

Numerical Matrix Analysis

Notes #19 — Eigenvalues Hessenberg Form, Rayleigh Quotient

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: March 28, 2024)

Outline

- 1 Eigenvalue Problems
 - Schur Factorization
 - Phase#1 – Upper Hessenberg Form
- 2 Detour — Classical Eigenvalue Algorithms
 - The Rayleigh Quotient
 - Power Iteration
 - Inverse Iteration
- 3 Rayleigh Quotient Iteration
 - Algorithm
 - Convergence
 - Work

Last Time: Introduction to Eigenvalue Problems

Three factorizations which expose the eigenvalues of a matrix.

Type	Form	Restrictions on A	Vectors	
Unitary Diagonalization	$A = Q\Lambda Q^*$	Normal, $A^*A = AA^*$	✓	↓ less restrictions ↓
Diagonalization	$A = X\Lambda X^{-1}$	Non-defective	✓	
Schur Triangularization	$A = QTQ^*$	None	—	

Eigenvalue problems are fundamentally more difficult than solution of linear systems and/or least squares problems. We cannot guarantee, **even in exact arithmetic**, a solution in a finite number of steps.

Therefore —

Fact

Any eigenvalue solver must be iterative.

Schur Factorization and Diagonalization

1 of 2

Modern general-purpose eigenvalue algorithms tend to be based on the Schur factorization. We get $A = QTQ^*$ by finding a sequence of unitary similarity transformations

$$\underbrace{Q_k^* \cdots Q_3^* Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 Q_3 \cdots Q_k}_Q = T, \quad k \rightarrow \infty,$$

where T is upper triangular.

If $A \in \mathbb{R}^{m \times m}$, but not symmetric ($A^T \neq A$), then T may have **complex eigenvalues**. — We either must implement complex arithmetic, or we can allow T to have (2×2) -blocks along the diagonal.

$$\begin{bmatrix} \ddots & & & & & \\ & \lambda_r & & & & \\ & & -\lambda_i & & & \\ & & \lambda_i & & & \\ & & & & \ddots & \\ & & & & & \lambda_r \end{bmatrix}, \quad \lambda = \lambda_r \pm \sqrt{-1} \lambda_i$$

Schur Factorization and Diagonalization

2 of 2

Allowing (2×2) -blocks along the diagonal saves the overhead of complex arithmetic, and is known as the **real Schur factorization**.

Special Case

When A is Hermitian ($A = A^*$), then

$$Q_k^* \cdots Q_3^* Q_2^* Q_1^* A Q_1 Q_2 Q_3 \cdots Q_k = T, \quad k \rightarrow \infty$$

is also Hermitian, *i.e.* $T = T^*$, and upper triangular \rightsquigarrow **T is diagonal**.

The eigenvalue computation is usually split into **2 phases** — (PHASE#1) completes in a finite number of steps and transforms the matrix into **upper Hessenberg** form; (PHASE#2) is iterative and converges ($k \rightarrow \infty$) to upper triangular form.

Two-Phase Eigenvalue Computation

1 of 2

When $A \neq A^*$: $A \mapsto H_A \mapsto T_A$

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \xrightarrow{\text{Phase\#1}} \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{bmatrix}}_{\text{Hessenberg Form}} \xrightarrow{\text{Phase\#2}} \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \\ & & & & * \end{bmatrix}$$

When $A = A^*$: $A \mapsto T_A \mapsto D_A$

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

In this case, the Hessenberg Form is Tri-Diagonal.

Two-Phase Eigenvalue Computation

2 of 2

Phase#1 Requires $\mathcal{O}(m^3)$ operations.

Phase#2 May (in theory) require infinitely many iterations, each of which requires $\mathcal{O}(m^2)$ operations. In practice, convergence to $\mathcal{O}(\varepsilon_{\text{mach}})$ can usually be achieved in $\mathcal{O}(m)$ iterations, *i.e.* the total work requirement is $\mathcal{O}(m^3)$.

When A is Hermitian, **Phase#2** can be executed with only $\mathcal{O}(m)$ operations/iteration; thus the total work estimate for the second phase is only $\mathcal{O}(m^2)$ in this case. Hence, the “*infinite*” part of the algorithm is an order of magnitude faster than the “*finite*” part.

