Math 566 **Homework 6** **Due April 30, 2021**
Grady Wright

1. (**GMRES**) Pseudocode for a simplified (and inefficient) version of the Generalize Minimum Residual (GMRES) method for solving $A\mathbf{x} = \mathbf{b}$ is presented in Algorithm 1. Your goal is to implement this method in some programming language. For the least squares solution step, you can use a built-in solver (such as "\" or `mldivide` in MATLAB).

Algorithm 1 GMRES

Input: A — m-by-m matrix, $\mathbf{b}$ — m-by-1 vector, $tol > 0$ — tolerance
Output: Approximate solution $\hat{\mathbf{x}}$ to $Ax = b$ that satisfies $\|A\hat{\mathbf{x}} - \mathbf{b}\|_2/\|\mathbf{b}\| \leq tol$

$\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$
for $n = 1, 2, \ldots, m$ **do** ▷ Arnoldi iteration
  $\mathbf{v} = A\mathbf{q}_n$
  for $j = 1, 2, \ldots, n$ **do**
    $h_{jn} = \mathbf{q}_j^T\mathbf{v}$
    $\mathbf{v} = \mathbf{v} - h_{jn}\mathbf{q}_j$
  end for
  $h_{n+1,n} = \|\mathbf{v}\|$
  $\mathbf{q}_{n+1} = \mathbf{v}/h_{n+1,n}$
  Find the least squares solution to $\tilde{H}_n\mathbf{y} = \|\mathbf{b}\|\mathbf{e}_1$ ▷ Approximate solution from $\mathcal{K}_n(A, b)$
  $\mathbf{x}_n = Q_n\mathbf{y}$
  $\mathbf{r}_n = A\mathbf{x}_n - \mathbf{b}$
  if $\|\mathbf{r}_n\|/\|\mathbf{b}\| \leq tol$ **then**
    break
  end if
end for
$\hat{\mathbf{x}} = \mathbf{x}_n$

In an actual "production-level" implementation of GMRES, one would exploit the upper Hessenberg structure of the Krylov matrix $\tilde{H}_n$ to solve the least squares problem. This is typically done with Givens rotation matrices [1, §6.5].

(a) Test your code on the linear system $A\mathbf{x} = \mathbf{b}$, where A is the 200-by-200 matrix with entries

$$a_{ij} = \mathcal{N}(0, 0.25/200) + \begin{cases} 2 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}.$$

Here $\mathcal{N}(0, 0.25/200)$ is a random number from the real normal distribution with mean 2 and standard deviation $0.5/\sqrt{200}$. This matrix can be generated in MATLAB using

```
A = 2*eye(200) + 0.5*randn(200)/sqrt(200)
```

For the right hand side $\mathbf{b}$, use a vector with all ones (i.e. `b=ones(200,1)` in MATLAB), and for set $tol = 10^{-10}$. Report the number of iterations your code takes to solve the system to the desired tolerance and report the relative residual associated with the approximate solution.

(b) **Extra credit:** Suppose you use GMRES to approximate a solution of $A\mathbf{x} = \mathbf{b}$. If $\mathbf{b}$ is an eigenvector of A, what are the minimum number of iterations it will take GMRES to compute the exact solution assuming no rounding errors? Explain your answer.

2. (**Companion matrix**) In this problem you will see how to compute the roots of a polynomial with algorithms for solving eigenvalue problems.

   Consider a general n^{th} degree monic polynomial of the form

$$p(z) = z^n + c_{n-1}z^{n-1} + \cdots + c_1 z + c_0,$$

   where c_j are some real or complex numbers. One can show[1] that $p(z) = \det(zI - A)$, where I is the n-by-n identity matrix and A is the n-by-n matrix given by

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \\ -c_0 & -c_1 & \cdots & \cdots & \cdots & -c_{n-1} \end{bmatrix}.$$

   A is called the *companion matrix* of p. This relationship tells us that the roots of $p(z)$ can be computed by solving for the eigenvalues its companion matrix.

   (a) Write down the companion matrix A for $p(z) = z^2 + c_1 z + c_0$ and show that $p(z) = \det(zI - A)$ for general coefficients c_1 and c_0.

