VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY | KATEDRA APLIKOVANÉ MATEMATIKY

Approximation of Karhunen-Loève Decomposition of Isotropic Gaussian Random Fields Using Orthogonal Polynomials and Gaussian Quadratures

Michal Sedlář

May 26, 2025

Goal of thesis

- Rewriting related theory in an understandable way
- Implementing said theory in python
- Experimenting with efficiency

Implementation

Legendre polynomials

Legendre polynomials

Three-term recurrence:

$$\pi_{k+1}(x) = (x - \alpha_k)\,\pi_k(x) - \beta_k \pi_{k-1}(x), \quad k = 0, 1, 2, \ldots,$$

$$\pi_{-1}(x) = 0, \pi_0(x) = 1.$$

Normalized Legendre polynomials

Three-term recurrence:

$$\sqrt{\beta_{k+1}}\tilde{\pi}_{k+1}(x) = (x - \alpha_k)\,\tilde{\pi}_k(x) - \sqrt{\beta_k}\tilde{\pi}_{k-1}(x), \quad k = 0, 1, 2, \ldots,$$

$$\tilde{\pi}_{-1}(x) = 0, \quad \tilde{\pi}_0(x) = 1/\sqrt{\beta_0},$$

Legendre polynomial transformation:

$$\pi_i(x) = \pi_i^L \left(\frac{a+b}{a-b} + \frac{2}{b-a} x \right) \sqrt{\frac{2}{b-a}}.$$

Quadrature transformations

- nodes:

$$\tau_i = \frac{a+b}{2} + \frac{b-a}{2} \tau_i^L$$

- weights:

$$\lambda_i = \frac{b-a}{2} \lambda_i^L$$

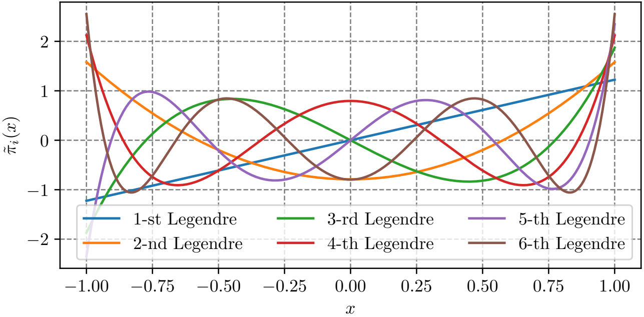

Figure 1: Normalized Legendre polynomials

Quadrature rules

Quadrature

Jacobi matrix represented vectors

eigenpairs from *scipy.linalg.eigh_tridiagonal*

Eigenvalues are the quadrature nodes

The quadrature weights λ_i:

$$\lambda_i = \beta_0 \boldsymbol{v}_{i,1}^2.$$

Computation time

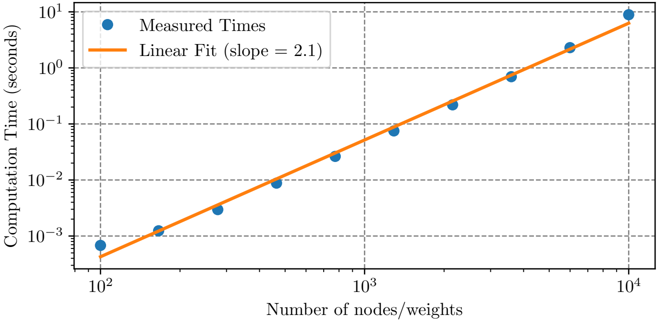

Figure 2: Log-Log Plot of Computation Time vs. Jacobi Size n

Karhunen-Loève expansion

$$X(t,\omega) = \sum_{j=1}^{\infty} \gamma_j(\omega)\sqrt{\lambda_j}u_j(t)$$

$$\gamma_j(\omega) := \langle X(t,\omega), u_j(t)\rangle_{L^2(\mathcal{T})}$$

Galerkin method

$$A\overline{\boldsymbol{u}}_i = \lambda_{i,N} W \overline{\boldsymbol{u}}_i,$$

$$A_{k,j} = \int_{\mathcal{T}} \int_{\mathcal{T}} \psi_k(\boldsymbol{y})\, c(\boldsymbol{x}, \boldsymbol{y})\, \psi_j(\boldsymbol{x})\, d\boldsymbol{x} d\boldsymbol{y},$$

$$W_{k,j} = \int_{\mathcal{T}} \psi_k(\boldsymbol{x})\, \psi_j(\boldsymbol{x})\, d\boldsymbol{x}.$$

