Eigenvalue Algorithms

Numerical Strategies Providing for the Time and Resource Efficient Computation of Eigenvalues for Any General Square Matrix

Jayden Li

Friday, March 7, 2025

The Problem

How can we find the eigenvalues of any $n \times n$ matrix A?

$$A\mathbf{x} = \lambda\mathbf{x}$$

Table of Contents

Section 1

Power Method

Power Method

Power Method

Power Method

Power Method

1. Let x_0 be a random vector in $\mathbb{R}^n$.

Power Method

Power Method

1. Let x_0 be a random vector in $\mathbb{R}^n$.
2. Define $x_{i+1} = \dfrac{Ax_i}{||Ax_i||}$.

Power Method

Power Method

1. Let x_0 be a random vector in $\mathbb{R}^n$.
2. Define $x_{i+1} = \dfrac{Ax_i}{||Ax_i||}$.
3. Iterate the above process for m steps. Now, we have have calculated x_m, which is parallel to $A^m x_0$.

Power Method

Power Method

1. Let x_0 be a random vector in $\mathbb{R}^n$.
2. Define $x_{i+1} = \dfrac{Ax_i}{||Ax_i||}$.
3. Iterate the above process for m steps. Now, we have have calculated x_m, which is parallel to $A^m x_0$.
4. Calculate the dominant eigenvalue $\lambda_1 = \dfrac{\|AA^m x_0\|}{\|A^m x_0\|} = \dfrac{\|Ax_m\|}{\|x_m\|} = \dfrac{Ax_m \cdot x_m}{x_m \cdot x_m}$.

Reference: [Koc25].

Derivation of the Power Method (1)

- A is an $n \times n$ matrix.

Derivation of the Power Method (1)

- A is an $n \times n$ matrix.
- We are basically multiplying on the left by A.

Derivation of the Power Method (1)

- A is an $n \times n$ matrix.
- We are basically multiplying on the left by A.
- Algorithm tends to $\lim_{k \to \infty} A^k x_0$.

Derivation of the Power Method (1)

- A is an $n \times n$ matrix.
- We are basically multiplying on the left by A.
- Algorithm tends to $\lim_{k \to \infty} A^k x_0$.
- Let $A = PDP^{-1}$ be the diagonalization.
- Then $\lim_{k \to \infty} A^k = PD^kP^{-1}$.

Derivation of the Power Method (1)

- A is an $n \times n$ matrix.
- We are basically multiplying on the left by A.
- Algorithm tends to $\lim\limits_{k \to \infty} A^k x_0$.
- Let $A = PDP^{-1}$ be the diagonalization.
- Then $\lim\limits_{k \to \infty} A^k = PD^kP^{-1}$.
- $\lim\limits_{k \to \infty} A^k P = PD^k$.

Derivation of the Power Method (2)

- Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. The λ_i's are eigenvalues of A.

Derivation of the Power Method (2)

- Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. The λ_i's are eigenvalues of A.
- Quick reminder: D is diagonal matrix of eigenvalues.

$$D = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

Derivation of the Power Method (2)

- Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. The λ_i's are eigenvalues of A.
- Quick reminder: D is diagonal matrix of eigenvalues.

$$D = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

- Choose a random vector $v_0 \in \mathbb{R}^n$, let $v = Pv_0$.

Derivation of the Power Method (2)

- Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. The λ_i's are eigenvalues of A.
- Quick reminder: D is diagonal matrix of eigenvalues.

$$D = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

- Choose a random vector $v_0 \in \mathbb{R}^n$, let $v = Pv_0$.
- $\lim\limits_{k\to\infty} A^k P = PD^k$ becomes $\lim\limits_{k\to\infty} A^k P v_0 = PD^k v_0$.

Derivation of the Power Method (3)

- $\lim_{k\to\infty} A^k P = PD^k$ becomes $\lim_{k\to\infty} A^k P v_0 = PD^k v_0 = A^k v$.

