ECS 231 Final Project: Solving Large Scale Eigenvalue Problem by Arnoldi Algorithm

Mingun Cho
mgcho@ucdavis.edu (915884447)
March 18, 2024

1. Abstract

This final project involves implementing the Arnoldi algorithm with different modifications and observing the convergence behavior of sparse matrices with different sizes. These modifications include performing the Arnoldi algorithm with reorthogonalization and shift-and-invert spectral transformation. With various experiments, the results show that these modifications can help with the loss of numerical inaccuracy from the floating-point operations of the Arnoldi algorithm.

2. Arnoldi Algorithm

The Arnoldi algorithm is a numerical algorithm for finding approximated eigenvalues and eigenvectors of sparse matrices by constructing an orthonormal basis of the Krylov subspace that is defined as

$$K_m(A, v) = \text{span}\{v, Av, A^2v, ..., A^{m-1}v\}$$

where A is an $n \times n$ matrix, and v is a column vector of length n. To build the orthonormal basis $\{v_1, v_2, ..., v_m\}$ of the Krylov subspace $K_m(A, v)$, Arnoldi algorithm uses a modified Gram-Schmidt orthogonalization process. The following is a pseudo-code of the Arnoldi algorithm:

```
V(1) = V / norm(V)
for j = 1:m
     w = A * V(j)
     for i = 1:j
          H(i,j) =  V(i)' * w
          w = w - H(i,j) * V(i)
     end
     H(j+1, j) = norm(w)
     if H(j+1, j) = 0, stop
     V(j+1) = w / H(j+1, j)
end
```

The order-m Arnoldi decomposition, also known as the governing equation of Arnoldi algorithm, defines that

$$AV_m = V_{m+1}\hat{H}_m,$$
$$V_{m+1}^T V_{m+1} = I_{m+1}$$

where V is a set of vectors formed (in matrix form) from the Arnoldi algorithm and H is an upper Hessenberg matrix. We can use the Rayleigh-Ritz procedure to get approximated eigenvalue μ of H and corresponding eigenvector y:

$$H_m y = \mu y$$

The corresponding Ritz pair is $(\mu, V_m y)$ which can be used to calculate residual and determine if the approximated eigenpair will be accepted.

3. Implementation

This final project involves implementing MATLAB scripts of two versions of the Arnoldi algorithm: `myarnoldi.m`, which performs the regular Arnoldi algorithm, and `myarnoldiro.m`, which performs the Arnoldi algorithm with reorthogonalization. Even though the modified Gram-Schmidt is more stable than the original Gram-Schmidt algorithm, it still comes with a loss of numerical accuracy. [1] To compensate for the numerical accuracy, `myarnoldiro.m` performs a reorthogonalization which is a repeat of modified Gram-Schmidt to orthogonalize the current working vector w to its previous vectors and modify the Hessenberg matrix with the change.

While the regular Arnoldi algorithm involves continuously incrementing m until the convergence, I will be setting the limit of m to be 20 and 40 to limit the memory usage of the program and shorten the execution of the program. As mentioned in Section 2, the Ritz pairs will represent the approximated eigenpairs of the Arnoldi algorithm. The Ritz pairs are eigenpairs of the Hessenberg matrix H, which can be computed using MATLAB's `eig` command. One of the options for calling the `eig` command is to output two matrices: a diagonal matrix D of eigenvalues and a matrix V whose columns are the corresponding right eigenvectors. These matrices have been extracted to find the Ritz pairs of the Arnoldi algorithm.

When calculating the relative errors, the following formula has been used:

$$\frac{|\lambda_k - \mu_k^{(m)}|}{|\lambda_k|} \text{for k = 1,2,3,}$$

where λ_k and $\mu_k^{(m)}$ are the absolutely largest eigenvalues of A and H_m respectively. By using the `eig` command, we can get the "exact" eigenvalues of A. I've computed the relative errors of the first three largest eigenvalues of A and

Matrix Name	Application	Size	NNZ
tols1090	Computational Fluid Dynamics	1090x1090	3546
cryg2500	Materials	2500x2500	12349
olm5000	Computational Fluid Dynamics	5000x5000	19996
dw8192	Electromagnetics	8192x8192	41746

