Eigenvalue problem $Ax = \lambda x$

First of all, $Ax = \lambda x$ no explicit solution (neither λ nor x); huge difference from $Ax = b$ for which $x = A^{-1}b$

- Eigenvalues are roots of characteristic polynomial
- For any polynomial p, $\exists$ (infinitely many) matrices whose eigvals are roots of p

Eigenvalue problem $Ax = \lambda x$

First of all, $Ax = \lambda x$ no explicit solution (neither λ nor x); huge difference from $Ax = b$ for which $x = A^{-1}b$

- Eigenvalues are roots of characteristic polynomial
- For any polynomial p, $\exists$ (infinitely many) matrices whose eigvals are roots of p
- Let $p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, $a_i \in \mathbb{C}$. Then $p(\lambda) = 0 \Leftrightarrow \lambda$ eigenvalue of

$$C = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_1 & -a_0 \\ 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & 0 \end{bmatrix} \in \mathbb{C}^{n\times n}$$

Eigenvalue problem $Ax = \lambda x$

First of all, $Ax = \lambda x$ no explicit solution (neither λ nor x); huge difference from $Ax = b$ for which $x = A^{-1}b$

- Eigenvalues are roots of characteristic polynomial
- For any polynomial p, $\exists$ (infinitely many) matrices whose eigvals are roots of p
- So no finite-step algorithm exists for $Ax = \lambda x$

Eigenvalue algorithms are necessarily iterative and approximate

- Same for SVD, as $\sigma_i(A) = \sqrt{\lambda_i(A^T A)}$
- But this doesn't mean they're inaccurate!

Usual goal: compute the Schur decomposition $A = UTU^*$: U unitary, T upper triangular

- For normal matrices $A^*A = AA^*$, automatically diagonalised (T diagonal)
- For nonnormal A, if diagonalisation $A = X\Lambda X^{-1}$ really necessary, done via Sylvester equations but nonorthogonal/unstable (nonexaminable)

Schur decomposition

Let $A \in \mathbb{C}^{n\times n}$ (square arbitrary matrix). Then $\exists$ unitary $U \in \mathbb{C}^{n\times n}$ s.t.

$$A = UTU^*,$$

with T upper triangular.

- $\text{eig}(A) = \text{eig}(T) = \text{diag}(T)$
- T diagonal iff A normal $A^*A = AA^*$

Proof:

Schur decomposition

Let $A \in \mathbb{C}^{n\times n}$ (square arbitrary matrix). Then $\exists$ unitary $U \in \mathbb{C}^{n\times n}$ s.t.

$$A = UTU^*,$$

with T upper triangular.

- $\text{eig}(A) = \text{eig}(T) = \text{diag}(T)$
- T diagonal iff A normal $A^*A = AA^*$

Proof: Let $Av = \lambda_1 v$ and find $U_1 = [v_1, V_\perp]$ unitary. Then

$$AU_1 = U_1 \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix} \Leftrightarrow U_1^* A U_1 = \begin{bmatrix} * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}.$$ Repeat on the lower-right $(n-1)\times(n-1)$ part to get $U_{n-1}^* U_{n-2}^* \dots U_1^* A U_1 U_2 \dots U_{n-1} = T$.

Recap: Matrix decompositions

- SVD $A = U\Sigma V^T$
- Eigenvalue decomposition $A = X\Lambda X^{-1}$
  - Normal: X unitary $X^*X = I$
  - Symmetric: X unitary and Λ real
- Jordan decomposition: $A = XJX^{-1}$, $J = \text{diag}\left(\begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}\right)$
- **Schur decomposition** $A = QTQ^*$: Q orthogonal, T upper triangular
- QR: Q orthonormal, U upper triangular
- LU: L lower triangular, U upper triangular

Red: Orthogonal decompositions, stable computation available

Recap: Matrix decompositions

- SVD $A = U\Sigma V^T$
- Eigenvalue decomposition $A = X\Lambda X^{-1}$
  - Normal: X unitary $X^*X = I$
  - Symmetric: X unitary and Λ real
- Jordan decomposition: $A = XJX^{-1}$, $J = \text{diag}\left(\begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}\right)$
- **Schur decomposition** $A = QTQ^*$: Q orthogonal, T upper triangular
- QR: Q orthonormal, U upper triangular
- LU: L lower triangular, U upper triangular
- QZ for $Ax = \lambda Bx$: (genearlised eigenvalue problem) Q, Z orthogonal s.t. QAZ, QBZ are both upper triangular