Derivation of the Power Method (3)

- $\lim_{k\to\infty} A^k P = PD^k$ becomes $\lim_{k\to\infty} A^k P v_0 = PD^k v_0 = A^k v$.

$$A^k v = P \begin{bmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_n^k \end{bmatrix} \begin{bmatrix} v_{0_1} \\ \vdots \\ v_{0_n} \end{bmatrix} = \begin{bmatrix} p_1 & \dots & p_n \end{bmatrix} \begin{bmatrix} \lambda_1^k v_{0_1} \\ \vdots \\ \lambda_n^k v_{0_n} \end{bmatrix}$$

Derivation of the Power Method (3)

- $\lim_{k\to\infty} A^k P = PD^k$ becomes $\lim_{k\to\infty} A^k P v_0 = PD^k v_0 = A^k v$.

$$A^k v = P \begin{bmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_n^k \end{bmatrix} \begin{bmatrix} v_{0_1} \\ \vdots \\ v_{0_n} \end{bmatrix} = \begin{bmatrix} p_1 & \dots & p_n \end{bmatrix} \begin{bmatrix} \lambda_1^k v_{0_1} \\ \vdots \\ \lambda_n^k v_{0_n} \end{bmatrix}$$

$$A^k v = \sum_{i=1}^{n} \lambda_i^k v_{0_i} p_i = \sum_{i=1}^{n} \lambda_1^k \frac{\lambda_i^k}{\lambda_1^k} v_{0_i} p_i = \lambda_1^k \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k v_{0_i} p_i$$

Derivation of the Power Method (3)

- $\lim_{k\to\infty} A^k P = PD^k$ becomes $\lim_{k\to\infty} A^k P v_0 = PD^k v_0 = A^k v$.

$$A^k v = P \begin{bmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_n^k \end{bmatrix} \begin{bmatrix} v_{0_1} \\ \vdots \\ v_{0_n} \end{bmatrix} = \begin{bmatrix} p_1 & \dots & p_n \end{bmatrix} \begin{bmatrix} \lambda_1^k v_{0_1} \\ \vdots \\ \lambda_n^k v_{0_n} \end{bmatrix}$$

$$A^k v = \sum_{i=1}^{n} \lambda_i^k v_{0_i} p_i = \sum_{i=1}^{n} \lambda_1^k \frac{\lambda_i^k}{\lambda_1^k} v_{0_i} p_i = \lambda_1^k \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k v_{0_i} p_i$$

- Now, if $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$, then $\lambda_i/\lambda_1 < 1$ for all $i \neq 1$.

$$A^k v = \lambda_1^k v_{0_1} p_1$$

Derivation of the Power Method (3)

- $\lim_{k\to\infty} A^k P = PD^k$ becomes $\lim_{k\to\infty} A^k P v_0 = PD^k v_0 = A^k v$.

$$A^k v = P \begin{bmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_n^k \end{bmatrix} \begin{bmatrix} v_{0_1} \\ \vdots \\ v_{0_n} \end{bmatrix} = \begin{bmatrix} p_1 & \dots & p_n \end{bmatrix} \begin{bmatrix} \lambda_1^k v_{0_1} \\ \vdots \\ \lambda_n^k v_{0_n} \end{bmatrix}$$

$$A^k v = \sum_{i=1}^{n} \lambda_i^k v_{0_i} p_i = \sum_{i=1}^{n} \lambda_1^k \frac{\lambda_i^k}{\lambda_1^k} v_{0_i} p_i = \lambda_1^k \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^k v_{0_i} p_i$$

- Now, if $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$, then $\lambda_i / \lambda_1 < 1$ for all $i \neq 1$.

$$A^k v = \lambda_1^k v_{0_1} p_1$$

- A^k is an eigenvector of A.

Derivation of the Power Method (4)

- Now suppose $|\lambda_1| = |\lambda_2|$.