Why Hessenberg Form?

1 of 2

We are looking to compute the Schur factorization $A = QTQ^*$.

Why not go straight for T ???

Ponder... the first standard Householder reflector Q_1^*

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \xrightarrow{Q_1^* A} \begin{bmatrix} * & * & * & * & * \\ \mathbf{0} & * & * & * & * \\ \mathbf{0} & * & * & * & * \\ \mathbf{0} & * & * & * & * \\ \mathbf{0} & * & * & * & * \end{bmatrix} \xrightarrow{Q_1^* A Q_1} \begin{bmatrix} * & * & * & * & * \\ \otimes & * & * & * & * \\ \otimes & * & * & * & * \\ \otimes & * & * & * & * \\ \otimes & * & * & * & * \end{bmatrix}$$

Whoops!!! The multiplication from the right will fill in \otimes the first column again... The sub-diagonal elements are typically reduced in magnitude, but at this point this does not get us closer to the goal...

Why Hessenberg Form?

2 of 2

Let's instead use a Householder reflector Q_1^* which ignores the first row (the \otimes s are completely untouched), and introduces zeros as shown below

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} \xrightarrow{Q_1^* A} \begin{bmatrix} \otimes & \otimes & \otimes & \otimes & \otimes \\ + & + & + & + & + \\ \mathbf{0} & + & + & + & + \\ \mathbf{0} & + & + & + & + \\ \mathbf{0} & + & + & + & + \end{bmatrix} \xrightarrow{Q_1^* A Q_1} \begin{bmatrix} \otimes & + & + & + & + \\ \otimes & + & + & + & + \\ 0 & + & + & + & + \\ 0 & + & + & + & + \\ 0 & + & + & + & + \end{bmatrix}$$

When we multiply by Q_1 from the right, the first column is completely untouched, and the other columns are replaced by linear combinations of the columns in $Q_1^* A$.

Note that when $A = A^*$, the Q -multiplication from the right leads to the analogous row combinations, so that the top row turns into $[\otimes + \mathbf{0} \mathbf{0} \mathbf{0}]$

We now repeat the same strategy...

To Hessenberg Form

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix} \xrightarrow{Q_2^* []} \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & + & + & + & + \\ 0 & \mathbf{0} & + & + & + \\ 0 & \mathbf{0} & + & + & + \end{bmatrix} \xrightarrow{Q_2^* [] Q_2} \begin{bmatrix} * & * & + & + & + \\ * & * & + & + & + \\ 0 & * & + & + & + \\ 0 & 0 & + & + & + \\ 0 & 0 & + & + & + \end{bmatrix}$$

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix} \xrightarrow{Q_3^* []} \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & + & + & + \\ 0 & 0 & \mathbf{0} & + & + \end{bmatrix} \xrightarrow{Q_3^* [] Q_3} \begin{bmatrix} * & * & * & + & + \\ * & * & * & + & + \\ 0 & * & * & + & + \\ 0 & 0 & * & + & + \\ 0 & 0 & 0 & + & + \end{bmatrix}$$

Elements marked with * are not changed/touched and elements marked with + are changed/touched. In the practical code, we can skip “touching” the known zeros to save some work.

Householder Reduction to Hessenberg Form

Algorithm (Householder Reduction to Hessenberg Form)

Transform $A \in \mathbb{R}^{m \times m}$ to Hessenberg Formfor $k = 1:(m-2)$

$$\vec{x} = A((k+1):m, k)$$

$$\vec{v}_k = \text{sign}(x_1) \|\vec{x}\| \vec{e}_1 + \vec{x}$$

$$\vec{v}_k = \vec{v}_k / \|\vec{v}_k\|_2$$

$$A((k+1):m, k:m) = A((k+1):m, k:m) - 2\vec{v}_k(\vec{v}_k^* A((k+1):m, k:m)) \quad [*]$$

$$A(1:m, (k+1):m) = A(1:m, (k+1):m) - 2(A(1:m, (k+1):m) \vec{v}_k) \vec{v}_k^*$$

endfor

[*] Only operates on the non-zero columns.

Just as when we compute the QR-factorization using Householder reflections, the matrix Q is never formed explicitly. If we save the vectors \vec{v}_k , then we can reconstruct Q , or the action of Q as needed.

