Numerical Methods 4. Interpolation problems

Interpolations problems

Univariate interpolation methods

Scattered data interpolation

Interpolation problems

For given locations $x_1, x_2, \ldots, x_N \in \mathbf{R}^n$ and corresponding values $u_1, u_2, \ldots, u_N \in \mathbf{R}^m$, find a function $u : \mathbf{R}^n \to \mathbf{R}^m$ such that $u(x_k) = u_k \quad (k = 1,2,\ldots,N)$.

Fields of application:

- curve fitting;
- surface fitting;
- completion of data;
- data definition for computational models etc.

If $n, m > 1$ (**vectorial** interpolation problems), then the corresponding values defined at the interpolation points $x_1, x_2, \ldots, x_N \in \mathbf{R}^n$ are vectors: in this case, some additional conditions are prescribed for the interpolation vector field, e.g. divergence-free and/or rotation-free property.

Numerical Methods 4. Interpolation problems

Interpolation problems

Univariate interpolation methods

Scattered data interpolation

The Lagrangian interpolation

Given: $x_1, x_2, \ldots, x_N \in \mathbf{R}$ locations (interpolation points) and the corresponding values $u_1, u_2, \ldots, u_N \in \mathbf{R}$. Find: **a polynomial P_{N-1} of degree at most $(N-1)$**, for which $P_{N-1}(x_k) = u_k \quad (k = 1,2,\ldots,N)$

Lagrange base polynomials:

$$l_j(x) := \prod_{r \neq j} \frac{x - x_r}{x_j - x_r} = \frac{(x-x_1)(x-x_2)\ldots(x-x_{j-1})(x-x_{j+1})\ldots(x-x_N)}{(x_j-x_1)(x_j-x_2)\ldots(x_j-x_{j-1})(x_j-x_{j+1})\ldots(x_j-x_N)}$$

The interpolation polynomial:

$$P_{N-1}(x) = \sum_{j=1}^{N} u_j \cdot l_j(x)$$

The Lagrange interpolation polynomial is unique.

If there existed two interpolation polynomials P_{N-1} and Q_{N-1} (of degree at most $(N-1)$), then their difference would have N roots, therefore $P_{N-1} - Q_{N-1} \equiv 0$.

The Lagrangian interpolation

Another way to compute the coefficients of the Lagrangian interpolation polynomial:

Suppose that the interpolation polynomial has the form: $P_{N-1}(x) = \sum_{j=1}^{N} a_j \cdot x^{j-1}$.

Then, from the interpolation conditions:

$$P_{N-1}(x_k) = \sum_{j=1}^{N} a_j \cdot x_k^{j-1} = u_k \qquad (k = 1,2,...,N)$$

This is a system of equations with *N* unknowns. The entries of the matrix are: $A_{kj} = x_k^{j-1}$. The matrix is regular (since it is a **Vandermonde matrix**), therefore the Lagrangian interpolation polynomial exists and is unique.

The Hermitian interpolation

Given: $x_1, x_2, \ldots, x_N \in \mathbf{R}$ locations (interpolation points) and the corresponding values $u_k^{(i_k)} \in \mathbf{R} \quad (i_k = 0,1,\ldots,m_k - 1)$. Denote by $m := m_1 + m_2 + \ldots + m_N$. Find: **a polynomial H_{m-1} of degree at most (m – 1)**, for which:

$$H_{m-1}^{(j)}(x_k) = u_k^{(j)} \quad (j = 0,1,\ldots,m_k - 1, \quad k = 1,2,\ldots,N)$$

The Hermitian interpolation polynomial exists and is unique.

The coefficients of the Hermite interpolation polynomial $H_{m-1}(x) = \sum_{i=1}^{m} a_j \cdot x^{i-1}$ can be calculated by solving the following system of equations:

$$H_{m-1}^{(j)}(x_k) = \sum_{i=1}^{m} a_j \cdot \frac{d^j (x^{i-1})}{dx^j}\Big|_{x=x_k} = u_k^{(j)} \qquad (j = 0,1,\ldots,m_k - 1, \quad k = 1,2,\ldots,N)$$

Special cases:

- $m_k = 1$ for each index k $\Rightarrow$ Lagrangian interpolation
- $N = 1, \quad m_1 = m$ $\Rightarrow$ Taylor polynomial

Two-point cubic Hermitian interpolation

Given: $x_0, x_1 \in \mathbf{R}$ locations (interpolation points) and the corresponding values u_0, u_1, u_0', u_1'.

