Numerical Matrix Analysis

Notes #20 — Eigenvalues
The QR-Algorithm

Peter Blomgren
⟨blomgren@sdsu.edu⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Spring 2024
(Revised: April 9, 2024)

Outline

Recap: Last Time

We noted that eigenvalue revealing computations are generally divided into 2 phases; in **phase 1** we transform the matrix in into **Hessenberg form** in a finite number of steps, and in **phase 2** we apply a (possibly infinite) number of transformations to transform the Hessenberg matrix into **upper triangular form**

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \xrightarrow{\text{Phase 1}} \begin{bmatrix} * & * & + & + & + \\ * & * & * & + & + \\ & * & * & * & + \\ & & * & * & * \\ & & & * & * \end{bmatrix} \xrightarrow{\text{Phase 2}} \begin{bmatrix} * & + & + & + & + \\ & * & + & + & + \\ & & * & + & + \\ & & & * & + \\ & & & & * \end{bmatrix}$$

Where the "+"-entries are zeros if A is Hermitian ($A = A^*$).

Recap: Last Time

Phase 1 Can be backwardly stably computed using a slightly modified version of the Householder-QR algorithm (making all the reflectors one element shorter, and applying Q_k from the left* **and** the right.)

Phase 2 Instead of directly talking about phase 2, we looked at...

- **Rayleigh quotient** (eigenvalue estimation).
- **Power iteration** (only useful as a basis for...)
- **Inverse iteration** (eigenvector estimation).
- **Rayleigh quotient iteration** (eigenvalue and eigenvector estimation — **cubically convergent**).

Next, we look at the QR-algorithm, and make some connections with the ideas above.

The QR-Algorithm

The QR-algorithm can be viewed as a stable procedure for computing QR-factorizations of the matrix powers A, A^2, A^3, ...

Algorithm (The "Pure" QR-Algorithm)

$\mathbf{A_{(0)}} = \mathbf{A}$
```
k = 1
while( ... )
```
$\quad\left[\mathbf{Q_{(k)}}, \mathbf{R_{(k)}}\right] \leftarrow \mathtt{qr}\left(\mathbf{A_{(k-1)}}\right)$

$\quad\mathbf{A_{(k)}} \leftarrow \mathbf{R_{(k)}}\mathbf{Q_{(k)}}$
```
    k ← k+1
endwhile
```

Under suitable (non-restrictive) assumptions, this simple algorithm converges to a Schur form for the matrix A — Upper triangular if A is arbitrary, and diagonal if A is Hermitian.

Before we go any further, let us illustrate this numerically...

The QR-Algorithm: Applied to a Non-Hermitian A ∃ Movie

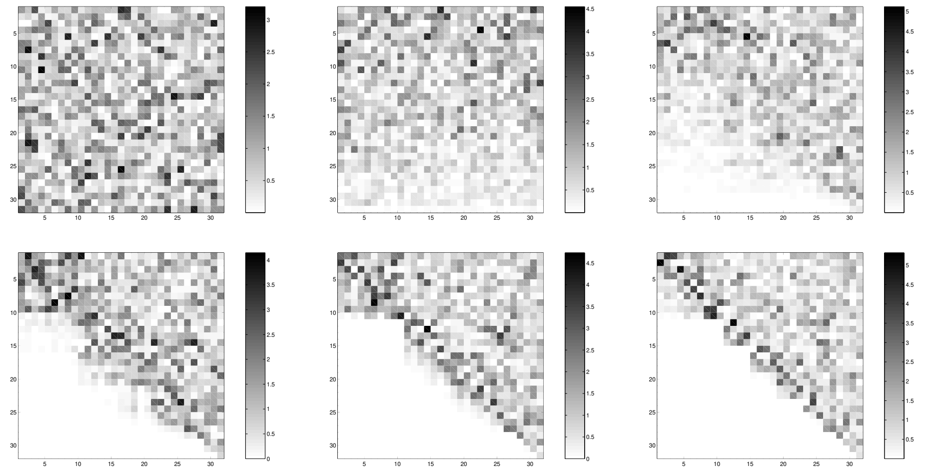


Figure: The QR-algorithm applied to a non-Hermitian matrix A. The panels show the initial matrix, and iterations 1, 4, 16, 32, and 64.