The work needed for Hessenberg reduction is $\sim \left(\frac{10}{3} m^3\right)$ operations.

Backward Stability of Hessenberg Reduction

1 of 2

Since Hessenberg reduction contains operations of the forms

- “Householder reflection from the left,” and
- “Householder reflection from the right,”

it should not come as a big surprise that the stability result looks very much like the one for QR-factorization (which is built on “Householder reflection from the left”-operations).

Backward Stability of Hessenberg Reduction

2 of 2

Theorem (Backward Stability of Hessenberg Reduction)

Let the Hessenberg reduction $A = QHQ^$ of a matrix $A \in \mathbb{C}^{m \times m}$ be computed by the algorithm described above, in a floating point environment satisfying the axioms. Let \tilde{H} be computed Hessenberg matrix and \tilde{Q} be the exactly unitary matrix corresponding to the computed reflection vectors \tilde{v}_k , then*

$$\tilde{Q}\tilde{H}\tilde{Q}^* = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\varepsilon_{mach})$$

for some $\delta A \in \mathbb{C}^{m \times m}$.

Phase#1 — ✓

Phase#2 — ?

We take a small detour and discuss some classical eigenvalue algorithms*; they are useful in their own right under certain circumstances, and will form the foundation for **“Phase#2-algorithms.”**

* The Rayleigh quotient, Power iteration, Inverse Iteration, and Rayleigh quotient iteration.

Restriction to $A \in \mathbb{R}^{m \times m}$, $A = A^*$

For simplicity, we briefly restrict our study to real symmetric matrices, and note that when we are ready to apply these methods (in Phase#2), A will be real, symmetric, and tri-diagonal.

The discussion is simplified since

- 1 we can guarantee that all eigenvalues $\lambda_k(A) \in \mathbb{R}$ are real, and
- 2 A has a complete set of orthonormal eigenvectors, \vec{q}_k .

For real quantities $\vec{x}^* = \vec{x}^T$, and $A^* = A^T$.

The Rayleigh Quotient

The **Rayleigh quotient** — after Lord Rayleigh (John William Strutt), Nobel Prize in Physics 1904, "*for his investigations of the densities of the most important gases and for his discovery of argon in connection with these studies.*"



Figure: Lord Rayleigh.

— of a vector $\vec{x} \in \mathbb{R}^m$ — is the scalar quantity

$$r(\vec{x}) = \frac{\vec{x}^* A \vec{x}}{\vec{x}^* \vec{x}}.$$

We note that if $\vec{x} = \vec{q}_k$ is an eigenvector, then $r(\vec{q}_k) = \lambda_k$.

(Copyright — Figure of Lord Rayleigh) — This file comes from Wellcome Images, a website operated by Wellcome Trust, a global charitable foundation based in the United Kingdom. Licensed under the Creative Commons Attribution 4.0 International license. File Located at https://commons.wikimedia.org/wiki/File:John_William_Strutt,_3rd_Baron_Rayleigh._Photogravure_after_Wellcome_V0006603.jpg

The Rayleigh Quotient

Interpretation

For a general \vec{x} , $r(\vec{x})$ is the value which “acts most like an eigenvalue” in the least squares sense, *i.e.*

$$r(\vec{x}) = \min_{r \in \mathbb{R}} \|A\vec{x} - r\vec{x}\|_2$$

The corresponding normal equation

$$[\vec{x}^* \vec{x}] r = \vec{x}^* A \vec{x}$$

gives r as the Rayleigh quotient.

The Rayleigh Quotient...

Quadratic Accuracy

1 of 2

Let's view the Rayleigh quotient as a function $r(\vec{x}) : \mathbb{R}^m \mapsto \mathbb{R}$.