Find: a cubic polynomial $H(x) = A + B \cdot \left(\frac{x - x_0}{h} \right) + C \cdot \left(\frac{x - x_0}{h} \right)^2 + D \cdot \left(\frac{x - x_0}{h} \right)^3$ (where h denotes the distance $h := x_1 - x_0$) , such that:

$$
\begin{aligned}
H(x_0) &= A &&= u_0 \\
H(x_1) &= A + B + C + D &&= u_1 \\
h \cdot H'(x_0) &= B &&= h \cdot u_0' \\
h \cdot H'(x_1) &= B + 2C + 3D &&= h \cdot u_1'
\end{aligned}
$$

The solution of the system:

$$
\begin{aligned}
A &= u_0 \\
B &= h \cdot u_0' \\
C &= -3u_0 + 3u_1 - 2h \cdot u_0' - h \cdot u_1' \\
D &= 2u_0 - 2u_1 + h \cdot u_0' + h \cdot u_1'
\end{aligned}
$$

Piecewise cubic Hermitian interpolation

Given: $x_0, x_1, ..., x_N \in \mathbf{R}$ locations (interpolation points) and the corresponding values $u_0, u_1, .. u_N, u'_0, u'_1, ..., u'_N$.

On each subinterval $[x_{k-1}, x_k]$, perform a cubic Hermitian interpolation based on the values $u_{k-1}, u'_{k-1}, u_k, u'_k$. The polynomials defined on the different subintervals are connected at the interpolation points always in a C^1**-continuous way**, i.e. the first derivative is continuous at the inner interpolation points.

Remark: The values $u'_0, u'_1, ..., u'_N \in \mathbf{R}$ are unknown in general.

A possible solution: define the derivatives as follows:

$$u'_0 := 0, \quad u'_k := \frac{u_{k+1} - u_{k-1}}{x_{k+1} - x_{k-1}} \quad (k = 1, ..., N-1), \quad u'_N := 0.$$ But there is a better technique!

Cubic spline interpolation

Given: $x_0, x_1, \ldots, x_N \in \mathbf{R}$ locations (interpolation points) and the corresponding values $u_0, u_1, \ldots, u_N \in \mathbf{R}$.

Find: a piecewise cubic polynomial S such that

$$S(x_k) = u_k \quad (k = 0,1,\ldots,N)$$

and the polynomials defined on the different subintervals are connected at the interpolation points in a C^2**-continuous way**, i.e. even the second derivative is continuous at the inner interpolation points.

Idea: with properly defined values $u_0', u_1', \ldots, u_N' \in \mathbf{R}$, perform a piecewise cubic Hermitian interpolation.

On the subinterval $[x_{k-1}, x_k]$ (denote by $h_{k-1} := x_k - x_{k-1}$):

$$H_{k-1}(x) = a_0 + a_1 \frac{x - x_{k-1}}{h_{k-1}} + a_2 \left(\frac{x - x_{k-1}}{h_{k-1}} \right)^2 + a_3 \left(\frac{x - x_{k-1}}{h_{k-1}} \right)^3$$

On the subinterval $[x_k, x_{k+1}]$ (denote by $h_k := x_{k+1} - x_k$):

$$H_k(x) = a_0 + a_1 \frac{x - x_k}{h_k} + a_2 \left(\frac{x - x_k}{h_k} \right)^2 + a_3 \left(\frac{x - x_k}{h_k} \right)^3$$

Cubic spline interpolation

From the condition $H''_{k-1}(x_k) = H''_k(x_k)$, after some algebraic manipulations we obtain:

$$\frac{1}{h_{k-1}}u'_{k-1} + \left(\frac{2}{h_{k-1}} + \frac{2}{h_k}\right)u'_k + \frac{1}{h_k}u'_{k+1} = -\frac{3}{h_{k-1}^2}u_{k-1} + \left(\frac{3}{h_{k-1}^2} - \frac{3}{h_k^2}\right)u_k + \frac{3}{h_k^2}u_{k+1} \qquad (k = 1,\ldots,N-1)$$

This is a 3-diagonal system for the a priori unknown values $u'_1,\ldots,u'_{N-1}$.

In case of *equidistant* interpolation points, where $h_0 = h_1 = \cdots = h_{N-1} = h$:

$$u'_{k-1} + 4u'_k + u'_{k+1} = -\frac{3}{h}u_{k-1} + \frac{3}{h}u_{k+1} \qquad (k = 1,\ldots,N-1)$$

where the first and last values u'_0, u'_N can be defined arbitrarily.