The QR-Algorithm: Applied to $A = A^*$ ∃ Movie

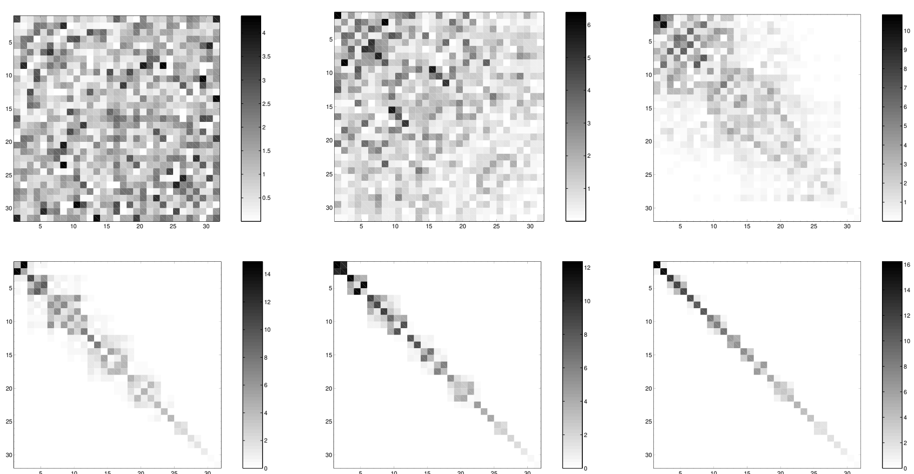


Figure: The QR-algorithm applied to a Hermitian matrix A. The panels show the initial matrix, and iterations 1, 4, 16, 32, and 64.

The QR-Algorithm: What Are We Doing???

Just a quick sanity-check... If $QR = A$, then

$$Q^*AQ = Q^*(QR)Q = (Q^*Q)RQ = RQ$$

Hence the matrices that we form **are** unitarily similar

$$A_{(k)} = Q^*_{(k)} \cdots Q^*_{(1)}\, A_{(0)}\, \underbrace{Q_{(1)} \cdots Q_{(k)}}_{\underline{Q}_{(k)}}$$

In fact, this is the idea we had to reject in our effort to compute the Hessenberg form of A. Even though it has to be rejected as a **finite-step** method for transforming A, it turns out to be quite powerful as the basis of an iterative scheme.

As in the last lecture, in order to keep the discussion simple(r), we assume that $A \in \mathbb{R}^{m\times m}$, and $A = A^*$ so that $\lambda_k(A) \in \mathbb{R}$, and the set of eigenvectors is orthonormal.

The QR-Algorithm: Modifications

Since we will be applying the QR-algorithm to real symmetric matrices, we are looking for the diagonalization $\Lambda(A)$.

Like the Rayleigh quotient algorithm, the QR-algorithm (for real symmetric matrices) **can be made to converge cubically**. In order to achieve this, we must introduce three modifications:

1. Before entering the iteration, A must be reduced to tri-diagonal form (using the "Hessenberg algorithm" (Phase#1)).
2. Instead of $A_{(k)}$, the **shifted matrix** $(A_{(k)} - \mu_{(k)} I)$ is factored at each step, where $\mu_{(k)}$ is an eigenvalue estimate.
3. Whenever possible, and in particular whenever an eigenvalue is found, the problem is **"deflated"** by breaking $A_{(k)}$ into sub-matrices.

The Modified QR-Algorithm

Algorithm (Modified QR-Algorithm)