Derivation of the Power Method (4)

- Now suppose $|\lambda_1| = |\lambda_2|$.

$$
\begin{aligned}
A^k v &= \sum_{i=1}^{n} \lambda_i^k v_{0_i} p_i = \sum_{i=1}^{n} \lambda_1^k \frac{\lambda_i^k}{\lambda_1^k} v_{0_i} p_i = \lambda_1^k \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k v_{0_i} p_i \\
&= \lim_{k\to\infty} \lambda_1^k \left(\cancel{\left(\frac{\lambda_1}{\lambda_1}\right)^k} v_{0_1} p_1 + \cancel{\left(\frac{\lambda_2}{\lambda_1}\right)^k} v_{0_2} p_2 + \cancel{\sum_{i=3}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k v_{0_i} p_i} \right) \\
&= \lim_{k\to\infty} \lambda_1^k \left(v_{0_1} p_1 + v_{0_2} p_2 \right)
\end{aligned}
$$

is not an eigenvector of A if $\lambda_1 \neq \lambda_2$.

Problems with the Power Method

- If $|\lambda_1| = |\lambda_2|$, then $(\lambda_1/\lambda_2)^k$ tends to 1, not 0.

Problems with the Power Method

- If $|\lambda_1| = |\lambda_2|$, then $(\lambda_1/\lambda_2)^k$ tends to 1, not 0.
- If dominant eigenvalue $\lambda_1 = a + bi$ is complex $(b \neq 0)$, then $\lambda_2 = \overline{\lambda_1} = a - bi$ is also eigenvalue.

Problems with the Power Method

- If $|\lambda_1| = |\lambda_2|$, then $(\lambda_1/\lambda_2)^k$ tends to 1, not 0.
- If dominant eigenvalue $\lambda_1 = a + bi$ is complex $(b \neq 0)$, then $\lambda_2 = \overline{\lambda_1} = a - bi$ is also eigenvalue.
- But $|\lambda_1| = \sqrt{a^2 + b^2} = |\lambda_2|$.

Problems with the Power Method

- If $|\lambda_1| = |\lambda_2|$, then $(\lambda_1/\lambda_2)^k$ tends to 1, not 0.
- If dominant eigenvalue $\lambda_1 = a + bi$ is complex $(b \neq 0)$, then $\lambda_2 = \overline{\lambda_1} = a - bi$ is also eigenvalue.
- But $|\lambda_1| = \sqrt{a^2 + b^2} = |\lambda_2|$.
- Cannot calculate complex eigenvalues.

Problems with the Power Method

- If $|\lambda_1| = |\lambda_2|$, then $(\lambda_1/\lambda_2)^k$ tends to 1, not 0.
- If dominant eigenvalue $\lambda_1 = a + bi$ is complex $(b \neq 0)$, then $\lambda_2 = \overline{\lambda_1} = a - bi$ is also eigenvalue.
- But $|\lambda_1| = \sqrt{a^2 + b^2} = |\lambda_2|$.
- Cannot calculate complex eigenvalues.
- Can use shifts (calculate eigenvalues of $A - cI$), but that is complicated.

Problems with the Power Method

- If $|\lambda_1| = |\lambda_2|$, then $(\lambda_1/\lambda_2)^k$ tends to 1, not 0.
- If dominant eigenvalue $\lambda_1 = a + bi$ is complex $(b \neq 0)$, then $\lambda_2 = \overline{\lambda_1} = a - bi$ is also eigenvalue.
- But $|\lambda_1| = \sqrt{a^2 + b^2} = |\lambda_2|$.
- Cannot calculate complex eigenvalues.
- Can use shifts (calculate eigenvalues of $A - cI$), but that is complicated.
- Only calculates dominant eigenvalue.

Problems with the Power Method

- If $|\lambda_1| = |\lambda_2|$, then $(\lambda_1/\lambda_2)^k$ tends to 1, not 0.
- If dominant eigenvalue $\lambda_1 = a + bi$ is complex $(b \neq 0)$, then $\lambda_2 = \overline{\lambda_1} = a - bi$ is also eigenvalue.
- But $|\lambda_1| = \sqrt{a^2 + b^2} = |\lambda_2|$.
- Cannot calculate complex eigenvalues.
- Can use shifts (calculate eigenvalues of $A - cI$), but that is complicated.
- Only calculates dominant eigenvalue.
- Speed of convergence depends on choice of starting vector.