The cubic spline minimizes the functional

$$F(u) := \int_{x_0}^{x_N} |u''(x)|^2 dx$$

among all functions that satisfy the interpolation conditions and the boundary conditions $u'(x_0) = u'_0\,, \quad u'(x_N) = u'_N.$

Numerical Methods 4. Interpolation problems

Interpolation problems

Univariate interpolation methods

Scattered data interpolation

Multivariate interpolation

Given: $x_1, x_2, ..., x_N \in \mathbf{R}^2$ (locations) and $u_1, u_2, ..., u_N \in \mathbf{R}$ (corresponding values).
We look for a function u (as smooth as possible), for which $u(x_k) = u_k \quad (k = 1,2,...,N)$ is valid.

If the interpolation points are located on a 2D rectangular grid: $(x_k^{(1)}, x_j^{(2)}) \in \mathbf{R}^2$ and the corresponding values are $u_{k,j} \quad (k = 1,2,...,N,\ j = 1,2,...,M)$, then a **bivariate Lagrangian interpolation** can be applied:

The Lagrange base polynomials:

$$l_{k,j}(x) := l_{k,j}(x^{(1)}, x^{(2)}) := \prod_{\substack{p \neq k \\ q \neq j}} \frac{x^{(1)} - x_p^{(1)}}{x_k^{(1)} - x_p^{(1)}} \cdot \frac{x^{(2)} - x_q^{(2)}}{x_j^{(2)} - x_q^{(2)}}$$

The interpolation polynomial:

$$P(x) = \sum_{k=1}^{N} \sum_{j=1}^{M} u_{k,j} \cdot l_{k,j}(x)$$

Shepard's method

Given: $x_1, x_2, ..., x_N \in \mathbf{R}^2$ (locations) and $u_1, u_2, ..., u_N \in \mathbf{R}$ (corresponding values).

$$u(x) := \frac{\sum_{j=1}^{N} u_j w_j(x)}{\sum_{j=1}^{N} w_j(x)}, \qquad w_j(x) := \frac{1}{\| x - x_j \|^2}$$

Then $\lim u(x) = u_k$, whenever $x \to x_k$ $(k = 1,2, \ldots, N)$, i.e. the interpolation conditions are fulfilled (in the sense of the limit value)

Numerical features:

- Numerically stable; no solution of a system of equations is required;
- Moderate computational cost ($O(N)$ algebraic operations at each point of evaluation);
- Moderate accuracy.

However:

Both partial derivatives of the Shepard interpolation function vanish at each interpolation point.

The method of Radial Basis Functions

Given: $x_1, x_2, ..., x_N \in \mathbf{R}^2$ (locations) and $u_1, u_2, ..., u_N \in \mathbf{R}$ (corresponding values).

$$u(x) := \sum_{j=1}^{N} \alpha_j \Phi_j (x - x_j)$$

where $\Phi_1, ..., \Phi_N$ are predefined, spherically symmetric functions (radial basis functions).
The a priori unknown coefficients $\alpha_1, \alpha_2, ..., \alpha_N$ can be computed by solving the **interpolation equations**:

$$\sum_{j=1}^{N} \alpha_j \Phi_j (x_k - x_j) = u_k \qquad (k = 1,2,...,N)$$

Numerical features:

- Very good accuracy;
- Solution of a system of equations is required;
- Large, dense and ill-conditioned matrices;
- High computational cost ($O(N^3)$ algebraic operations).

The method of Radial Basis Functions

Some special cases:

Multiquadrics, MQ:

$$\Phi_j(x) := \sqrt{\| x \|^2 + c_j^2}$$

($c_1, c_2, \ldots, c_N \in \mathbf{R}$ are predefined **scaling parameters**)

Inverse multiquadrics, iMQ:

$$\Phi_j(x) := \frac{1}{\sqrt{||x||^2 + c_j^2}}$$

($c_1, c_2, \ldots, c_N \in \mathbf{R}$ are predefined **scaling parameters**)

Thin plate splines, TPS:

$$\Phi_j(x) := \| x \|^2 \log \| x \|$$

(no scaling parameters are required)

Gauss functions:

$$\Phi_j(x) := e^{-c_j^2 \cdot \|x\|^2}$$

($c_1, c_2, \ldots, c_N \in \mathbf{R}$ are predefined scaling parameters)