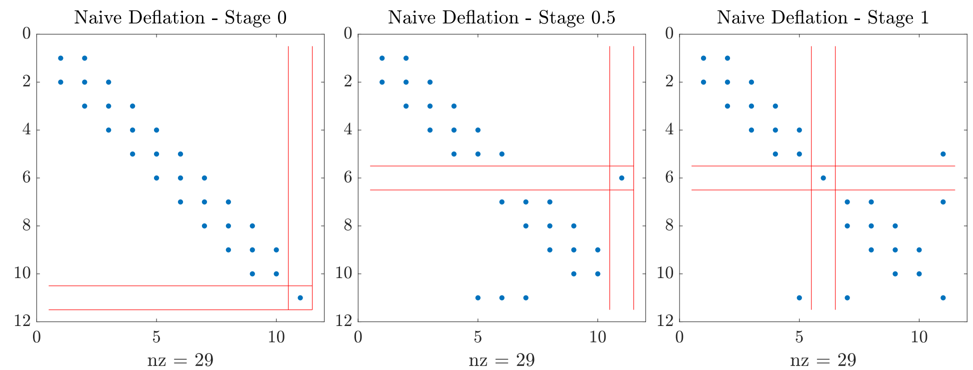
$\mathbf{A}_{(0)} \leftarrow$ `hessenberg_form`$(\mathbf{A})$
$\delta \leftarrow$ `small tolerance` $\sim \sqrt{\varepsilon_{\mathtt{mach}}}$?
`k` $\leftarrow$ `0`
`while( ... )`
  `Select` $\mu_{(\mathbf{k})}$
  $\left[\mathbf{Q}_{(\mathbf{k})}, \mathbf{R}_{(\mathbf{k})}\right] \leftarrow \mathtt{qr}\left(\mathbf{A}_{(\mathbf{k-1})} - \mu_{(\mathbf{k})}\mathbf{I}\right)$
  $\mathbf{A}_{(\mathbf{k})} \leftarrow \mathbf{R}_{(\mathbf{k})}\mathbf{Q}_{(\mathbf{k})} + \mu_{(\mathbf{k})}\mathbf{I}$
  `If any 1st super-diagonal` $|\mathbf{A}_{(\mathbf{k}),\mathbf{j},\mathbf{j+1}}| \le \delta$ `then`
   `Set` $\mathbf{A}_{(\mathbf{k}),\mathbf{j},\mathbf{j+1}} = \mathbf{A}_{(\mathbf{k}),\mathbf{j+1},\mathbf{j}} = \mathbf{0}$, `so that`
   $$\left[\begin{array}{c|c} A^{[1]}_{(k)} & 0 \\ \hline 0 & A^{[2]}_{(k)} \end{array}\right] = \mathbf{A}_{(\mathbf{k})}$$
   `recursively apply QR-algorithm to` $A^{[1]}_{(k)}$ `and` $A^{[2]}_{(k)}$
  `endif`
  `k` $\leftarrow$ `k + 1`
`endwhile`

The Modified QR-Algorithm: Components

1. We have already discussed reduction to Hessenberg form.
2. We will return to a discussion on selecting the shifts $\mu_{(k)}$.
3. We leave the discussion on deflation as *"an exercise for the motivated student."* (There are many details, *e.g.* pivoting to split the matrix exactly in "half," to be taken care of to make this step maximally efficient)

For now, we focus the discussion on the "pure" form of the QR-algorithm... We relate the QR-algorithm to another method — **simultaneous iteration** — whose behavior is more intuitive.

The Modified QR-Algorithm: Naive Deflation Fails



We will see that the bottom right entry is where we get fastest (cubic) convergence. A naive pivoting strategy which tries to split the matrix into two blocks of equal size fails in that the upper-right and lower-left blocks are not empty. "Some" more work is required.

Unnormalized Simultaneous Iteration

The Idea: Apply the power iteration to several vectors at once.

Suppose we have a set of **linearly independent** vectors $\{v_1^{(0)}, \ldots, v_n^{(0)}\}$, then the spaces spanned by the vectors $\{A^k \vec{v}_1^{(0)}, \ldots, A^k \vec{v}_n^{(0)}\}$ generated by simultaneous power iteration, converge to the space spanned by the n eigenvectors $\vec{q}_k$ corresponding to the n abs-largest eigenvalues $|\lambda_k| > 0$, *i.e.*

$$\lim_{k\to\infty} \mathrm{span}\left(A^k \vec{v}_1^{(0)}, \ldots, A^k \vec{v}_n^{(0)}\right) = \mathrm{span}\left(\vec{q}_1, \ldots, \vec{q}_n\right).$$

In matrix form

$$V_{(0)} = \left[\begin{array}{c|c|c} \vec{v}_1^{(0)} & \cdots & \vec{v}_n^{(0)} \end{array}\right], \quad V_{(k)} = A^k V_{(0)} = \left[\begin{array}{c|c|c} \vec{v}_1^{(k)} & \cdots & \vec{v}_n^{(k)} \end{array}\right]$$