We are interested in the local behavior of $r(\vec{x})$ when \vec{x} is close to an eigenvector... We compute the gradient of $r(\vec{x})$

$$\begin{aligned} \frac{\partial}{\partial x_j} r(\vec{x}) &= \frac{1}{\vec{x}^* \vec{x}} \left[\frac{\partial}{\partial x_j} (\vec{x}^* A \vec{x}) \right] - \frac{(\vec{x}^* A \vec{x})}{(\vec{x}^* \vec{x})^2} \left[\frac{\partial}{\partial x_j} (\vec{x}^* \vec{x}) \right] \\ &= \frac{2(A\vec{x})_j}{\vec{x}^* \vec{x}} - \frac{(\vec{x}^* A \vec{x}) 2x_j}{(\vec{x}^* \vec{x})^2} = \frac{2}{\vec{x}^* \vec{x}} \left[A\vec{x} - r(\vec{x})\vec{x} \right]_j, \end{aligned}$$

i.e.

$$\nabla_{\vec{x}} r(\vec{x}) = \frac{2}{\vec{x}^* \vec{x}} \left[A\vec{x} - r(\vec{x})\vec{x} \right].$$

Bottom line: $\nabla_{\vec{x}} r(\vec{x}) = 0$, $\vec{x} \neq 0$ if and only if $(\vec{x}, r(\vec{x}))$ is an eigenvector-eigenvalue pair.

The Rayleigh Quotient...

Quadratic Accuracy

2 of 2

Now, let \vec{q}_k be one of the eigenvectors of A , and let $\vec{x} = (\vec{q}_k + \vec{\epsilon})$, with $\|\vec{\epsilon}\|_2 \ll 1$. By Taylor's theorem

$$r(\vec{x}) - r(\vec{q}_k) = \underbrace{\vec{\epsilon}^* \nabla(r(\vec{q}_k))}_0 + \frac{1}{2} \vec{\epsilon}^* \underbrace{\nabla^2(r(\vec{q}_k + t\vec{\epsilon}))}_{\text{The Hessian}} \vec{\epsilon}, \quad t \in [0, 1].$$

This shows that

$$|r(\vec{x}) - r(\vec{q}_k)| = \mathcal{O}(\|\vec{\epsilon}\|^2), \quad \vec{x} = \vec{q}_k + \vec{\epsilon}.$$

Thus,

Theorem

The Rayleigh quotient is a quadratically accurate estimate of an eigenvalue.

Power Iteration

We have already written this idea off once... but it turns out that it can be made useful.

Algorithm (Power Iteration)

$\vec{v}_{(0)}$ = some vector, so that $\|\vec{v}_{(0)}\|_2 = 1$

$k = 0$

while(*termination criteria (details swept under the rug)*)

$k = k + 1$

$\vec{w} = A\vec{v}_{(k-1)}$

$\vec{v}_{(k)} = \vec{w} / \|\vec{w}\|$

$\lambda_{(k)} = \vec{v}_{(k)}^* A \vec{v}_{(k)}$

endwhile

This algorithm produces a sequence of approximate eigenvalue-vector pairs $(\lambda_{(k)}, \vec{v}_{(k)})$ which converge to $(\lambda_{\max}, \vec{q}_{\max})$

Power Iteration

Convergence

Theorem

Suppose $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_m| \geq 0$ and $\vec{q}_1^* \vec{v}_{(0)} \neq 0$. Then the iterates of the power iteration satisfy

$$\|\vec{v}_{(k)} \mp \vec{q}_1\| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right), \quad |\lambda_{(k)} - \lambda_1| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$$

As it stands this is not very useful —

- (1) We can only find the eigenvector corresponding to the largest eigenvalue;
- (2) convergence for the eigenvector is only linear;
- (3) the convergence factor $|\lambda_2/\lambda_1|$ can be very close to 1.

It turns out we can use this basic idea (power iteration) to build scheme where we can guarantee that $|\lambda_2/\lambda_1|$ is small, and further we can find any eigenvector...

Inverse Iteration

Motivation: For any $\mu \in \mathbb{R}$ that is **not** an eigenvalue of A , the eigenvectors of

$$A \quad \text{and} \quad (\mathbf{A} - \mu \mathbf{I})^{-1},$$

are the same, and the corresponding eigenvalues are

$$\lambda_j \quad \text{and} \quad \frac{1}{\lambda_j - \mu}.$$

Suppose μ is close to λ_k for some k , then since $\lim_{\mu \rightarrow \lambda_k} \frac{1}{\lambda_k - \mu} = \infty$, this suggests that

$$\frac{1}{|\lambda_k - \mu|} \gg \frac{1}{|\lambda_j - \mu|}, \quad j \neq k.$$

Thus applying power iteration to $(A - \mu I)^{-1}$ should give rapid convergence to \vec{q}_k .

