Exercise Report-2

For

INF 4140/9140 -Numerical Analysis

Hao, Zhao

Department of Geoscience & Department of Informatics, University of Oslo

March14, 2016

Exercise 4.2.5:

4.2.5 Derive the Lagrange interpolation formula. Show how it can be rewritten in barycentric form. When is the latter form more efficient to use?

Answer:

The lagrange's interpolation can be expressed as the following expression.

$$p(x) = \sum_{j=1}^{n} f(x_j) \prod_{\substack{i=1 \\ i \neq j}}^{n} \frac{(x - x_i)}{(x_j - x_i)}.$$

Where the lagrange polynomial is expressed as:

$$\ell_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{n} \frac{(x - x_i)}{(x_j - x_i)}, \quad j = 1 : n,$$

This form is not efficient for the practical computation, since the Lagrange polynomial needs to be recomputed for each new x location. The total cost for this Lagrange interpolation is 'O(n^2). To solve this problem, the Barycentric from of Lagrange interpolation is derived as following:

Rewrite the Lagrange polynomial expression by taking out the common factor $\phi_n(x)$ and introduce the support coefficient w_j :

$$\ell_j(x) = \Phi_n(x) \frac{w_j}{x - x_j}.$$

Where

$$w_j = \frac{1}{\prod_{\substack{i=1 \\ i \neq j}} (x_j - x_i)}, \quad j = 1 : n. \qquad \Phi_n(x) = \prod_{i=1}^{n} (x - x_i).$$

So the Lagrang's interpolation can be rewritten as:

$$p(x) = \Phi_n(x) \sum_{j=1}^{n} \frac{w_j}{x - x_j} f(x_j).$$

Here the w_j is only dependent on the given x location, and can be pre-computed in 2(n-1) loops for the given n locations. And the total cost of Lagrange interpolation will be reduced to 'O(n).

Also the above modified Lagrang interpolations expression can be further modified to below expression, if we assume the interpolation f(x) ≡1.

Then here we have

$$1 = \Phi_n(x) \sum_{j=1}^{n} \frac{w_j}{x - x_j}.$$

And based on this, the Lagrange interpolation can be rewritten as:

$$p(x) = \frac{\displaystyle\sum_{j=1}^{n} \frac{w_j}{x - x_j} f(x_j)}{\displaystyle\sum_{i=1}^{n} \frac{w_j}{x - x_j}} \quad \text{if } x \neq x_j, \quad j = 1 : n,$$

This is the Barycentric form of Lagrange's interpolation.

This Barycentirc expression of Lagrang's interpolation is more efficient than the original Lgarang's interpolation in computation, which also has good numerical stability than the original form.

Exercise 4.3.2:

4.3.2 (a) Write down the confluent Vandermonde matrix for the Lagrange–Hermite cubic interpolation problem.

(b) Express the divided difference $[x_0, x_0, x_1, x_1]f$ in terms of f_0, f

---

Answer:

(a) For the Lagrange–Hermite cubic interpolation problem

Assuming the function f(x) can be fitted by the below cubic polynomial

$P(x) = C_0 + C_1 * x + C_2 * X^2 + C_3 * X^3$

Thus for the the Hermite interpolation for this case is equivalent to find a polynomial P(x) that interpolate the function f and it's corresponding derivatives f\` at the given locations. This can be expressed as a linear system as $V^T.c = f$, where $f = (f(x1), f'(x1), f''(x1), f(x2), f'(x2), f''(x2) \ldots\ldots)^T$, and the confluent Vandermonde matrix V is expressed as following: (As example here, I assume f(x) is second order differentiable on location x1 and x2,)

$$V = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & \\ x_1 & 1 & 0 & x_2 & 1 & 0 & \\ {x_1}^2 & 2x_1 & 2 & {x_2}^2 & 2x_2 & 2 & \cdots\cdots \\ {x_1}^3 & 3{x_1}^2 & 6x_1 & {x_2}^3 & 3{x_2}^2 & 6x_2 & \end{pmatrix}$$

(b) Express the divided difference [x0, x0, x1, x1]f in terms of f0, f

The divided difference of [x0, x0, x1, x1]f can be expressed as below expression:

$$[x_0, x_0, x_1, x_1]f = \frac{\dfrac{f'(x_1) - \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_1 - x_0} - \dfrac{\dfrac{f(x_1) - f(x_0)}{x_1 - x_0} - f'(x_0)}{x_1 - x_0}}{x_1 - x_0}$$

Problems and Computer Exercises

(The corresponding matlab code files are provided for reference)

Exercise 4.3.9

4.3.9(a) Show the bilinear interpolation formula
$p(x0 +ph, y0 +qk) = (1-p)(1-q)f0,0 +p(1-q)f1,0 +(1-p)qf0,1 +pqf1,1(4.3.66)$ with error bound..