Unnormalized Simultaneous Iteration

Since we are interested in $\mathrm{span}\left(A^k \vec{v}_1^{(0)}, \ldots, A^k \vec{v}_n^{(0)}\right)$, *i.e.* the column-space / span / image of $V_{(k)}$ we compute the reduced QR-factorization

$$\widehat{Q}_{(k)} \widehat{R}_{(k)} = V_{(k)}.$$

We can justify that the columns of $\widehat{Q}_{(k)}$ converge to the eigenvectors $\vec{q}_k$; if we write both $\vec{v}_\ell^{(0)}$ and $\vec{v}_\ell^{(k)}$ in term of the eigenvectors of A

$$\begin{aligned} \vec{v}_\ell^{(0)} &= a_{1\ell}\vec{q}_1 + \cdots + a_{m\ell}\vec{q}_m \\ \vec{v}_\ell^{(k)} &= \lambda_1^k a_{1\ell}\vec{q}_1 + \cdots + \lambda_m^k a_{m\ell}\vec{q}_m. \end{aligned}$$

For simplicity *(we can discuss eigenvectors rather than invariant eigenspaces)* we assume that the first n eigenvalues are distinct, and ordered so that

Assumption #1

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > |\lambda_{n+1}| \geq |\lambda_{n+2}| \geq \cdots \geq |\lambda_m|$$

Unnormalized Simultaneous Iteration

We need one further assumption before we can state a theorem — Let $\widehat{Q}$ be the $(m \times n)$ matrix whose columns are the eigenvectors $\vec{q}_k$. We need the following to be true

Assumption #2

All the leading principal sub-matrices of $\widehat{Q}^* V_{(0)}$ are non-singular.

A leading principal sub-matrix is anchored in the upper left corner (the m_{11}-element) and is a square matrix of size (1×1), (2×2), $\ldots$, $(n \times n)$.

With these assumptions we can say something about how the vectors generated by the simultaneous iteration converge to the eigenvectors.

Unnormalized Simultaneous Iteration

Theorem

Suppose the iteration defined by

$$V_{(0)} = \left[\begin{array}{c|c|c} \vec{v}_1^{(0)} & \cdots & \vec{v}_n^{(0)} \end{array}\right], \quad V_{(k)} = A^k V_{(0)}, \quad \widehat{Q}_{(k)} \widehat{R}_{(k)} = V_{(k)}$$

is carried out, and that the assumptions (see slide 14 and 15) are satisfied. Then, as $k \to \infty$, the columns of the matrices $\widehat{Q}_{(k)}$ converge linearly to the eigenvectors of A

$$\|\vec{q}_j^{(k)} \mp \vec{q}_j\| = \mathcal{O}(c^k)$$

for each $j \in [1, n]$, where $c < 1$ is the constant

$$c = \max_{1\leq k<n} \left|\frac{\lambda_{k+1}}{\lambda_k}\right| < 1.$$

Simultaneous Iteration

We have a problem: As $k \to \infty$, all the vectors $\vec{v}_1^{(k)}, \ldots, \vec{v}_n^{(k)}$ in the un-normalized simultaneous iteration converge to the **same** dominant eigenvector $\vec{q}_1(A)$.

Even though $\mathrm{span}\left(\vec{v}_1^{(k)}, \ldots, \vec{v}_n^{(k)}\right)$ converges to something useful, *i.e.* $\mathrm{span}\left(\vec{q}_1, \ldots, \vec{q}_n\right)$, these vectors constitute a **highly ill-conditioned basis** (nearly linearly dependent basis) for that space. For practical purposes this approach is useless.

The fix is straight-forward:

Necessary Improvement

We must **orthonormalize** the basis in **every iteration**. Instead of forming the sequence $V_{(k)}$, we form a sequence $Z_{(k)}$ with the same column spaces / spans / images, but where $Z_{(k)}$ is orthonormal.

Simultaneous Iteration

Algorithm (Simultaneous Iteration)

```
Let Q̂(0) ∈ ℝ^(m×n) with orthonormal columns
k = 0
while( ... )
    Z(k) ← A Q̂(k−1)
    [Q(k), R(k)] ← qr(Z(k))
    k ← k + 1
endwhile
```

Clearly, the column spaces / spans / images of $Q_{(k)}$ and $Z_{(k)}$ are the same.

Also, as long as the initial matrices $(\widehat{Q}_{(0)})$ are the same, this algorithm generates the same sequence $Q_{(k)}$ as the unnormalized simultaneous iteration.

For the price of a QR-factorization per iteration we get a much better conditioned sequence of basis for the space; $\mathrm{span}\left(\vec{z}_1^{(k)}, \ldots, \vec{z}_n^{(k)}\right) \to \mathrm{span}\left(\vec{q}_1, \ldots, \vec{q}_n\right)$.