TABLE 1: Large test matrices from "SuiteSparse Matrix Collection"

H_m. When testing with larger matrices (sizes from 1000s to 10,000s), I've modified the formula for calculating relative error since we cannot compute the "exact" eigenvalues of A. The following is the modified formula:

$$\frac{\|A\tilde{u}_k^{(m)} - \mu_k^{(m)}\tilde{u}_k^{(m)}\|}{(\|A\| + |\mu_k^{(m)}|)\|\tilde{u}_k^{(m)}\|},$$

where $(\mu_k^{(m)}, \tilde{u}_k^{(m)})$ are Ritz pair from order-m Arnoldi decomposition.

Lastly, the project involves modifying the `myarnoldiro.m` to incorporate the shift-and-invert spectral transformation. This method involves shifting the spectrum of a matrix so that the eigenvalues are closer to a target value τ. When applying the shift-and-invert spectral transformation method to the Arnoldi algorithm, it involves multiplying a vector $V(j)$ to matrix $(A - \tau I)^{-1}$ when defining w. In MATLAB, using `inv` command to get an inverse of a matrix is considered slow and numerically inaccurate. [2] Therefore, I've used a sparse LU factorization command `lu` to obtain three matrices: L, R, P. Here, matrix L is a unit lower triangular matrix, R is a unit upper triangular matrix, and P is a permutation matrix. Now, I can find the inverse of $(A-\tau I)$ by following pseudo-code:

```
[L, R, P] = lu(A - tau*speye(n));

for j = 1:m
    Solve LRx_j = Pb_j by:
        % Forward substitution
        Ly = Pb_j
        % Backward substitution
        Rx_j = y;
end
return X = [x_1, x_2,...,x_m]
```

4. Results

This section presents numerical results by the convergence behaviors for test matrices. Section 4.1 will go over residuals, convergence of Ritz values, and relative errors of eigenvalues of Arnoldi algorithm with `west0479` matrix. Section 4.2 will be similar to its previous section but the Arnoldi algorithm will be tested with larger matrices that have been obtained from "SuiteSparse Matrix Collection". Table 1 contains information on these large matrices. Lastly, Section 4.3 will go over numerical results from performing shift-and-invert spectral transformation of the Arnoldi algorithm using all the mentioned matrices. (`west0479` and large matrices from Section 4.2)

Residuals	Value (m = 20)	Value (m = 40)
$\|AV_m - V_{m+1}\hat{H}_m\|$	9.7543e-13	2.1004e-12
$\|I - V_{m+1}^H V_{m+1}\|$	1.7728e-13	7.6112e-13

TABLE 2: Residuals of `west0479` with regular Arnoldi Algorithm

Residuals	Value (m = 20)	Value (m = 40)
$\|AV_m - V_{m+1}\hat{H}_m\|$	6.3234e-13	4.0382e-12
$\|I - V_{m+1}^H V_{m+1}\|$	5.5302e-16	1.0673e-15

TABLE 3: Residuals of `west0479` with Arnoldi Algorithm with reorthogonalization

4.1. `west0479`

Tables 2 and 3 show computed residuals $\|AV_m - V_{m+1}\hat{H}_m\|$ and $\|I - V_{m+1}^H V_{m+1}\|$ of `myarnoldi.m` and `myarnoldiro.m` with `west0479`. The values in Table 3 show the decrease of residuals when running the Arnoldi algorithm with reorthogonalization. While the second residual $\|I - V_{m+1}^H V_{m+1}\|$ is able to be around at the machine precision (1e-16), the first residual $\|AV_m - V_{m+1}\hat{H}_m\|$ still results in similar precision as the regular Arnoldi algorithm. I've tried computing the first residual $\|AV_m - V_{m+1}\hat{H}_m\|$ with scaling as the following formula:

$$\frac{\|AV_m - V_{m+1}\hat{H}_m\|}{n(\|A\| + \|H_m\|)}$$

with a constant n. When the constant is set to be a very large number ($>$ 1 billion), the residual gets closer to the machine precision.