A is calculated using *numpy.einsum*

$A_{i,j}$ calculation error

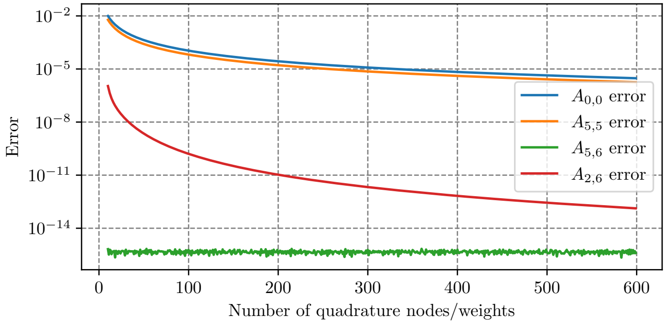

Figure 3: Error of entries $A_{i,j}$

Thank you for your attention

Supervisor question

V grafu 4.1 sledujete pomalou konvergenci k přesné hodnotě integrálu při exponenciální kovarianci. Lze tento integrál efektivněji vyčíslit s ohledem na nespojitost derivace na přímce $x = y$? Jaký byste zvolil postup, aby bylo možné dosáhnout požadované přesnosti s výrazně menším počtem kvadraturních bodů?

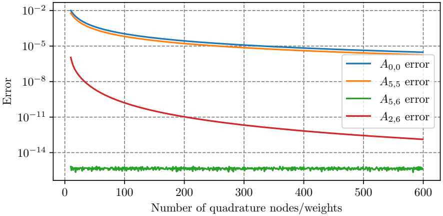

Figure 4: Error of entries $A_{i,j}$

Answer

- Rozdělíme na část pod $y = x$ a část nad $y = x$.
- 2 ekvivalentní trojúhelníky
- Substituce na čtverec
- Integrál přes čtverec násobený dvěma

Opponent questions

Odkud v důkazu věty 15 plyne, že $\langle x\pi_k, \pi_{k-1}\rangle = \langle \pi_k, \pi_k\rangle$?

Answer

$$\langle x\pi_k, \pi_{k-1}\rangle = \langle \pi_k, x\pi_{k-1}\rangle$$

π_{k-1} je monický $\implies x\pi_{k-1}$ je monický

$$\begin{aligned} \exists p_{k-1} \in P_{k-1} : x\pi_{k-1} &= x^k + p_{k-1} \\ \exists q_{k-1} \in P_{k-1} : \pi_k &= x^k + q_{k-1} \end{aligned}$$

$$x\pi_{k-1} - \pi_k = p_{k-1} - q_{k-1} = r_{k-1} \in P_{k-1}$$

Answer

$$x\pi_{k-1} = \pi_k + x\pi_{k-1} - \pi_k = \pi_k + r_{k-1} = \pi_k + \sum_{i=0}^{k-1} c_i \pi_i, \quad r_{k-1} \in P_{k-1}$$

$$\langle \pi_k, x\pi_{k-1} \rangle = \langle \pi_k, \pi_k + \sum_{i=0}^{k-1} c_i \pi_i \rangle = \langle \pi_k, \pi_k \rangle + \sum_{i=0}^{k-1} c_i \langle \pi_k, \pi_i \rangle = \langle \pi_k, \pi_k \rangle$$

Opponent questions

Na straně 32 píšete, že efektivita Gaussovské kvadratury závisí na hladkosti integrované funkce. Z čeho toto tvrzení vychází?

Answer

Vychází z Remark 25 (The n-point Gauss quadrature rule integrates polynomials of degree up to $2n - 1$ exactly.)

Spojité funkce se dají aproximovat polynomem. Efektivita Gaussové kvadratury poté závisí na nepřesnosti aproximace polynomem.

Opponent questions

Jak by Vaše implementace KL rozkladu fungovala ve více než dvou prostorových dimenzích?

Answer

Implementace KL by fungovala podobným způsobem.

Bázové funkce $\psi(\boldsymbol{x})$ by byly násobky orthonormálních polynomů $(\psi_{i,j,k,l}(\boldsymbol{x}) = \tilde{\pi}_i(x_1)\tilde{\pi}_j(x_2)\tilde{\pi}_k(x_3)\tilde{\pi}_l(x_4))$

Tensor A by měl více dimenzí

Výpočetní náročnost by stoupla

Opponent questions

Bylo by možné provést KL rozklad i na jiných doménách než kartézských součinech intervalů?

Answer

Ano, ale ne s mojí implementací.