Inverse Iteration

The Algorithm

Algorithm (Inverse Iteration)

$\vec{v}_{(0)}$ = some vector, so that $\|\vec{v}_{(0)}\|_2 = 1$

$k = 0$

while(*termination criteria (details swept under the rug)*)

$k = k + 1$

 Solve^[1] $(A - \mu I)\vec{w} = \vec{v}_{(k-1)}$ for \vec{w}

$\vec{v}_{(k)} = \vec{w} / \|\vec{w}\|$

$\lambda_{(k)} = \vec{v}_{(k)}^* A \vec{v}_{(k)}$

endwhile

Even though $(A - \mu I)$ becomes singular as $\mu \rightarrow \lambda_k$, the solution $\vec{w} = (A - \mu I)^{-1} \vec{v}_{(k-1)}$ still gives a good **rescaled** $\vec{v}_{(k)} = \vec{w} / \|\vec{w}\|$.

[1] Solve by QR-, or Cholesky-factorization.

Inverse Iteration

Discussion

Like power iteration, inverse iteration only exhibits linear convergence.

However, the positive features are

- We can **choose** what eigenvector to compute by supplying and estimate μ of the corresponding eigenvalue.
- We can control the rate of linear convergence since for $\mu \approx \lambda_k$

$$\left| \frac{\lambda_2([A - \mu I]^{-1})}{\lambda_1([A - \mu I]^{-1})} \right| = \max_{j \neq k} \left| \frac{\lambda_k - \mu}{\lambda_j - \mu} \right| \ll 1.$$

We make this precise in a theorem...

Inverse Iteration

Convergence Theorem

Theorem

Suppose that λ_J is the closest eigenvalue to μ , and λ_K is the second closest, i.e. $|\mu - \lambda_J| < |\mu - \lambda_K| \leq |\mu - \lambda_j|, \forall j \notin \{J, K\}$. Furthermore, assume $\vec{q}_J^* \vec{v}_{(0)} \neq 0$. Then the iterates of the inverse iteration satisfy

$$\|\vec{v}_{(k)} \mp \vec{q}_J\| = \mathcal{O}\left(\left|\frac{\mu - \lambda_J}{\mu - \lambda_K}\right|^k\right), \quad |\lambda_{(k)} - \lambda_J| = \mathcal{O}\left(\left|\frac{\mu - \lambda_J}{\mu - \lambda_K}\right|^{2k}\right)$$

Inverse iteration is the **standard method** for calculating the eigenvectors of a matrix if the eigenvalues are already known. In this setting, the algorithm is applied as described, but the calculation of the Rayleigh quotient $\lambda_{(k)} = \vec{v}_{(k)}^* A \vec{v}_{(k)}$ is skipped.

Rayleigh Quotient + Inverse Iteration = Rayleigh Quotient Iteration

Rayleigh Quotient Get an eigenvalue estimate from a eigenvector estimate.

Inverse Iteration Get an eigenvector estimate from an eigenvalue estimate.



Mix them together, and **BAM!!!**

Algorithm (Rayleigh Quotient Iteration)

$\vec{v}_{(0)}$ = some vector, so that $\|\vec{v}_{(0)}\|_2 = 1$

$\lambda_{(0)} = \vec{v}_{(0)}^* A \vec{v}_{(0)}$, $k = 0$

while(*termination criteria (details swept under the rug)*)

$k = k + 1$

 Solve $(A - \lambda_{(k-1)} I) \vec{w} = \vec{v}_{(k-1)}$ for \vec{w}

$\vec{v}_{(k)} = \vec{w} / \|\vec{w}\|$

$\lambda_{(k)} = \vec{v}_{(k)}^* A \vec{v}_{(k)}$

endwhile

Rayleigh Quotient Iteration

Cubic Convergence

Theorem (Convergence of the Rayleigh Quotient Iteration)

*Rayleigh Quotient Iteration converges to an eigenvalue-eigenvector pair for all, except a set of measure zero, starting vectors $\vec{v}_{(0)}$. When it converges, the convergence is ultimately **cubic** in the sense that if λ_J is an eigenvalue of A and $\vec{v}_{(0)}$ is sufficiently close to the eigenvector \vec{q}_J , then*

$$\|\vec{v}_{(k+1)} \mp \vec{q}_J\| = \mathcal{O}\left(\|\vec{v}_{(k)} \mp \vec{q}_J\|^3\right)$$

and

$$|\lambda_{(k+1)} - \lambda_J| = \mathcal{O}\left(|\lambda_{(k)} - \lambda_J|^3\right)$$

as $k \rightarrow \infty$. The \mp signs are not necessarily the same on the two sides of the equalities.