Red: Orthogonal decompositions, stable computation available

Power method for $Ax = \lambda x$

$x \in \mathbb{R}^n$:=random vector, $x = Ax$, $x = \frac{x}{\|x\|}$, $\hat{\lambda} = x^T Ax$, repeat

Power method for $Ax = \lambda x$

$x \in \mathbb{R}^n$:=random vector, $x = Ax$, $x = \frac{x}{\|x\|}$, $\hat{\lambda} = x^T Ax$, repeat

- Convergence analysis: suppose A is diagonalisable (generic assumption). We can write $x_0 = \sum_{i=1}^n c_i v_i$, $Av_i = \lambda_i v_i$ with $|\lambda_1| > |\lambda_2| > \cdots$. Then after k iterations,
$$x = C \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k c_i v_i \to C c_1 v_1 \quad \text{as } k \to \infty$$
- Converges geometrically $(\lambda, x) \to (\lambda_1, x_1)$ with **linear rate** $\frac{|\lambda_2|}{|\lambda_1|}$
- What does this imply about $A^k = QR$ as $k \to \infty$? First vector of $Q \to v_1$

Power method for $Ax = \lambda x$

$x \in \mathbb{R}^n$:=random vector, $x = Ax$, $x = \frac{x}{\|x\|}$, $\hat{\lambda} = x^T Ax$, repeat

- Convergence analysis: suppose A is diagonalisable (generic assumption). We can write $x_0 = \sum_{i=1}^{n} c_i v_i$, $Av_i = \lambda_i v_i$ with $|\lambda_1| > |\lambda_2| > \cdots$. Then after k iterations,
$$x = C \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k c_i v_i \to C c_1 v_1 \quad \text{as } k \to \infty$$
- Converges geometrically $(\lambda, x) \to (\lambda_1, x_1)$ with **linear rate** $\frac{|\lambda_2|}{|\lambda_1|}$
- What does this imply about $A^k = QR$ as $k \to \infty$? First vector of $Q \to v_1$

Notes:

- Google pagerank & Markov chain linked to power method
- As we'll see, power method is basis for refined algs (QR algorithm, Krylov methods (Lanczos, Arnoldi,...))

Why compute eigenvalues? Google PageRank

'Importance' of websites via dominant eigenvector of column-stochastic matrix

$$A = \alpha P + (1-\alpha)\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

P: adjacency matrix, $\alpha \in (0,1)$

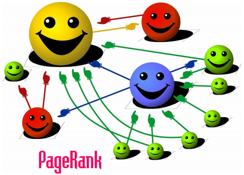

image from wikipedia

Google does (did) a few steps of Power method: with initial guess x_0, $k = 0, 1, \ldots$

1. $x_{k+1} = Ax_k$
2. $x_{k+1} = x_{k+1}/\|x_{k+1}\|_2$, $\quad k \leftarrow k+1$, repeat.

- $x_k \to$ PageRank vector $v_1 : Av_1 = \lambda_1 v_1$

Inverse power method

Inverse (shift-and-invert) power method: $x := (A - \mu I)^{-1}x$, $x = x/\|x\|$

- Converges with improved **linear rate** $\frac{|\lambda_{\sigma(2)} - \mu|}{|\lambda_{\sigma(1)} - \mu|}$ to eigval closest to μ (σ: permutation)

Inverse power method

Inverse (shift-and-invert) power method: $x := (A - \mu I)^{-1}x$, $x = x/\|x\|$

- Converges with improved **linear rate** $\frac{|\lambda_{\sigma(2)}-\mu|}{|\lambda_{\sigma(1)}-\mu|}$ to eigval closest to μ (σ: permutation)
- μ can change adaptively with the iterations. The choice $\mu := x^T Ax$ gives Rayleigh quotient iteration, with **quadratic** convergence $\|Ax^{(k+1)} - \lambda^{(k+1)}x^{(k+1)}\| = O(\|Ax^{(k)} - \lambda^{(k)}x^{(k)}\|^2)$ (cubic if A symmetric)