Simultaneous Iteration ⇔ QR-Algorithm

The QR-algorithm is **equivalent** to simultaneous iteration applied to the full set $(n = m)$ of initial vectors, *i.e.* $Q_{(0)} = I_{m \times m}$.

We are now dealing with the full QR-factorizations, so we drop the hats on $Q_{(k)}$, and $R_{(k)}$. Further, let $\underline{Q}_{(k)}$ denote the matrices generated by the simultaneous iteration, and $Q_{(k)}$ be the matrices generated by the QR-algorithm...

Simultaneous Iteration	**Pure QR-Algorithm**
$\underline{Q}_{(0)} = I$ $Z_{(k)} = A\underline{Q}_{(k-1)}$ $\underline{Q}_{(k)}R_{(k)} = Z_{(k)}$	$A^{(0)} = A$ $Q_{(k)}R_{(k)} = A^{(k-1)}$ $A^{(k)} = R_{(k)}Q_{(k)}$
$A^{(k)} = (\underline{Q}_{(k)})^* A \underline{Q}_{(k)}$	$\underline{Q}_{(k)} = Q_{(1)}Q_{(2)} \cdots Q_{(k)}$
$\underline{R}_{(k)} = R_{(k)}R_{(k-1)} \cdots R_{(1)}$	

Table: The operations and quantities that define the Simultaneous Iteration algorithm and Pure QR-Algorithm.

Simultaneous Iteration ⇔ QR-Algorithm

Theorem

The Simultaneous Iteration algorithm and the Pure QR-algorithm generate identical sequences of matrices $\underline{R}_{(k)}$, $\underline{Q}_{(k)}$, and $A^{(k)}$, namely those defined by the QR-factorization of A^k,

$$\underline{Q}_{(k)}\underline{R}_{(k)} = A^k,$$

together with the projection (similarity relation)

$$A^{(k)} = (\underline{Q}_{(k)})^* A \underline{Q}_{(k)}.$$

This is not obvious at first glance, so let's look at the proof...

Equivalence of Simultaneous Iteration and QR-Algorithm

Proof: [By Induction] The base case ($k = 0$) is trivial, for both `SI` and `QR-Alg` we immediately see that

$$A^0 = \underline{Q}_{(0)} = \underline{R}_{(0)} = I, \quad A^{(0)} = A,$$

from which

$$A^0 = \underline{Q}_{(0)}\underline{R}_{(0)}, \quad A^{(0)} = (\underline{Q}_{(0)})^* A \underline{Q}_{(0)}. \qquad \checkmark$$

Now, consider $k \geq 1$ for `SI`: The second part of the theorem is valid by definition — $A^{(k)} = (\underline{Q}_{(k)})^* A \underline{Q}_{(k)}$. The first part follows from

$$A^k \overset{[1]}{=} A\underline{Q}_{(k-1)}\underline{R}_{(k-1)} \overset{[2]}{=} \underline{Q}_{(k)}R_{(k)}\underline{R}_{(k-1)} \overset{[3]}{=} \underline{Q}_{(k)}\underline{R}_{(k)}$$

[1] Follows from the inductive hypothesis $A^{k-1} = \underline{Q}_{(k-1)}\underline{R}_{(k-1)}$

[2] From $\underline{Q}_{(k)}R_{(k)} = Z^{(k)} = A\underline{Q}_{(k-1)}$

[3] From $\underline{R}_{(k)} = R_{(k)}R_{(k-1)} \cdots R_{(1)}$

First Section
The QR-Algorithm
Back from the Detour
Simultaneous Iteration ⇔ QR-Algorithm
Equivalence Proof
Putting it Together: Convergence

Equivalence of Simultaneous Iteration and QR-Algorithm

Next we consider $k \geq 1$ for `QR-Alg`: We verify the first part of the theorem by the sequence

$$A^k \overset{[1]}{=} A\underline{Q}_{(k-1)}\underline{R}_{(k-1)} \overset{[2]}{=} \underline{Q}_{(k-1)}A^{(k-1)}\underline{R}_{(k-1)} \overset{[3]}{=} \underline{Q}_{(k)}\underline{R}_{(k)}$$

[1] Follows from the inductive hypothesis $A^{k-1} = \underline{Q}_{(k-1)}\underline{R}_{(k-1)}$

[2] From the inductive hypothesis $A^{(k-1)} = (\underline{Q}_{(k-1)})^* A \underline{Q}_{(k-1)}$, multiplied from the left by $\underline{Q}_{(k-1)}$.