Section 2

Basic QR Algorithm

Basic QR Algorithm

Schur form

Let A be an $n \times n$ real matrix. The **Schur form** of A is:

$$A = QUQ^T = QUQ^{-1}$$

where Q is orthogonal and U is upper triangular. A and U are similar so they share the same eigenvalues, which are the diagonal entries of U.

Basic QR Algorithm

Basic QR Algorithm

Basic QR Algorithm

Basic QR Algorithm

1. Set $A_1 = A$.

Basic QR Algorithm

Basic QR Algorithm

1. Set $A_1 = A$.
2. Define $A_{i+1} = R_i Q_i$, where $A_1 = Q_i R_i$.

Basic QR Algorithm

Basic QR Algorithm

1. Set $A_1 = A$.
2. Define $A_{i+1} = R_i Q_i$, where $A_1 = Q_i R_i$.
3. Iterate step 2 m times to calculate A_{m+1}.

Basic QR Algorithm

Basic QR Algorithm

1. Set $A_1 = A$.
2. Define $A_{i+1} = R_i Q_i$, where $A_1 = Q_i R_i$.
3. Iterate step 2 m times to calculate A_{m+1}.
4. If "good case," A_{m+1} tends to a triangular matrix (U in Schur form). Trivial to read off eigenvalues.

Reference: [Arb16, p. 64].

Basic QR Algorithm

Basic QR Algorithm

1. Set $A_1 = A$.
2. Define $A_{i+1} = R_i Q_i$, where $A_1 = Q_i R_i$.
3. Iterate step 2 m times to calculate A_{m+1}.
4. If "good case," A_{m+1} tends to a triangular matrix (U in Schur form). Trivial to read off eigenvalues.

Reference: [Arb16, p. 64].

Proof is hard and complicated. We will "prove" by example.

Example of the Basic QR Algorithm (1)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix}$$

Eigenvalues

$\lambda_1 \approx 12.4542$

$\lambda_2 \approx -5.0744$

$\lambda_3 \approx -0.379762$

Example of the Basic QR Algorithm (2)

$$A_1 = \begin{bmatrix} 1.00 & 2.00 & 3.00 \\ 4.00 & 5.00 & 6.00 \\ 7.00 & 8.00 & 1.00 \end{bmatrix}$$

Example of the Basic QR Algorithm (2)

$$A_1 = \begin{bmatrix} 1.00 & 2.00 & 3.00 \\ 4.00 & 5.00 & 6.00 \\ 7.00 & 8.00 & 1.00 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} -0.12 & 0.90 & 0.41 \\ -0.49 & 0.30 & -0.82 \\ -0.86 & -0.30 & 0.41 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} -8.12 & -9.60 & -4.19 \\ 0.00 & 0.90 & 4.22 \\ 0.00 & 0.00 & -3.27 \end{bmatrix}$$

Example of the Basic QR Algorithm (2)

$$A_1 = \begin{bmatrix} 1.00 & 2.00 & 3.00 \\ 4.00 & 5.00 & 6.00 \\ 7.00 & 8.00 & 1.00 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} -0.12 & 0.90 & 0.41 \\ -0.49 & 0.30 & -0.82 \\ -0.86 & -0.30 & 0.41 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} -8.12 & -9.60 & -4.19 \\ 0.00 & 0.90 & 4.22 \\ 0.00 & 0.00 & -3.27 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 9.33 & -8.98 & 2.81 \\ -4.08 & -1.00 & 0.98 \\ 2.81 & 0.98 & -1.33 \end{bmatrix}$$

Example of the Basic QR Algorithm (3)

$$A_2 = \begin{bmatrix} 9.33 & -8.98 & 2.81 \\ -4.08 & -1.00 & 0.98 \\ 2.81 & 0.98 & -1.33 \end{bmatrix}$$