   (b) Using the eigenvalue solver `eig` in MATLAB or NumPy, compute the roots of the sixth degree Legendre polynomial

$$q(z) = \frac{1}{16}(231z^6 - 315z^4 + 105z^2 - 5).$$

   Note that you will need to make a monic polynomial p out of q that still has the same roots as q. Report the roots in ascending order to 15 decimal digits. Check that they are actual roots by plugging them into the polynomial q.

   > The companion matrix technique for computing roots is actually what MATLAB uses in its `roots` function. The companion matrix also arises in computing Gaussian Quadrature formulas for numerical integration [2]. Roots of Legendre polynomials also play a key role here.

3. (**Rayleigh quotient iteration**) The Rayleigh Quotient Iteration (RQI) algorithm discussed in class (see Algorithm 27.3 from Trefethen and Bau) is a powerful method for finding an eigenvalue–eigenvector pairs of certain matrices (especially symmetric tridiagonal ones!)

   (a) Implement a function for the RQI algorithm discussed in the lecture in your favorite programming language. Your function should take as input a matrix A, an initial starting guess $x^{(0)}$, and a tolerance ε to determine when the algorithm should stop. The output of the function should be an approximate eigenvalue–eigenvector pair of A. A MATLAB function declaration could be

```
[v,lam] = rqi(A,x0,ep)
```

   where `v` and `lam` are the corresponding eigenvalue–eigenvector pair that the algorithm converged to. Feel free to use a built in linear system solver in your code (for example, "\" or `mldivide` in MATLAB). Include your code in your write-up.

[1] The proof of this result can be expanding the determinant in the appropriate minors of A and using induction.

(b) Use your algorithm with an initial starting guess of $x^{(0)} = [1\ 1\ 1\ 1\ 1\ 1]/\sqrt{6}^T$ to compute an eigenvalue–eigenvector pair of the matrix

$$\begin{bmatrix} -2 & 2 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix}$$

Verify that your algorithm converges to an actual eigenvalue-eigenvector pair by comparing the answer you get with the answer from a builtin function such as `eig` in MATLAB or NumPy. Report the number of iterations your algorithm required to converge and the error in the approximate answer.

4. (**Gerschgorin's theorem**) Here is Gerschgorin's theorem[2]: *Every eigenvalue of an n-by-n matrix A lies in at least one of the n circular disks in the complex plane with centers a_{ii} and radii $\sum_{j\neq i}^{n} |a_{ij}|$. Moreover, if m of these disks form a connected domain that is disjoint from the $n-m$ other disks, then there are precisely m eigenvalues of A within this domain. The disks for each a_{ii} are called Gerschgorin disks.*

Gerschgorin's theorem allows a simple way to find regions in the complex plane where the eigenvalues of a matrix are located. For example, according to Gerschgorin's theorem, the matrix

$$A = \begin{bmatrix} 1 & 1 & -1 \\ \frac{1}{2} & -2 & 0 \\ 1 & 1 & 4 \end{bmatrix} \tag{1}$$

has eigenvalues in the union of the Gerschgorin disks $|\lambda - 1| < 2$, $|\lambda + 2| < \frac{1}{2}$, and $|\lambda - 4| < 2$. These disks are displayed in Figure 1. From this figure, it is obvious that the disk $|\lambda+2| < \frac{1}{2}$ is disjoint from the other 2, so Gerschgorin's theorem further tells us that one of the eigenvalues must be in that disk, with the remaining two in the union of the other two disks. The actual eigenvalues of A are $\lambda_1 = 3.558455\ldots$, $\lambda_2 = 1.615693\ldots$, and $\lambda_3 = -2.174149\ldots$, which verify the results from Gerschgorin's theorem.

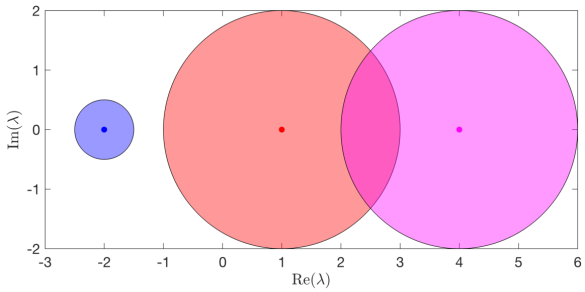

Figure 1: Gerschgorin disks for the matrix 1.