Figures 1 and 2 show the convergence of Ritz values of `myarnoldi.m` and `myarnoldiro.m` with `west0479`. While there are more existing eigenvalues that are located exteriorly, the figures have been zoomed in to enhance their visualization. The red "+"s represent the "exact" eigenvalues of A, and the blue "o"s represent the approximated eigenvalues from the Arnoldi algorithm. For both figures, I've observed that exterior eigenvalues converge first as expected since this is the typical convergence phenomenon of the Arnoldi algorithm.

Figures 3 and 4 show the relative error of eigenvalues of `myarnoldi.m` and `myarnoldiro.m` with `west0479`. I've collected the first three (k = 3) absolutely largest eigenvalues of A and H_m where each represents red, green, and blue respectively in the graphs. These lines also have a different starting point of m, since H_m cannot contain number of eigenvalues that is greater than m. Based on the data I've collected, the relative errors for the first and second absolutely largest eigenvalues are identical so red and green lines are overlapping with each other. For some "disconnected" points in the graphs, they are due to the computed

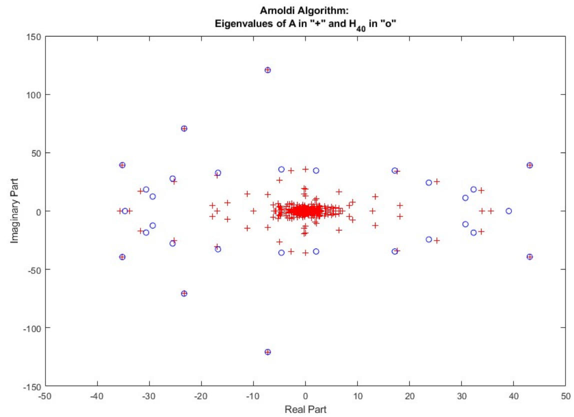

FIGURE 1: Convergence of Ritz values of `west0479` with regular Arnoldi Algorithm

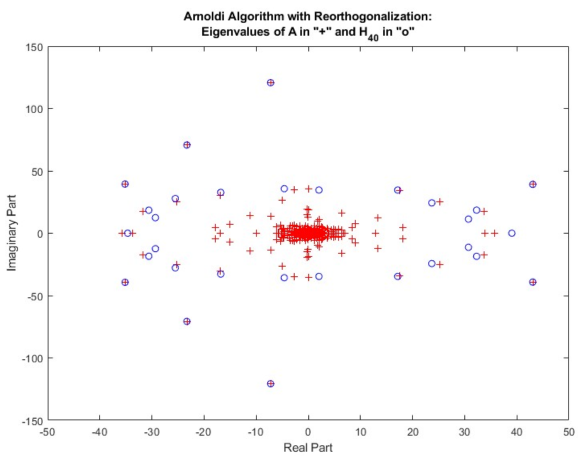

FIGURE 2: Convergence of Ritz values of `west0479` with Arnoldi Algorithm with reorthogonalization

relative errors being exactly zero. After running $m = 40$ steps, I was able to observe all three relative errors reaching the machine precision (1e-16) for both `myarnoldi.m` and `myarnoldiro.m`. While the relative errors for the third absolutely largest eigenvalues require more iterations of m to reach the machine precision, the relative errors for the first and second absolutely largest eigenvalues reach the machine precision within 20 iterations. Once they have reached the machine precision, I've observed that they continue to stay within the machine precision for the later iterations.

4.2. Large Matrices

Figures 5 and 6 show the relative error of eigenvalues of `myarnoldi.m` and `myarnoldiro.m` with large matrices obtained from the "SuiteSparse Matrix Collection". For more information on these matrices, refer to Table 1. I've tested the relative errors with $m = 100$ and collected the first absolutely largest eigenvalues of A and H_m. The red line represents `tols1090`, the green line represents