Example of the Basic QR Algorithm (3)

$$A_2 = \begin{bmatrix} 9.33 & -8.98 & 2.81 \\ -4.08 & -1.00 & 0.98 \\ 2.81 & 0.98 & -1.33 \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} -0.88 & -0.47 & 0.02 \\ 0.39 & -0.70 & 0.60 \\ -0.27 & 0.54 & 0.80 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} -10.57 & 7.28 & -1.75 \\ 0.00 & 5.44 & -2.73 \\ 0.00 & 0.00 & -0.42 \end{bmatrix}$$

Example of the Basic QR Algorithm (3)

$$A_2 = \begin{bmatrix} 9.33 & -8.98 & 2.81 \\ -4.08 & -1.00 & 0.98 \\ 2.81 & 0.98 & -1.33 \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} -0.88 & -0.47 & 0.02 \\ 0.39 & -0.70 & 0.60 \\ -0.27 & 0.54 & 0.80 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} -10.57 & 7.28 & -1.75 \\ 0.00 & 5.44 & -2.73 \\ 0.00 & 0.00 & -0.42 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 12.61 & -1.09 & 2.74 \\ 2.83 & -5.28 & 1.08 \\ 0.11 & -0.22 & -0.33 \end{bmatrix}$$

Example of the Basic QR Algorithm (4)

$$A_4 = \begin{bmatrix} 12.15 & -4.87 & -3.01 \\ -1.07 & -4.77 & -0.65 \\ 0.00 & 0.02 & -0.38 \end{bmatrix}$$

Example of the Basic QR Algorithm (4)

$$A_4 = \begin{bmatrix} 12.15 & -4.87 & -3.01 \\ -1.07 & -4.77 & -0.65 \\ 0.00 & 0.02 & -0.38 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} 12.54 & -3.34 & 2.96 \\ 0.46 & -5.16 & 0.93 \\ 0.00 & -0.00 & -0.38 \end{bmatrix}$$

Example of the Basic QR Algorithm (4)

$$A_4 = \begin{bmatrix} 12.15 & -4.87 & -3.01 \\ -1.07 & -4.77 & -0.65 \\ 0.00 & 0.02 & -0.38 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} 12.54 & -3.34 & 2.96 \\ 0.46 & -5.16 & 0.93 \\ 0.00 & -0.00 & -0.38 \end{bmatrix}$$

$$A_{11} = \begin{bmatrix} 12.45 & -3.79 & 2.98 \\ 0.00 & -5.07 & 0.85 \\ 0.00 & -0.00 & -0.38 \end{bmatrix}$$

Example of the Basic QR Algorithm (5)

$$A_{11} = \begin{bmatrix} \boxed{12.45} & -3.79 & 2.98 \\ 0.00 & \boxed{-5.07} & 0.85 \\ 0.00 & -0.00 & \boxed{-0.38} \end{bmatrix}$$

Eigenvalues

$\lambda_1 \approx 12.4542$

$\lambda_2 \approx -5.0744$

$\lambda_3 \approx -0.379762$

We found <u>EVERY</u> eigenvalue!

Is it good enough?

- Similar convergence problems to Power Method: only guaranteed to work if $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$

Is it good enough?

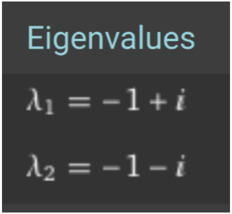
- Similar convergence problems to Power Method: only guaranteed to work if $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$
- Needs modifications to find complex eigenvalues.

Is it good enough?

- Similar convergence problems to Power Method: only guaranteed to work if $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$
- Needs modifications to find complex eigenvalues.
- Could be faster: runs in $\mathcal{O}(n^3)$ time for $n \times n$ matrix.

