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

---

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩
Three factorizations which expose the eigenvalues of a matrix.

<table>
<thead>
<tr>
<th>Type</th>
<th>Form</th>
<th>Restrictions on $A$</th>
<th>Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonalization</td>
<td>$A = X \Lambda X^{-1}$</td>
<td>Non-defective</td>
<td>✓</td>
</tr>
<tr>
<td>Unitary Diagonalization</td>
<td>$A = Q \Lambda Q^*$</td>
<td>Normal, $A^<em>A = AA^</em>$</td>
<td>✓</td>
</tr>
<tr>
<td>Schur Triangularization</td>
<td>$A = QTQ^*$</td>
<td>None</td>
<td>—</td>
</tr>
</tbody>
</table>

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.
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

$$Q_1^*AQ_1$$
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

$$Q_2^* Q_1^* A Q_1 Q_2$$
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

$$Q_3^* Q_2^* Q_1^* A Q_1 Q_2 Q_3$$
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

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

where $T$ is upper triangular.

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

$$
\begin{bmatrix}
\cdots \\
& \lambda_r & -\lambda_i \\
& \lambda_i & \lambda_r \\
& & \cdots \\
\end{bmatrix}, \quad \lambda = \lambda_r \pm \sqrt{-1} \lambda_i
$$
Allowing $2 \times 2$-blocks along the diagonal saves all the overhead of complex arithmetic, and is known as the \textbf{real Schur factorization}.

\textbf{Special Case}

When $A$ is Hermitian, then

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

is also Hermitian, \textit{i.e.} $T = T^*$, and upper triangular $\sim T$ is diagonal.

The eigenvalue computation is usually split into 2 phases — the first phase completes in a finite number of steps and transforms the matrix into \textbf{upper Hessenberg} form, the second phase is iterative and converges ($k \to \infty$) to upper triangular form.
Two-Phase Eigenvalue Computation

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

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

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩
Phase 1 requires $O(m^3)$ operations.

Phase 2 may (in theory) require infinitely many iterations, each of which requires $O(m^2)$ operations. In practice, convergence to $O(\epsilon_{\text{mach}})$ can be achieved in $O(m)$ iterations, so that the total work requirement is $O(m^3)$.

When $A$ is Hermitian, Phase 2 can be executed with only $O(m)$ operations/iteration; thus the total work estimate for the second phase is only $O(m^2)$ in this case. Hence, the "infinite" part of the algorithm is an order of magnitude faster than the "finite" part.
Why Hessenberg Form?

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

Why not go straight for the “big prize,” — $T$???

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

Whoops!!! The multiplication from the right will fill in 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...
Let’s instead use a Householder reflector $Q_1^*$ which ignores the first row (the $\times$s are completely untouched), and introduces zeros as shown below

\[
\begin{bmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & *
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\times & \times & \times & \times & \times \\
* & * & * & * & * \\
0 & * & * & * & * \\
0 & * & * & * & * \\
0 & * & * & * & *
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\times & * & * & * & * \\
\times & * & * & * & * \\
0 & * & * & * & * \\
0 & * & * & * & * \\
0 & * & * & * & *
\end{bmatrix}
\]

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 $Q_1^*A$.

We now repeat the same strategy...
To Hessenberg Form

Elements marked with $*$ are not changed/touched and elements marked with $+$ are changed/touched.
Householder Reduction to Hessenberg Form

Algorithm (Householder Reduction to Hessenberg Form)

Transform \( A \in \mathbb{R}^{m \times m} \) to Hessenberg Form
for \( k = 1:(m-2) \)
\[
\bar{x} = A((k+1):m,k)
\]
\[
\bar{v}_k = \text{sign}(x_1) ||\bar{x}|| \bar{e}_1 + \bar{x}
\]
\[
\bar{v}_k = \bar{v}_k / ||\bar{v}_k||_2
\]
\[
A((k+1):m,k:m) = A((k+1):m,k:m) - 2\bar{v}_k \bar{v}_k^* A((k+1):m,k:m)
\]
\[
A(1:m,(k+1):m) = A(1:m,(k+1):m) - 2(A(1:m,(k+1):m) \bar{v}_k) \bar{v}_k^*
\]
endfor

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

The work needed for Hessenberg reduction is \( \sim \frac{10}{3} m^3 \) operations.
Since Hessenberg reduction contains operations of the form “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).
Theorem

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

\[
\tilde{Q} \tilde{H} \tilde{Q}^* = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\epsilon_{\text{mach}})
\]

for some \( \delta A \in \mathbb{C}^{m \times m} \).
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, $\bar{q}_k$.