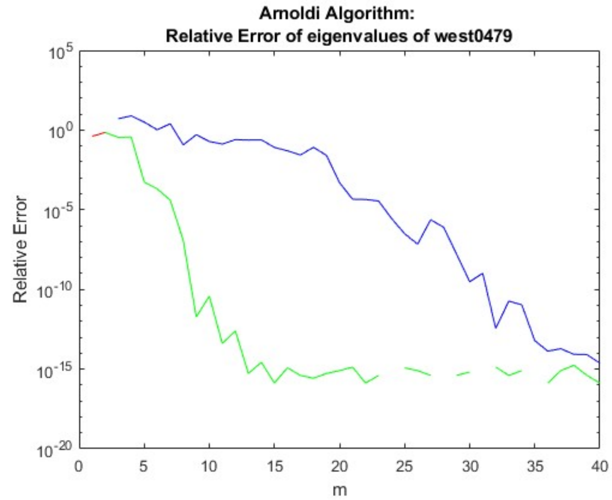

FIGURE 3: Relative Error of eigenvalues of `west0479` with regular Arnoldi Algorithm

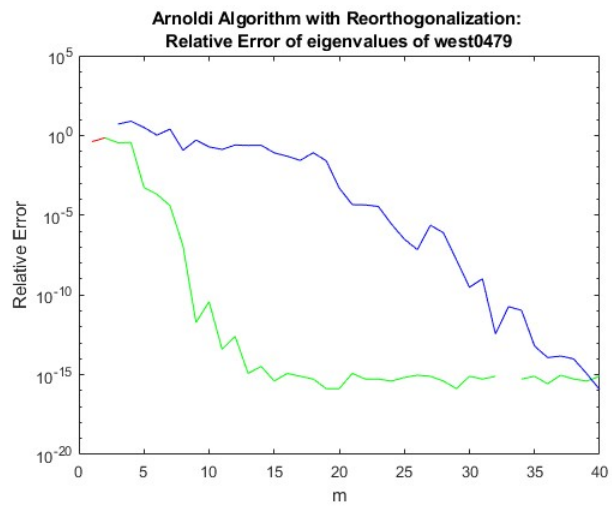

FIGURE 4: Relative Error of eigenvalues of `west0479` with Arnoldi Algorithm with reorthogonalization

`cryg2500`, the blue line represents `olm5000`, and the magenta line represents `dw8192`. Overall, the relative errors of all matrices are decreasing over the m steps. The matrix `cryg2500` converges faster than other matrices, where its relative error reaches the machine precision after $m = 30$ steps. While the relative error of other matrices did not reach the machine precision after $m = 100$ steps, they will eventually reach as m continues to be incremented.

4.3. Shift-and-Invert Spectral Transformation

Figures 7 and 8 show the relative error of eigenvalues of the Arnoldi algorithm with shift-and-invert spectral transformation with `west0479` and large matrices. For more information on these large matrices, refer to Table 1. After performing experiments with different values of τ, I've

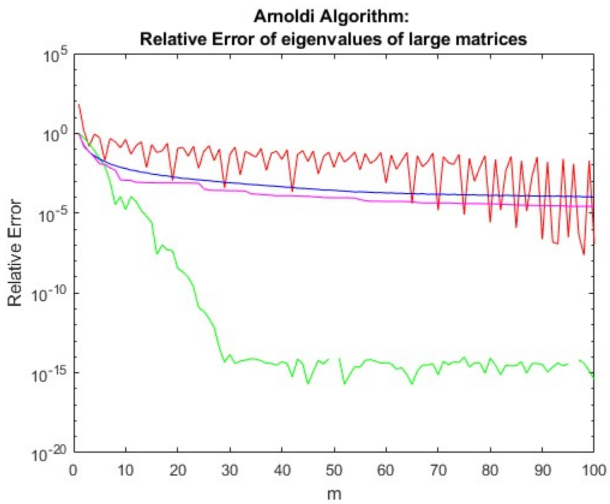

FIGURE 5: Relative Error of eigenvalues of large matrices with regular Arnoldi Algorithm

