Interpolation Methods

Mohamed Aburakhis and Mohammed Almatrafi.

Adviser: Muhammad Usman, Ph.D.

Objectives

An interpolation survey is presented in this work. Polynomial interpolation, cubic splines interpolation, Akima cubic spline interpolation, Sinc function interpolation, and Radial Basis Functions (RBFs) interpolation are implemented using MATLAB. The derivation and mathematical equations are presented. Then all methods are applied to one example for the sake of comparison, a randomly chosen sample points are taken according to Sinc function interpolation method due to its exclusiveness.

Introduction

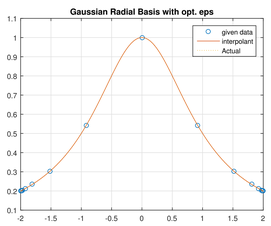
The general concept of the interpolation can be defined as: Given a data set $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, x_n)\}$ of (n+1) points, we would like to find an interpolant $p(x)$ that passes through the data. This can be used to approximate the function values within the range of the data.

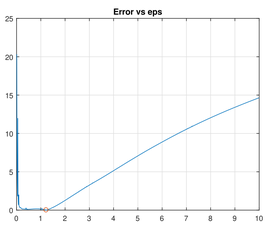
In this poster, we would like to compare different type of interpolations in terms of absolute error and time complexity. The general formula of interpolant can be defined as

$$P_n = \sum_{i=0}^{n} c_i \, \phi_i(x), \tag{1}$$

where $\phi_i(x)$ is called the basis functions. These basis functions will be defined based on the interpolation type.

Results





Monomial

This method represents the interpolant p as a linear combination of monomials.

- Disadvantage: computing of inverse of the coefficient matrix is numerically unstable because as number of data points increases the matrix becomes more ill conditioned.

$$P_n = \sum_{i=0}^{n} = c_i \, x^i \tag{2}$$

where $c = [c_0, c_1, \ldots, c_n]^\top$ can be found

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix}. \tag{3}$$

Cubic Splines

The cubic splines are used because for higher order polynomial interpolant the oscillations are quite high. For the given data $(x_0, f_0(x_0)), (x_1, f_1(x_1)), \ldots, (x_n, f_n(x_n))$, the cubic spline is

$$S(x) = S_i(x) \quad \text{on} \quad x_1 \le x \le x_{i+1} \quad 0 \le i \le n-1. \tag{4}$$

and

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i. \tag{5}$$

where

$$a_1 = \frac{z_{i+1} - z_i}{6h}, \; b_i = \frac{z_i}{2}, \; c_i = \frac{f_{i+1} - y_i}{h} - h\frac{2z_i + z_{i+1}}{6},$$

$d_i = f_i$. and

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} 4 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & 4 \end{bmatrix}^{-1} \begin{bmatrix} f_2 - 2f_1 + y_0 \\ f_3 - 2f_2 + y_1 \\ \vdots \\ f_n - 2f_{n-1} + y_{n-2} \end{bmatrix} \tag{6}$$

Akima cubic spline interpolation

This technique was first introduced by Hiroshi Akima in the Journal of the Association for Computing Machinery in 1970. It is a cubic spline with different conditions at the data points.

$$t_i = \frac{|m_{i+1} - m_i| m_{i-1} + |m_{i-1} - m_{i-2}| m_i}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|} \tag{7}$$

where m_i is given by:

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{8}$$

$$t_i = \frac{m_i - m_{i-1}}{2} \tag{9}$$

$$d_i = y_i \tag{10}$$

$$c_i = t_i \tag{11}$$

$$b_i = \frac{3m_i - 2t_i - t_{i+1}}{x_{i+1} - x_i} \tag{12}$$

$$a_i = \frac{-2m_i + t_i + t_{i+1}}{(x_{i+1} - x_i)^2} \tag{13}$$

Lagrange Interpolation

The interpolating polynomial can be written as a linear system of equations which is better conditioned and much easier to solve.

- It overcomes the ill-conditioning of coefficient matrix

These polynomial are denoted by $l_i(x)$, where

$$l_i = \prod_{\substack{j=0 \\ j \ne i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}. \tag{14}$$

$$P_n = \sum_{i=0}^{n} y_i l_i(x), \tag{15}$$

Newton Interpolation

The coefficients of Newton interpolation are computed using divided differences.

$$p(x) = \sum_{i=0}^{n} f[x_1, x_2, \ldots, x_i] \prod_{j=1}^{i} (x - x_{j-1}), \tag{16}$$

Where the coefficients for Newton interpolant of degree n can be computed through divided differences as following:

$$f[x_i] = f(x_i) \qquad i = 0, 1, \ldots, n \tag{17}$$

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{18}$$

$$\vdots$$

$$f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0} \tag{19}$$

Radial basis functions interpolation

RBF's are circular functions centered at certain points. RBFs interpolant is a linear combination of a basis function. The general equation for the RBFs method can be obtained by using basis function $\phi(\|x - x_i\|)$ in where $\|x - x_i\|$ is 2-norm of the distance between the x and the points x_i The weights of the basis function c_i can be computed as follows

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{21} & \ldots & \phi_{N1} \\ \phi_{12} & \phi_{22} & \ldots & \phi_{N2} \\ \vdots & & & \\ \phi_{1N} & \phi_{2N} & \ldots & \phi_{NN} \end{bmatrix}^{-1} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix} \tag{20}$$

Contact Information

- musman1@udayton.edu
- aburakhism1@udyaton.edu
- almatrafim2@udayton.edu

Sinc Interpolation

An exponential convergence rate of the residual error entailed is expected with Sinc Interpolation. The Sinc interpolation uses the logarithmic transformation which is invoked as:

$$Q(x) = \log\left(\frac{x - a}{b - x}\right) \tag{21}$$

The Sinc points can be defined as $x_k = \frac{a + be^{kh}}{1 + e^{kh}}$, where $h = \sqrt{\frac{\pi d}{\beta N}}$, where $d = \pi$, $\beta = 1$ and N denotes the number of Sinc points. The Sinc interpolant can be obtained above equations as following:

$$P(x) = \sum_{k=0-N}^{N} = C_k \, S[Q(x), kh] \tag{22}$$

where the Sinc function $S[Q(x), kh]$ is defined as following:

$$S[Q(x), kh] = \frac{\sin(\pi[Q(x) - kh]/h)}{\pi[Q(x) - kh]/h} \tag{23}$$

Moreover, the Sinc coefficients, C_k can be computed through the following equation when the Sinc function is approximated at the nodal points $[I]_{(2N+1)(2N+1)}[C_k]_{(2N+1)(1)} = [f(x_k)]_{(2N+1)(1)}$. where I is the identity matrix and $f(x_k)$ is the given function evaluated at Sinc points. Thus, C_k cab be obtained as:

$$[C_k]_{(2N+1)(1)} = [f(x_k)]_{(2N+1)(1)} \tag{24}$$

Results