For real quantities $\bar{x}^* = \bar{x}^T$, 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" — 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 $r(\vec{q}_k) = \lambda_k$. 

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩
For a general $\bar{x}$, $r(\bar{x})$ is the value which "acts most like an eigenvalue" in the least squares sense, i.e.

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

The normal equation $[\bar{x}^*\bar{x}] r = \bar{x}^*A\bar{x}$ gives $r$ as the Rayleigh quotient.
Let’s view the Rayleigh coefficient as a function $r(\bar{x}) : \mathbb{R}^m \to \mathbb{R}$.

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

$$\frac{\partial r(\bar{x})}{\partial x_j} = \frac{1}{\bar{x}^*\bar{x}} \left[ \frac{\partial}{\partial x_j} (\bar{x}^* A\bar{x}) \right] - \frac{(\bar{x}^* A\bar{x})}{(\bar{x}^*\bar{x})^2} \left[ \frac{\partial}{\partial x_j} (\bar{x}^*\bar{x}) \right]$$

$$= \frac{2(A\bar{x})_j}{\bar{x}^*\bar{x}} - \frac{(\bar{x}^* A\bar{x})2x_j}{(\bar{x}^*\bar{x})^2} = \frac{2}{\bar{x}^*\bar{x}} \left[ A\bar{x} - r(\bar{x})\bar{x} \right]_j,$$

i.e.

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

**Bottom line:** $\nabla r(\bar{x}) = 0, \bar{x} \neq 0$ if and only if $(\bar{x}, r(\bar{x}))$ is an eigenvector-eigenvalue pair.
Now, let $\tilde{q}_k$ be one of the eigenvectors of $A$, and let $\tilde{x} = \tilde{q}_k + \tilde{\epsilon}$, with $\|\tilde{\epsilon}\|_2 \ll 1$. By Taylor’s theorem

$$r(\tilde{x}) - r(\tilde{q}_k) = \tilde{\epsilon}^* \nabla(r(\tilde{q}_k)) + \frac{1}{2} \tilde{\epsilon}^* \nabla^2(r(\tilde{q}_k + t\tilde{\epsilon}))\tilde{\epsilon}, \quad t \in [0, 1].$$

This shows that

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

Thus,

**Theorem**

*The Rayleigh quotient is a quadratically accurate estimate of an eigenvalue.*
We have already written this idea off once… but it turns out that it can be made useful.

**Algorithm (Power Iteration)**

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

\[ k = 0 \]

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

\[ k = k + 1 \]

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

\[ \vec{v}_{(k)} = \frac{\vec{w}}{\|\vec{w}\|} \]

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

endwhile

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

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

\[
\| \bar{v}(k) \mp \bar{q}_1 \| = O \left( \frac{|\lambda_2|}{\lambda_1}^k \right), \quad |\lambda(k) - \lambda_1| = O \left( \frac{|\lambda_2|^{2k}}{\lambda_1} \right)
\]

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

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

are the same, and the corresponding eigenvalues are

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

Suppose $\mu$ is close to $\lambda_k$ for some $k$, then since

$$\lim_{\mu \to \lambda_k} \frac{1}{\lambda_k - \mu} = \infty,$$

this suggests that

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

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

"In the image, the content is presented in a readable format with a header section titled "Inverse Iteration". The text explains the motivation for using inverse iteration in the context of eigenvalue problems.

The key points include:

- For any $\mu \in \mathbb{R}$ that is not an eigenvalue of $A$, the eigenvectors of $A$ and $(A - \mu I)^{-1}$ are the same, and the corresponding eigenvalues are $\lambda_j$ and $\frac{1}{\lambda_j - \mu}$.

- Suppose $\mu$ is close to $\lambda_k$ for some $k$, then the limit $\lim_{\mu \to \lambda_k} \frac{1}{\lambda_k - \mu} = \infty$ suggests that $\frac{1}{|\lambda_k - \mu|} \gg \frac{1}{|\lambda_j - \mu|}$ for $j \neq k$.

- Applying power iteration to $(A - \mu I)^{-1}$ should give rapid convergence to $\bar{\mathbf{q}}_k$."