[3] From $Q_{(k)}R_{(k)} = A^{(k-1)}$, $\underline{Q}_{(k)} = Q_{(1)}Q_{(2)} \cdots Q_{(k)}$, and $\underline{R}_{(k)} = R_{(k)}R_{(k-1)} \cdots R_{(1)}$

First Section
The QR-Algorithm
Back from the Detour
Simultaneous Iteration ⇔ QR-Algorithm
Equivalence Proof
Putting it Together: Convergence

Equivalence of Simultaneous Iteration and QR-Algorithm

Finally, we verify the second part by the sequence

$$A^{(k)} \overset{[1]}{=} (Q_{(k)})^* A^{(k-1)} Q_{(k)} \overset{[2]}{=} (\underline{Q}_{(k)})^* A \underline{Q}_{(k)}$$

[1] Follows from $Q_{(k)}R_{(k)} = A^{(k-1)}$, and $A^{(k)} = R_{(k)}Q_{(k)}$

[2] From the inductive hypothesis $A^{(k-1)} = (\underline{Q}_{(k-1)})^* A \underline{Q}_{(k-1)}$, and $\underline{Q}_{(k)} = Q_{(1)}Q_{(2)} \cdots Q_{(k)}$ $\square$

First Section
The QR-Algorithm
Back from the Detour
Simultaneous Iteration ⇔ QR-Algorithm
Equivalence Proof

Convergence of the QR-Algorithm

Let's put together the pieces of the **"QR-Algorithm Jigsaw Puzzle"**

- The relations (from the theorem)

[i] $\underline{Q}_{(k)}\underline{R}_{(k)} = A^k$, tell us why we expect to find the eigenvectors — the QR-Algorithm constructs orthonormal bases for successive powers of A^k

[ii] $A^{(k)} = (\underline{Q}_{(k)})^* A \underline{Q}_{(k)}$ explain why we find the eigenvalues — the diagonal elements of $A^{(k)}$ are the Rayleigh coefficients of A corresponding to the columns of $\underline{Q}_{(k)}$. As the columns converge to eigenvectors, the Rayleigh coefficients converge (*"quadratically faster"*) to the corresponding eigenvalues. Since $\underline{Q}_{(k)}$ converge to an orthonormal matrix, the off-diagonal elements in $A^{(k)}$ must converge to zero.

Convergence of the QR-Algorithm

Theorem

Let the pure QR-Algorithm be applied to a real symmetric matrix A whose eigenvalues satisfy $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_m|$ *and whose corresponding eigenvector matrix Q has all non-singular leading principal sub-matrices. Then as* $k \to \infty$, $A^{(k)}$ *converges linearly with constant* $\max_{1 \le j < n} \left| \frac{\lambda_{j+1}}{\lambda_j} \right|$ *to* $\mathrm{diag}(\lambda_1, \ldots, \lambda_m)$ *and* $\underline{Q}_{(k)}$ *converges at the same rate to* Q $(\mathrm{mod}\ \pm 1 \cdot \underline{\vec{q}}_j^{(k)})$.

Next, we look into adding shifts to the QR-Algorithm in order to speed up the convergence.

Building for the Final

Final-Fragments:

- **Inverse Iteration:** You are going to need an implementation of the inverse iteration [LECTURE#19]. You are free to use a library / built-in call to solve the linear system in the inverse iteration.
- **Rayleigh Quotient:** You are going to need an implementaton of the Rayleigh quotient [LECTURE#19] (not to be confused with the Rayleigh quotient iteration).

Now is a good time to start building and testing...

Homework AI-Policy Spring 2024

AI-era Policies — SPRING 2024

AI-3 Documented: *Students can use AI in any manner for this assessment or deliverable, but they must provide appropriate documentation for all AI use.*

This applies to ALL MATH-543 WORK during the SPRING 2024 semester.

The goal is to leverage existing tools and resources to generate HIGH QUALITY SOLUTIONS to all assessments.

You MUST document what tools you use and HOW they were used (including prompts); AND how results were VALIDATED.

BE PREPARED to DISCUSS homework solutions and AI-strategies. **Participation in the in-class discussions will be an essential component of the grade for each assessment.**