Lecture on
Eigenvalue Approximations

Eskil Hansen

Approx. chap. 12.1 in Sauer
The algebraic eigenvalue problem

Given an \( n \times n \) matrix \( A \), find a scalar \( \lambda \) and a vector \( x \neq 0 \) such that

\[
Ax = \lambda x
\]

**Terminology:**

- \( \lambda \) is an eigenvalue
- \( x \) is the corresponding eigenvector

**Properties:**

- \( n + 1 \) unknowns \((x, \lambda)\), but only \( n \) equations
- Impose one extra requirement \( \|x\| = 1 \)

**Equation system:**

\[
(A - \lambda I)x = 0 \quad \Rightarrow
\]

- \( x = 0 \) (trivial solution, ruled out by \( \|x\| = 1 \)), or
- \( \det(A - \lambda I) = 0 \), the characteristic equation
The characteristic equation

The characteristic polynomial \( P(\lambda) := \det(A - \lambda I) \) has degree \( n \).

**Fundamental thm of algebra:** \( P(\lambda) = 0 \) has \( n \) complex roots \( \lambda_i \).

**Theorem:** An \( n \times n \) matrix \( A \) has \( n \) eigenvalues.

If \( 0 \neq x_i \in \mathcal{N}(A - \lambda_i I) \) then \( x_i \) is an eigenvector to \( \lambda_i \):

\[
(A - \lambda_i I)x_i = 0
\]

**Note:** If \( x \) is an eigenvector then so is \( \alpha x \).
Complex vector spaces

Because eigenvalues and -vectors are often complex, consider

\[ A : \mathbb{C}^n \to \mathbb{C}^n; \quad A \in \mathbb{C}^{n \times n} \]

Complex transposition:

vectors: \( x^H = \bar{x}^T \) and matrices: \( A^H = \bar{A}^T \)

Euclidean norm:

\[ \|x\|_2^2 = x^H x \]

Eigenvalue problem:

\[ Ax = \lambda x \quad \text{and} \quad x^H x = 1 \]
Eigenvalue computation

Solving the eigenvalue problem is “equivalent” to solving a polynomial equation $P(\lambda) = 0$.

**Abel:** If $\deg(P) > 4$ the roots cannot be found analytically.

**Methods for computing eigenvalues are iterative.**

**Computational methods:**
- Power iteration
- Inverse iteration
- $QR$ iteration

**Note:** Do not construct and solve the characteristic equation (very sensitive to perturbations).
The power method

Take any vector \(y_0\) and compute iteratively

\[ y_{k+1} = Ay_k \quad \Leftrightarrow \quad y_k = A^k y_0. \]

Assume \(Ax_i = \lambda_i x_i\) and \(y_0 = \sum_1^n \alpha_i x_i\). Then

\[ y_k = A^k y_0 = A^k \sum_{i=1}^n \alpha_i x_i = \sum_{i=1}^n \alpha_i A^k x_i = \sum_{i=1}^n \alpha_i \lambda_i^k x_i \]

Assume \(|\lambda_1| > |\lambda_i|, \quad i = 2, \ldots, n\). Then

\[ y_k = \lambda_1^k \left( \alpha_1 x_1 + \sum_{i=2}^n \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k x_i \right) \approx \lambda_1^k \alpha_1 x_1 \quad \text{for } k \gg 1 \]

**Problem:** In general \(\lim_{k \to \infty} \lambda_1^k = 0\) or \(\infty\).
The power method...

Rayleigh quotient:

\[ \sigma = \frac{y^H A y}{y^H y} \]

If \( y \) is an eigenvector with eigenvalue \( \lambda \), then \( \sigma = \lambda \).

Power iteration algorithm: Choose \( y_0 \) and

\[ \hat{y}_k = \frac{y_k}{\|y_k\|_2} \]

\[ y_{k+1} = A \hat{y}_k \]

\[ \sigma_k = \hat{y}_k^H y_{k+1} \]

Note Converges \( \sigma_k \to \max_i |\lambda_i| \), if this is strictly greater than the other eigenvalues. Convergence is linear and often slow.

Normalization step avoids overflow/underflow.
The power method: Convergence

Nonsymmetric matrices: linear convergence
Symmetric matrices: quadratic convergence

Example:

\[
A = \begin{pmatrix}
12 & 2 & 2.05 \\
2 & 9 & 1 \\
1.95 & 1 & 7 \\
\end{pmatrix}
\]

\[
B = \begin{pmatrix}
12 & 2 & 2 \\
2 & 9 & 1 \\
2 & 1 & 7 \\
\end{pmatrix}
\]
Inverse iteration (extra material)

Reverse the power iteration:

\[ y_k = Ay_{k+1} \iff y_{k+1} = A^{-1}y_k \]

Note:

\[ Ax = \lambda x \implies \begin{align*}
    \bullet & \quad (A - sI)x = (\lambda - s)x \\
    \bullet & \quad A^{-1}x = \lambda^{-1}x \\
    \bullet & \quad (A - sI)^{-1}x = (\lambda - s)^{-1}x
\end{align*} \]

Idea:

\[ A \quad \longrightarrow \quad B = (A - sI)^{-1} \]

\[ \lambda_i \quad \longrightarrow \quad \mu_i = (\lambda_i - s)^{-1} \]

Choose the shift \( s \approx \lambda_i \), the closer the better.

\( B \) will then have a very large eigenvalue; power iteration on \( B \).
Inverse iteration algorithm (extra material)

Algorithm:

- Choose starting value $y_0$ and shift $s$
- $LU$–factorize $A - sI$
- Iterate

$$\hat{y}_k = y_k / \|y_k\|_2$$

$$(A - sI)y_{k+1} = \hat{y}_k$$

$$\sigma_k = \hat{y}_k^H y_{k+1}$$

where $\sigma_k \rightarrow 1/(\lambda_i - s) \iff s + 1/\sigma_k \rightarrow \lambda_i$.

**Note** Convergence is linear for a fixed shift, but quadratic if updated by $s_{k+1} := s_k + 1/\sigma_k$.

(symmetric matrices + updated $s$: cubic convergence).