Numerical Analysis
Chen Xi
Summer, 2025

Jacobi Matrices

1 Jacobi Matrix

A *Jacobi matrix* is a symmetric tridiagonal matrix of the form:

$$T_n = \begin{bmatrix} \gamma_1 & \delta_2 & & & \\ \delta_2 & \gamma_2 & \delta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{n-1} & \gamma_{n-1} & \delta_n \\ & & & \delta_n & \gamma_n \end{bmatrix}. \tag{$*$}$$

More generally, a *Jacobi operator* is a symmetric linear operator on sequences, represented by an infinite tridiagonal matrix. It specifies systems of orthonormal polynomials with respect to a finite, positive Borel measure. The name originates from Jacobi's 1848 theorem stating that every symmetric matrix over a principal ideal domain is congruent to a tridiagonal matrix.

2 Algebraic Properties

Let $\{\varphi_n\}_{n=0}^{\infty}$ be orthonormal polynomials satisfying the three-term recurrence

$$\delta_{n+1}\varphi_n(\lambda) = (\lambda - \gamma_n)\varphi_{n-1}(\lambda) - \delta_n\varphi_{n-2}(\lambda). \tag{2.1}$$

Then the finite Jacobi matrix T_n is as in ($*$).

A simple calculation shows that the characteristic polynomial satisfies:

$$\det(\lambda I - T_n) = (\lambda - \gamma_n)\det(\lambda I - T_{n-1}) - \delta_n^2 \det(\lambda I - T_{n-2}). \tag{2.2}$$

Defining $\psi_n(\lambda) := \det(\lambda I - T_n)$, (2.2) becomes

$$\psi_n(\lambda) = (\lambda - \gamma_n)\psi_{n-1}(\lambda) - \delta_n^2\psi_{n-2}(\lambda). \tag{2.3}$$

This recurrence corresponds to another tridiagonal matrix

$$\hat{T}_n = \begin{bmatrix} \gamma_1 & 1 & & & \\ \delta_2^2 & \gamma_2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{n-1}^2 & \gamma_{n-1} & 1 \\ & & & \delta_n^2 & \gamma_n \end{bmatrix}. \tag{2.4}$$

In fact,

$$\psi_n(\lambda) = \Big(\prod_{j=2}^{n+1} \delta_j\Big)\varphi_n(\lambda),$$

so ψ_n is the monic version of φ_n. Consequently, the eigenvalues of T_n are the roots of both $\psi_n(\lambda)$ and $\varphi_n(\lambda)$.

2.1 Lanczos Algorithm

The *Lanczos algorithm* is the special version of the *Arnoldi algorithm* applied to symmetric matrices, which is used to compute a orthonormal basis of the Krylov subspace $\mathcal{K}_n(A, v) = \operatorname{span}\{v, Av, A^2v, \ldots, A^{n-1}v\}$.

Algorithm 1 Arnoldi Algorithm

Input: Matrix $A \in \mathbb{R}^{m\times m}$, vector $v \in \mathbb{R}^m$, integer n.
Output: Orthonormal basis $\{q_1, q_2, \ldots, q_n\}$ of $\mathcal{K}_n(A, v)$ and upper Hessenberg matrix H_n.
1: Normalize $q_1 = \frac{v}{\|v\|}$.

---

3 Multiplication Matrix

The *multiplication matrix* for Chebyshev polynomials is defined by

$$M_x = \begin{bmatrix} 0 & \frac{1}{2} & & & & \\ 1 & 0 & \frac{1}{2} & & & \\ & \frac{1}{2} & 0 & \frac{1}{2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{1}{2} & 0 & \frac{1}{2} \\ & & & & \frac{1}{2} & 0 \end{bmatrix}, \tag{3.1}$$

which is **not** symmetric and thus not a Jacobi matrix. However, M_x and the Jacobi matrix share eigen-structure: the eigenvalues of M_x are the first-kind Chebyshev points,

$$x_k = \cos\left(\tfrac{(2k-1)\pi}{2n}\right), \quad k = 1, 2, \ldots, n,$$

and the eigen-decomposition is

$$M_x V = V X,$$

with

$$X = \mathrm{diag}(x_1, x_2, \ldots, x_n),$$

$$V = \begin{bmatrix} 1 & & & & \\ & 2 & & & \\ & & 2 & & \\ & & & \ddots & \\ & & & & 2 \end{bmatrix} \begin{bmatrix} T_0(x_1) & T_0(x_2) & \cdots & T_0(x_n) \\ T_1(x_1) & T_1(x_2) & \cdots & T_1(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ T_{n-1}(x_1) & T_{n-1}(x_2) & \cdots & T_{n-1}(x_n) \end{bmatrix}.$$

One verifies that if $f_1(x) = T_0(x) = 1$, then the relations

$$\begin{aligned} f_2(x_j) &= 2x_j f_1(x_j), \\ f_3(x_j) &= 2x_j f_2(x_j) - 2f_1(x_j), \\ f_{k+1}(x_j) &= 2x_j f_k(x_j) - f_{k-1}(x_j), \quad k = 3, \ldots, n-1, \end{aligned}$$

yield $f_{k+1}(x) = 2T_k(x) = 2\cos(k \arccos(x))$ for $k \geq 1$, matching the Chebyshev definition.

Thus the matrix–vector product $b = Vc$ has entries

$$b_k = (2 - \delta_{1k}) \sum_{j=1}^{n} c_j \cos\left(k \tfrac{(2j-1)\pi}{2n}\right), \quad \delta_{1k} = \begin{cases} 1, & k = 1, \\ 0, & k \neq 1, \end{cases}$$

which is a DCT-II computation and can be performed in $\mathcal{O}(n \log n)$ time.