test matrices are welcome.

4. Report your implementations and numerical findings. Instruction on the report is listed at the end of this assignment.

Part II. Iterative subspace projection methods for large scale eigenvalue problems

1. Write a MATLAB function "`myarnoldi.m`" that implements the Arnoldi method with and without reorthogonalization to compute a few eigenpairs of a large sparse matrix A.

2. MATLAB supplies a sparse demonstration matrix call `west0479`. Try the following commands in MATLAB:

```
>> load west0479;
>> A = west0479;
>> issparse(A)
>> size(A)
>> nnz(A)
>> spy(A)
```

Since this matrix is not that large, you can compute its "exact" eigenvalues using MATLAB's `eig` function: `lam = eig(full(A))`; and you can plot them use `plot(real(lam),imag(lam),'r+')`. Notice that this matrix has some outlying eigenvalues.

3. Run $j = 30$ steps of your Arnoldi method, starting with an initial vector of ones. Compute $\|AV_j - V_{j+1}\widehat{H}_j\|$ and $\|I - V_{j+1}^H V_{j+1}\|$ to check the correctness of your code. Both of these residuals should be at the machine precision.

4. The Ritz values are the eigenvalues of $j \times j$ Hessenberg matrix H_j. Compute these using the `eig` command, and plot the Ritz values and the exact eigenvalues together. You should see that the outlying eigenvalues are approximated well by the Ritz values.

5. If we had taken fewer Arnoldi steps, the Ritz values would not have been as good. Compute the Ritz values from 10-step and 20-step Arnoldi runs, and plot them together witht the exact eigenvalues.

6. Modify your Arnoldi code so that the reorthogonalization is not done. Do 30 Arnoldi steps and then check the residual $\|I - V_{j+1}^H V_{j+1}\|$. Noticed that the orthogonality has been significantly degraded. Repeat this with a run of 60 steps to see a complete lossof orthogonality. However about the approximation of Ritz values?

7. To find the eigenvalues of `west0479` that are closest to a target τ, modify your Arnoldi code so that it does shift-and-invert spectral transformation. Now, instead of multiplying a vector by A at each step, multiply the vector by $(A - \tau I)^{-1}$.

   Note that do not form the matrix $(A - \tau I)^{-1}$ explicitly. Instead perform a sparse LU factorization: `[L, R, P] = lu(A - tau*speye(n))`; and use the factors L, R and P. The factorization has to be done only once for a given run; and then the factors can be reused repeatedly. Try $\tau = 10$, $\tau = -10$ and other values. Complex values are okey as well.

8. Report your work and summarize your findings

9. If you use C or FORTRAN, you can find some implementation in "Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide".

Outline for technical report.

1. Short summary (main findings)

2. Provide a high-level description of algorthm in pseudo-code style (it's fine to copy heavily from lecture notes or other sources, just make sure to be self-content and provide proper citation)

3. Provide implementation remarks, such as stopping criterion, and accuracy assessment

4. Present numerical experiment results by the convergence behaviors for test matrices. Specifically, use plots to show relative residual norms of different methods.

5. Acknowledgment and references

6. limit to your report within 4 pages for each part (total 8 pages).