Bad Example (1)

$$A = \begin{bmatrix} 1 & 5 \\ -1 & -3 \end{bmatrix}$$



Bad Example (1)

$$A = \begin{bmatrix} 1 & 5 \\ -1 & -3 \end{bmatrix}$$

Eigenvalues

$$\lambda_1 = -1 + i$$

$$\lambda_2 = -1 - i$$

$$|\lambda_1| = \sqrt{(-1)^2 + 1^2} = \sqrt{2}$$
$$|\lambda_2| = \sqrt{(-1)^2 + (-1)^2} = \sqrt{2}$$

Bad Example (2)

$$A_1 = \begin{bmatrix} 1.00 & 5.00 \\ -1.00 & -3.00 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -3.00 & -5.00 \\ 1.00 & 1.00 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} -1.40 & 5.80 \\ -0.20 & -0.60 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} -0.60 & -5.80 \\ 0.20 & -1.40 \end{bmatrix}$$

Bad Example (2)

$$A_1 = \begin{bmatrix} 1.00 & 5.00 \\ -1.00 & -3.00 \end{bmatrix} \qquad A_5 = \begin{bmatrix} 1.00 & 5.00 \\ -1.00 & -3.00 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -3.00 & -5.00 \\ 1.00 & 1.00 \end{bmatrix} \qquad A_6 = \begin{bmatrix} -3.00 & -5.00 \\ 1.00 & 1.00 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} -1.40 & 5.80 \\ -0.20 & -0.60 \end{bmatrix} \qquad A_7 = \begin{bmatrix} -1.40 & 5.80 \\ -0.20 & -0.60 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} -0.60 & -5.80 \\ 0.20 & -1.40 \end{bmatrix} \qquad A_8 = \begin{bmatrix} -0.60 & -5.80 \\ 0.20 & -1.40 \end{bmatrix}$$

Bad Example (2)

$$A_1 = \begin{bmatrix} 1.00 & 5.00 \\ -1.00 & -3.00 \end{bmatrix} \qquad A_5 = \begin{bmatrix} 1.00 & 5.00 \\ -1.00 & -3.00 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -3.00 & -5.00 \\ 1.00 & 1.00 \end{bmatrix} \qquad A_6 = \begin{bmatrix} -3.00 & -5.00 \\ 1.00 & 1.00 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} -1.40 & 5.80 \\ -0.20 & -0.60 \end{bmatrix} \qquad A_7 = \begin{bmatrix} -1.40 & 5.80 \\ -0.20 & -0.60 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} -0.60 & -5.80 \\ 0.20 & -1.40 \end{bmatrix} \qquad A_8 = \begin{bmatrix} -0.60 & -5.80 \\ 0.20 & -1.40 \end{bmatrix}$$

$\lim_{k\to\infty} A_k$ will not converge to a triangular matrix.

Another Bad Example

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Another Bad Example

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\lambda_1 = 1, \lambda_2 = -1, \lambda_3 = 1, \lambda_4 = -1$$

Another Bad Example

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\lambda_1 = 1, \lambda_2 = -1, \lambda_3 = 1, \lambda_4 = -1$$

$$A_1 = A = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{Q_1} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_1} = A_1 I$$

Another Bad Example

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\lambda_1 = 1, \lambda_2 = -1, \lambda_3 = 1, \lambda_4 = -1$$

$$A_1 = A = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{Q_1} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_1} = A_1 I$$

$$A_2 = R_1 Q_1 = I A_1 = A_1$$

Another Bad Example

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\lambda_1 = 1, \lambda_2 = -1, \lambda_3 = 1, \lambda_4 = -1$$

$$A_1 = A = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{Q_1} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_1} = A_1 I$$

$$A_2 = R_1 Q_1 = I A_1 = A_1$$

This will never converge to a triangular matrix, since $A_i = A$ for all i.

Section 3

Improved QR Algorithm

Improved QR Algorithm

Hessenberg matrix

Improved QR Algorithm

Hessenberg matrix

A Hessenberg matrix is a matrix with nonzero entries only in the diagonal just above or just below its main diagonal. A **lower** Hessenberg matrix H satisfies $H_{ij} = 0$ for all $j > i + 1$, and an **upper** Hessenberg matrix H satisfies $H_{ij} = 0$ for all $i > j + 1$.