(a) Similar to the above example, use Gerschgorin's theorem to find regions in the complex plane where the eigenvalues of the following matrix are located

$$A = \begin{bmatrix} 3 & -1 & -1 & 1 \\ -1 & 2 & -\frac{1}{4} & 1 \\ -1 & -1 & -3 & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{4} & 0 & -7 \end{bmatrix}$$

Verify your results numerically.

[2] See also page 82 or 150 of ANLA or the beautiful book [3]

(b) **Extra credit.** Recall that we call a matrix A strictly (row) diagonally dominant if

$$|a_{ii}| > \sum_{j=1, i \neq j}^{n} |a_{ij}|.$$

Use Gerschgorin's theorem to show that such matrices are nonsingular.

5. (**Sylvester equations**) *Sylvester equations* are *matrix equations* that take the form

$$AX - XB = C, \tag{2}$$

where X is the unknown matrix of size m-by-n and A, B, C are given m-by-m, n-by-n and m-by-n matrices, respectively. These equations appear in many applications, from numerical methods for partial differential equations to optimal control [4].

(a) Suppose A and B are diagonalizable, i.e. they have Jordan normal forms $A = S_A^{-1}\Lambda_A S_A$ and $B = S_B^{-1}\Lambda_B S_B$, where Λ_A and Λ_B are diagonal matrices containing the eigenvalues of A and B, respectively. Show how the Jordan normal forms can be used to solve (2).

Hint: You should use the Jordan forms of A and B to transform the system (2) into one that looks like

$$\Lambda_A \widehat{X} - \widehat{X}\Lambda_B = \widehat{C}.$$

This system is particularly easy to solve for the entries of $\widehat{X}$, one at a time. After doing this, you only need to determine X from $\widehat{X}$. Be sure to state the conditions for which a solution to the system exists.

Note that this technique of solving a matrix equations is particularly effective when Λ_A and Λ_B are known explicitly and matrix-vector products involving S_A, S_A^{-1}, S_B, and S_B^{-1} can be done fast. This is often the case for solving certain separable partial differential equations with finite difference methods.

(b) Use the technique from part (a) to solve the Sylvester equation (2) where

$$S_A = \begin{bmatrix} 4 & 7 & -6 & 10 & 9 \\ 4 & -6 & 4 & 9 & 5 \\ -2 & 4 & 6 & 10 & 3 \\ -4 & 6 & 3 & -3 & 7 \\ -1 & 8 & 0 & 6 & 2 \end{bmatrix}, \qquad \Lambda_A = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix},$$

$$S_B = \begin{bmatrix} 6 & 6 & -1 & 5 \\ 8 & 7 & -6 & -6 \\ -3 & 3 & -5 & 10 \\ -6 & -6 & -9 & -7 \end{bmatrix}, \qquad \Lambda_B = \begin{bmatrix} -7 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 \\ 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & -5 \end{bmatrix},$$

$$C = \begin{bmatrix} -9 & 10 & 6 & -7 \\ -8 & -2 & -5 & 3 \\ -7 & 0 & -6 & 5 \\ -8 & 9 & 0 & 8 \\ -4 & -9 & -4 & 5 \end{bmatrix}$$

Report you answer in matrix form to at least 5 significant digits and turn in a listing of your code.

(c) **Extra credit.** Suppose that instead of the Jordan form of A and B, one has the Schur form, i.e. $A = Q_A^* T_A Q_A$ and $B = Q_B^* T_B Q_B$, where Q_A & Q_B are unitary and T_A & T_B are upper triangular. Explain how to solve the Sylvester equation (2).

Hint: The same approach as part (a) can be applied, but in the end you have to solve a system for the transformed $\widehat{X}$ of the form

$$T_A \widehat{X} - \widehat{X} T_B = \widehat{C}.$$

You need to explain how to solve this system with a process analogous to back substitution.

References

[1] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2003.

[2] Gene H. Golub and John H. Welsch. Calculation of Gauss quadrature rules. *Math. Comp.*, 23:221–230, 1969.

[3] Richard S. Varga. *Geršgorin and His Circles*. Springer-Verlag, Berlin, 2004.

[4] V. Simoncini. Computational methods for linear matrix equations. *SIAM Review*, 58(3):377–441, 2016.