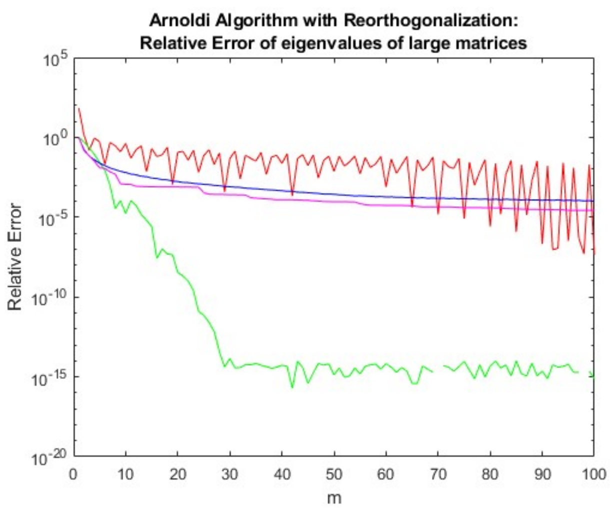

FIGURE 6: Relative Error of eigenvalues of large matrices with Arnoldi Algorithm with reorthogonalization

decided to $\tau = 5$ for `west0479`, $\tau = 7$ for `tols1090`, $\tau = -4$ for `cryg2500`, $\tau = 1$ for `olm5000` and `dw8192`. These values of τ generated results with higher accuracy with relative error decreasing as m increases. In Figure 8, the red line represents `tols1090`, the green line represents `cryg2500`, the blue line represents `olm5000`, and the magenta line represents `dw8192`. Overall, the relative errors of all matrices are decreasing over the m steps. Some matrix converges faster than others, especially the matrix `dw8192` converges with a lower number of m steps.

5. Appendix

To access MATLAB scripts that have been used for this project, go to:

https://github.com/MangoShip/ECS231Arnoldi

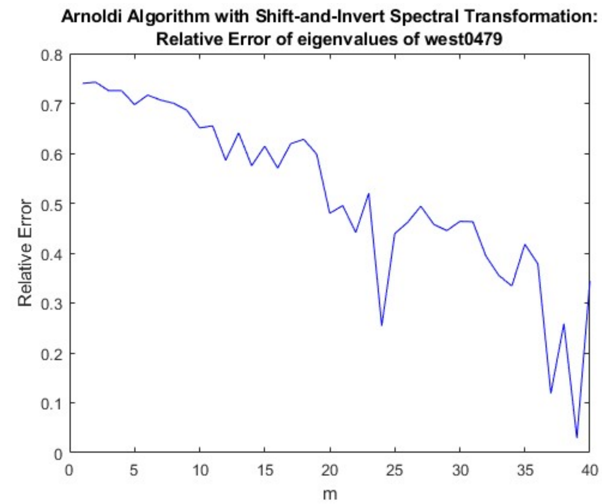

FIGURE 7: Relative Error of eigenvalues of `west0479` with regular Arnoldi Algorithm with shift-and-invert spectral transformation

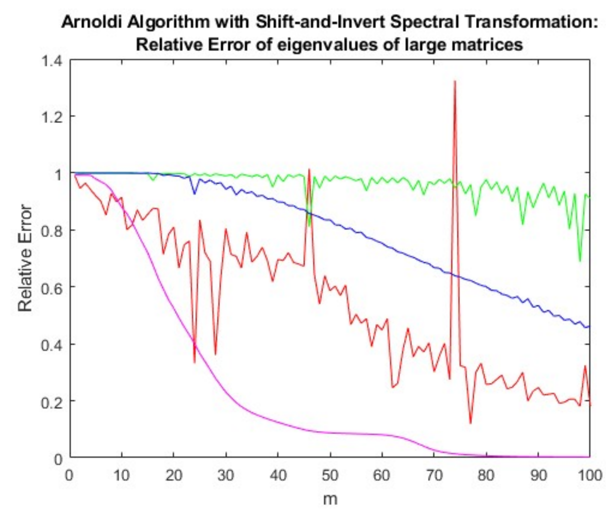

FIGURE 8: Relative Error of eigenvalues of large matrices with Arnoldi Algorithm with shift-and-invert spectral transformation

References

[1] Matrix computations (cs 6210) - lecture notes. https://www.cs.cornell.edu/~bindel/class/cs6210-f16/lec/2016-11-16.pdf/.

[2] Purpose of inv - Loren on the Art of MATLAB. https://blogs.mathworks.com/loren/2007/05/16/purpose-of-inv/.