Rayleigh Quotient Iteration

Convergence

Informal Pattern

$$\|\vec{v}_{(k)} \mp \vec{q}_J\| \qquad |\lambda_{(k)} - \lambda_J|$$

$$\mathcal{O}(\epsilon) \rightarrow \mathcal{O}(\epsilon^2)$$

$$\downarrow \swarrow$$

$$\mathcal{O}(\epsilon^3) \rightarrow \mathcal{O}(\epsilon^6)$$

$$\downarrow \swarrow$$

$$\mathcal{O}(\epsilon^9) \rightarrow \mathcal{O}(\epsilon^{18})$$

$$\vdots \qquad \vdots$$

$\mathcal{O}(\epsilon^k) \rightarrow \mathcal{O}(\epsilon^{2k})$ comes from quadratic accuracy of the Rayleigh quotient. $\{\mathcal{O}(\epsilon^k), \mathcal{O}(\epsilon^{2k})\} \rightarrow \mathcal{O}(\epsilon^{3k})$, since for the inverse iteration

$$\|\vec{v}_{(k)} \mp \vec{q}_J\| = \mathcal{O} \left(\left| \frac{\lambda_{(k)} - \lambda_J}{\lambda_{(k)} - \lambda_K} \right| \cdot \|\vec{v}_{(k-1)} \mp \vec{q}_J\| \right) = \mathcal{O}(\epsilon^{2k}) \cdot \mathcal{O}(\epsilon^k) = \mathcal{O}(\epsilon^{3k})$$

Work per Iteration...

$A \in \mathbb{R}^{m \times m}$, Full, $A^* = A$

Power Iteration	$\mathcal{O}(m^2)$	
Inverse Iteration	$\mathcal{O}(m^2)$	LU, QR, or Cholesky
Inverse Iteration	$\mathcal{O}(m^3)$	Unfactored
Rayleigh Quotient Iteration	$\mathcal{O}(m^3)$	$(A - \lambda_{(k)}I)$ changes ^[1]

$A \in \mathbb{R}^{m \times m}$, Tri-Diagonal, $A^* = A$ **Hessenberg, $A^* \neq A$**

Power Iteration	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$
Inverse Iteration	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$
Rayleigh Quotient Iteration	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$

[1] Unless we can find an update formula for the factorization of $(A - \lambda_{(k)}I)$, beating $\mathcal{O}(m^3)$ operations per iteration is hard...

Homework #7

Due Date in Canvas/Gradescope

[SUBMIT] Trefethen-&-Bau 24.3

Hints: 1: Use `expm` or `scipy.linalg.expm` (not the `exp` versions) for matrix exponentiation e^{tA} . 2: Make sure you have many points in the interval of interest, e.g. use `linspace/np.linspace` with at least 100 points. 3: It is useful to (additionally) plot $\|e^{tA}\|_2/e^{t\alpha(A)}$.

[SUBMIT] Implement-and-Test — Householder Reduction to Hessenberg form.

- Submit: Code + Validation, show working (5×5) and (7×7) examples.
- Compare with a library call (e.g. `hess/scipy.linalg.hessenberg`) — for validation use a (9×9) example. *Comment on similarities and differences.*

[SUBMIT] Implement-and-Test — Rayleigh Quotient Iteration.

- Submit: Code + Validation
- Minimum Validation: (11×11) matrix; (explicitly) show that at least one eigenvalue–eigenvector pair matches library (matlab/python) call.

Trefethen-&-Bau 26.1, 26.3, 27.3 — Read and think.

[OPTIONAL] Trefethen-&-Bau 26.2 (bonus fun) —

Use `eigtool` (<http://www.cs.ox.ac.uk/projects/pseudospectra/eigtool/>) or `pseudopy` (<https://github.com/andrenarchy/pseudopy>) to compute the pseudospectra