Improved QR Algorithm

Hessenberg matrix

A Hessenberg matrix is a matrix with nonzero entries only in the diagonal just above or just below its main diagonal. A **lower** Hessenberg matrix H satisfies $H_{ij} = 0$ for all $j > i + 1$, and an **upper** Hessenberg matrix H satisfies $H_{ij} = 0$ for all $i > j + 1$.

$$H_l = \begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \qquad H_u = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

Improved QR Algorithm

Used by the Linear Algebra PACKage (LAPACK), which is used by NumPy and SciPy behind the scenes.

LAPACK Algorithm

Improved QR Algorithm

Used by the Linear Algebra PACKage (LAPACK), which is used by NumPy and SciPy behind the scenes.

LAPACK Algorithm

1. Reduce A to an *upper Hessenberg* form by calculating $A = QHQ^T$, where Q is orthogonal and H is an upper Hessenberg matrix. H and A are similar and share the same eigenvalues.

Improved QR Algorithm

Used by the Linear Algebra PACKage (LAPACK), which is used by NumPy and SciPy behind the scenes.

LAPACK Algorithm

1. Reduce A to an *upper Hessenberg* form by calculating $A = QHQ^T$, where Q is orthogonal and H is an upper Hessenberg matrix. H and A are similar and share the same eigenvalues.
2. Calculate the Schur form of H using the Francis double-step algorithm.

Reference: [Bla99].

Hessenberg Example

$$A = \begin{bmatrix} 5 & 4 & 2 & 0 & 59 \\ 12 & 5 & 3 & 12 & 6 \\ 96 & 4 & 696 & 12 & 3 \\ 23 & 4 & 1 & 2 & 2 \\ 66 & 7 & 8 & 22 & 1 \end{bmatrix}$$

Hessenberg Example

$$A = \begin{bmatrix} 5 & 4 & 2 & 0 & 59 \\ 12 & 5 & 3 & 12 & 6 \\ 96 & 4 & 696 & 12 & 3 \\ 23 & 4 & 1 & 2 & 2 \\ 66 & 7 & 8 & 22 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 5 & -34.64 & 43.54 & -4.41 & 19.65 \\ -119.35 & 461.78 & 325.31 & 0.33 & 22.18 \\ 0 & 324.49 & 245.37 & 8.50 & -9.67 \\ 0 & 0 & 10.68 & 2.38 & -6.14 \\ 0 & 0 & 0 & -1.57 & -5.52 \end{bmatrix}$$

Advantages

- Calculates every eigenvalue.

Advantages

- Calculates every eigenvalue.
- Works on all matrices (because of the fancy Schur algorithm).

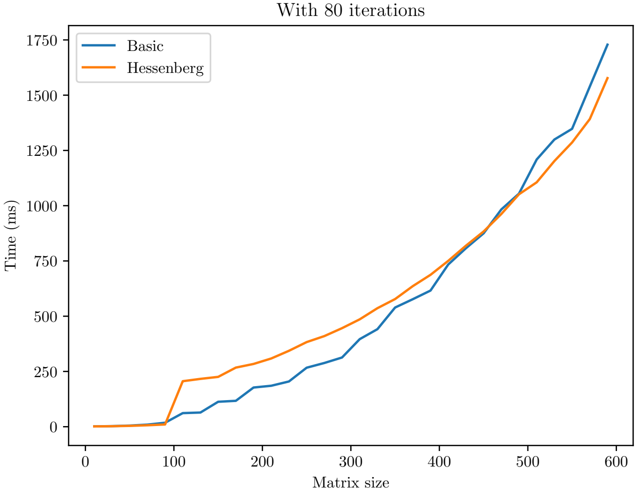
Advantages

- Calculates every eigenvalue.
- Works on all matrices (because of the fancy Schur algorithm).
- Hessenberg reduction reduces each QR step to $\mathcal{O}(n^2)$.

