Cubic spline interpolation complexity

Zicong Jiang

Contents

1. With pre-calculated coefficients

Once the coefficients are calculated for input, we can do inference for different coordinates with a complexity: The querying in bicubic surface based on this expression,

$$s(z,t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ z \\ z^2 \\ z^3 \end{bmatrix}.$$

For t^2, t^3, z^2, z^3, we need 6 times real multiplication per coordinate. Then the matrix multiplication needs 20 times multiplication. In order to solve the coefficients matrix, we can calculate

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \tag{1}$$

for the querying surface. In order to solve the problem, we need $f(0,0), f(0,1), f(1,0), f(1,1)$, which are known (input data). So we need to get $f_x(0,0), \ldots, f_y(0,0), \ldots, f_{xy}(0,0), \ldots, f_{yx}(0,0)$, which can be obtained from pre-calculated coefficients from natural cubic spline (1D-interpolation calculated for z,t directions).

In natural cubic spline, we use m_i to represent all coefficients $\{a_i, b_i, c_i, d_i\}$, which related to the second order derivative of cubic polynomial $m_i = f''_i(t_i) = 2c_i$ in corresponding piece i. The first order derivative can also be calculated from m_i, with $f'_i(t_i) = b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1} = f'_i(t_{i+1})$, where $h_i = t_{i+1} - t_i$ is the piece step size. We can assume the implementation is based on b_{i+1}, so we don't need extra multiplication for getting derivatives once we pre-calculate these coefficients.

The total number of real multiplications required by PIDT per coordinate is approximately $\mathbf{2 \times 26 = 52}$, accounting for both real and imaginary components. In contrast, PINO must process the input coordinates through at least the trunk network to obtain location embeddings, requiring $\mathbf{2Q + (J-1)Q^2 + Qq \approx 1.2 \times 10^5}$ multiplications for embedding computation and an additional $\mathbf{2q = 400}$ multiplications for merging embeddings to predict the complex output.

2. Calculation of all coefficients

We can calculate coefficients for the interpolated surface using the (1), which requires $4 \times 4 \times 4 \times 2$ multiplications for coefficients in one surface, and for the whole space we have $(M-1)(N-1)$ surfaces, and $2(M-1)(N-1)$ for real and imaginary, which leads to $\mathbf{256(M-1)(N-1)}$ real multiplications. However, we need to use a natural cubic spline to get first and second order derivatives for (1).

The natural cubic spline is based on building and solving a tridiagonal matrix. We build piece-wise cubic polynomials $S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$, for $i = 1, \ldots, n$. In general, they define the second-

order derivative of polynomials $S_i''(x_i) = 2c_i = m_i$, then each coefficients can be represented by m_i, as

$$\begin{aligned} a_i &= y_i \\ b_i &= \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{2}m_i - \frac{h_i}{6}(m_{i+1}-m_i) \\ c_i &= \frac{m_i}{2} \\ d_i &= \frac{m_{i+1}-m_i}{6h_i} \end{aligned} \quad , \tag{2}$$

where $h_i = x_{i+1} - x_i$. Then we can write piece-wise equations into $h_i m_i + 2(h_i + h_{i+1})m_{i+1} + h_{i+1}m_{i+2} = 6\left[\frac{y_{i+2}-y_{i+1}}{h_{i+1}} - \frac{y_{i+1}-y_i}{h_i}\right]$. Which is a tridiagonal system $a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$ (Wikipedia), can be expressed as

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix} \tag{3}$$

with natural boundary condition $m_0 = 0, m_n = 0$. The system can be solved by Thomas algorithm. We can rewrite c_i, d_i into new coefficients c_i^*, d_i^* as shown in Wikipedia. Then the system in (3) can be expressed as

$$\begin{bmatrix} 1 & c_1^* & 0 & 0 & \dots & 0 \\ 0 & 1 & c_2^* & 0 & \dots & 0 \\ 0 & 0 & 1 & c_3^* & 0 & 0 \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & c_{n-1}^* \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \cdot \\ \cdot \\ \cdot \\ m_n \end{bmatrix} = \begin{bmatrix} d_1^* \\ d_2^* \\ d_3^* \\ \cdot \\ \cdot \\ \cdot \\ d_n^* \end{bmatrix} . \tag{4}$$

The m_i can be easily solved, starting from the last row, $m_n = d_n^*, \quad m_i = d_i^* - c_i^* m_{i+1}, \quad i = n-1, n-2, \dots, 2, 1$. After getting all m, we can calculate the coefficients based on the inverse of (3). If we use the 1st-order derivative as input value to another dimension, do the same calculation, we can get derivatives like S_{xy} and S_{yx} for bicubic spline.

The number of real multiplications for (3) is 7, so for all coefficients we have $7n$ multiplications. Then for transferring from (3) to (5), we need $2+5(n-1)$ multiplications. In order to solve (5), we need $n-1$ multiplications. Therefore, we need $7n+2+5(n-1)+n-1 = 13n-4$ multiplications one time, leads to $4(13n-4) = 52n-16$ multiplications for real and imag for two dimension.

If we combine all values, the number of real multiplications is $\mathbf{52N - 16 + 256(M-1)(N-1)}$. Which is around 6 times faster than $\text{PINO}_{\text{small}}$-32.