Peter Blomgren, (blomgren.peter@gmail.com)
Algorithm (Inverse Iteration)

\[
\vec{v}(0) = \text{some vector, so that } \|\vec{v}(0)\|_2 = 1
\]

\[
k = 0
\]

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

\[
k = k + 1
\]

Solve \([A - \mu I]\vec{w} = \vec{v}_{(k-1)}\) 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 \(\mu \to \lambda_k\), 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.
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 $\mu$ of the corresponding eigenvalue.
- We can control the rate of linear convergence since for $\mu \approx \lambda_k$

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

We make this precise in a theorem...
Inverse Iteration

**Theorem**

*Suppose that* $\lambda_J$ *is the closest eigenvalue to* $\mu$, *and* $\lambda_K$ *is the second closest, i.e. $|\mu - \lambda_J| < |\mu - \lambda_K| \leq |\mu - \lambda_j|$, $\forall j \not\in \{J, K\}$. Furthermore, assume* $\bar{q}_J^*\bar{v}(0) \neq 0$. *Then the iterates of the inverse iteration satisfy*

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

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 coefficient $\lambda(k) = \bar{v}^*_k A\bar{v}(k)$ 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.
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)**

\[
\begin{align*}
\bar{v}_{(0)} &= \text{some vector, so that } \|\bar{v}_{(0)}\|_2 = 1 \\
\lambda_{(0)} &= \bar{v}_{(0)}^* A \bar{v}_{(0)}, \quad k = 0 \\
\text{while (termination criteria \textit{(details swept under the rug)})} \\
& \quad k = k + 1 \\
& \quad \text{Solve } (A - \lambda_{(k-1)}I) \bar{w} = \bar{v}_{(k-1)} \text{ for } \bar{w} \\
& \quad \bar{v}_{(k)} = \bar{w}/\|\bar{w}\| \\
& \quad \lambda_{(k)} = \bar{v}_{(k)}^* A \bar{v}_{(k)} \\
\end{align*}
\]
Theorem

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

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

and

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

as $k \to \infty$. The $\mp$ signs are not necessarily the same on the two sides of the equalities.
Rayleigh Quotient Iteration

\[ \| \bar{v}_{(k)} \mp \bar{q}_J \| \quad | \lambda_{(k)} - \lambda_J | \]

\[
\begin{align*}
O(\epsilon) & \quad \rightarrow \quad O(\epsilon^2) \\
\downarrow & \quad \checkmark \\
O(\epsilon^3) & \quad \rightarrow \quad O(\epsilon^6) \\
\downarrow & \quad \checkmark \\
O(\epsilon^9) & \quad \rightarrow \quad O(\epsilon^{18}) \\
\vdots & \quad \vdots
\end{align*}
\]

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

\[
\| \bar{v}_{(k)} \mp \bar{q}_J \| = O \left( \frac{\left| \lambda_{(k)} - \lambda_J \right|}{\lambda_{(k)} - \lambda_K} \cdot \| \bar{v}_{(k-1)} \mp \bar{q}_J \| \right) = O(\epsilon^{2k}) \cdot O(\epsilon^k) = O(\epsilon^{3k})
\]
<table>
<thead>
<tr>
<th></th>
<th>$A \in \mathbb{R}^{m\times m}$, <strong>Full</strong>, $A^* = A$</th>
<th>$A \in \mathbb{R}^{m\times m}$, <strong>Tri-Diagonal</strong>, $A^* = A$</th>
<th><strong>Hessenberg</strong>, $A^* \neq A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Iteration</td>
<td>$O(m^2)$</td>
<td>$O(m)$</td>
<td>$O(m^2)$</td>
</tr>
<tr>
<td>Inverse Iteration</td>
<td>$O(m^2)$</td>
<td>$O(m)$</td>
<td>$O(m^2)$</td>
</tr>
<tr>
<td>Inverse Iteration</td>
<td>$O(m^3)$</td>
<td>$O(m^3)$</td>
<td>Unfactored</td>
</tr>
<tr>
<td>Rayleigh Quotient Iteration</td>
<td>$O(m^3)$</td>
<td>$(A - \lambda_{(k)} I)$ changes(^{[1]})</td>
<td></td>
</tr>
</tbody>
</table>

\(^{[1]}\) Unless we can find an update formula for the factorization of $(A - \lambda_{(k)} I)$, beating $O(m^3)$ operations per iteration is hard...
Homework #6 — Due at 11:00am, Friday April 29

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

Trefethen-&-Bau 24.3

Trefethen-&-Bau 26.2 (extra credit) —

Hint: You may use eigtool downloadable from (http://http://www.cs.ox.ac.uk/projects/pseudospectra/eigtool/) to compute the pseudospectra. Also, use expm (not exp) for matrix exponentiation $e^{tA}$.

Implement-and-Test — Householder Reduction to Hessenberg form. (Compare with e.g. matlab’s hess).

Implement-and-Test — Rayleigh Quotient Iteration.

Trefethen-&-Bau 27.3 — Read and think.