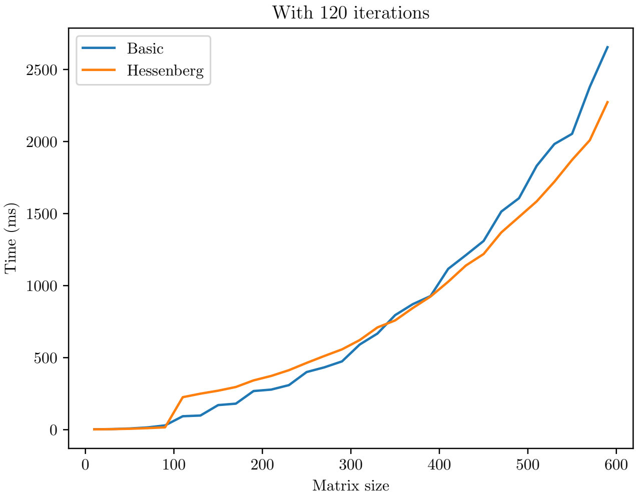
Advantages

- Calculates every eigenvalue.
- Works on all matrices (because of the fancy Schur algorithm).
- Hessenberg reduction reduces each QR step to $\mathcal{O}(n^2)$.
- Very easy to recover complex eigenvalues.

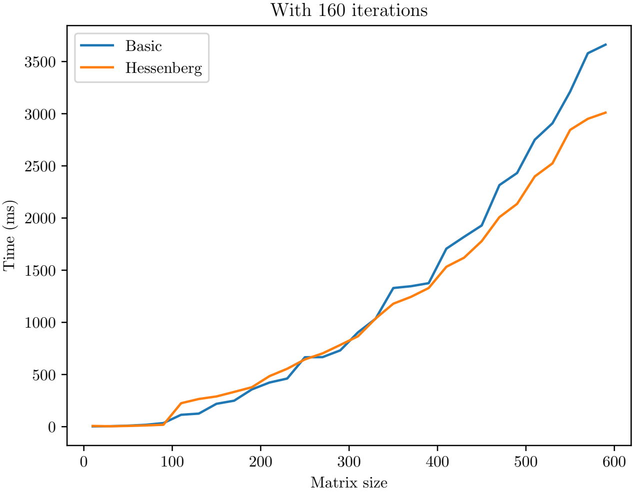
Basic vs Improved QR Algorithm



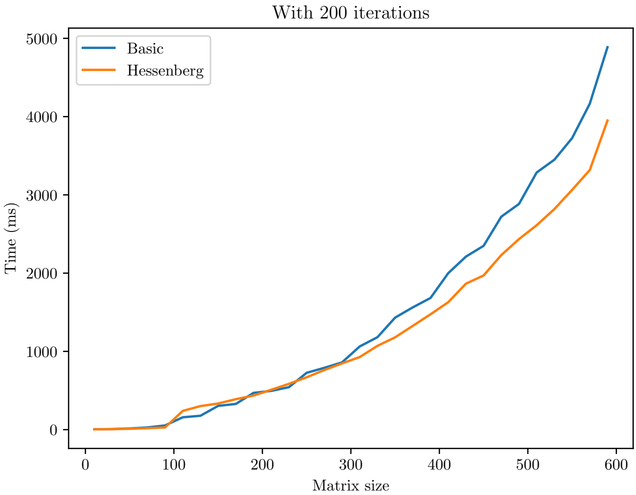
Basic vs Improved QR Algorithm



Basic vs Improved QR Algorithm



Basic vs Improved QR Algorithm



References

[Arb16] Peter Arbenz. *Lecture Notes on Solving Large Scale Eigenvalue Problems*. 2016. URL: https://people.inf.ethz.ch/arbenz/ewp/Lnotes/lsevp.pdf.

[Bla99] Susan Blackford. *Eigenvalues, Eigenvectors and Schur Factorization*. Oct. 1, 1999. URL: https://www.netlib.org/lapack/lug/node50.html.

[Koc25] Gregory Koch. *Lab #9: Power Method for Approximating Eigenvalues*. Feb. 10, 2025. URL: https://peddie.instructure.com/courses/7772/assignments/182563.