(b) Compute by bilinear interpolation f (0.5, 0.25) when
f (0, 0) = 1, f(1, 0) = 2, f(0, 1) = 3, f(1, 1) = 5.

Plots Display:

(a)
Sorry, I haven't solved this question yet ☹

Remarks:

(b)
Derived bilinearal Interpolation function f(x,y) = 1 + 1 * x + 2 * y + 1 * xy
For x = 0.5,y=0.25 The bilinear interpolated value is: 2.125000

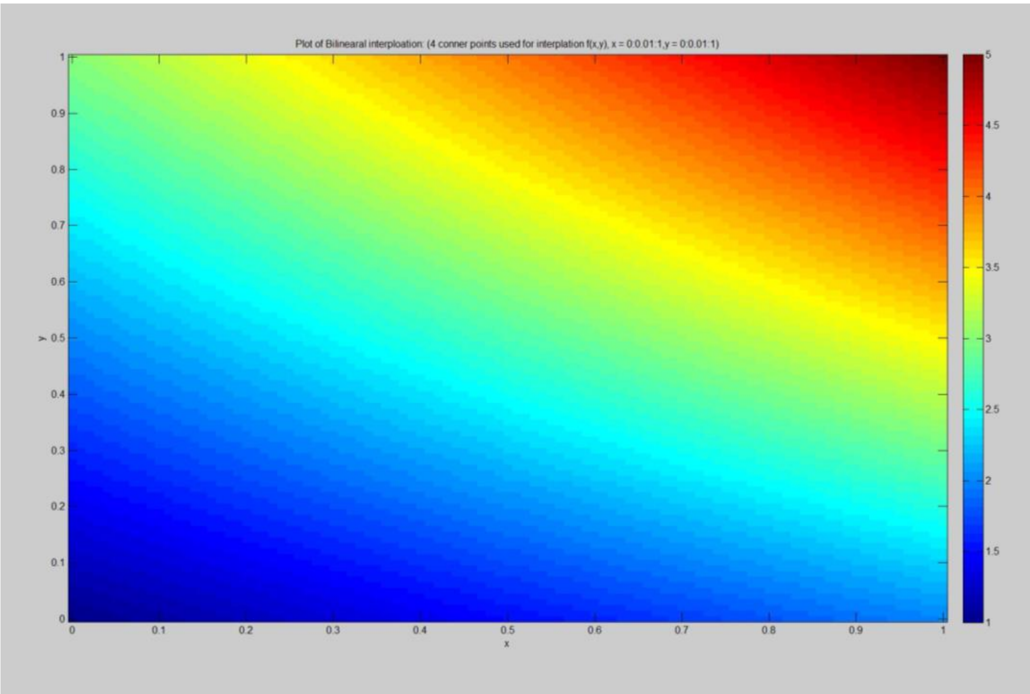

Figure1. Plot of bilinear interpolation

To illustrate the bilinear interpolation, the four corner points are used to interpolate the all the surface points (x = 0:0.01:1, and y = 0:0.01:1)

Exercise 4.1.1 (a-2):

4.3.912 (Bulirsch and Rutishauser (1968))

(a) The function cot x has a singularity at $x = 0$. Use values of cot x for $x = 1^\circ$, $2^\circ, \ldots, 5^\circ$, and rational interpolation of order (2,2) to determine an approximate value of cot x for $x = 2.5^\circ$, and its error.

(b) Use polynomial interpolation for the same problem. Compare the result with that in (a).

Plots Display:

Remarks:

To derive the rational interpolation function, the q1 = 1 is inserted to solve the 5 parameters linear system. The derived equation and test values are presented in the end.

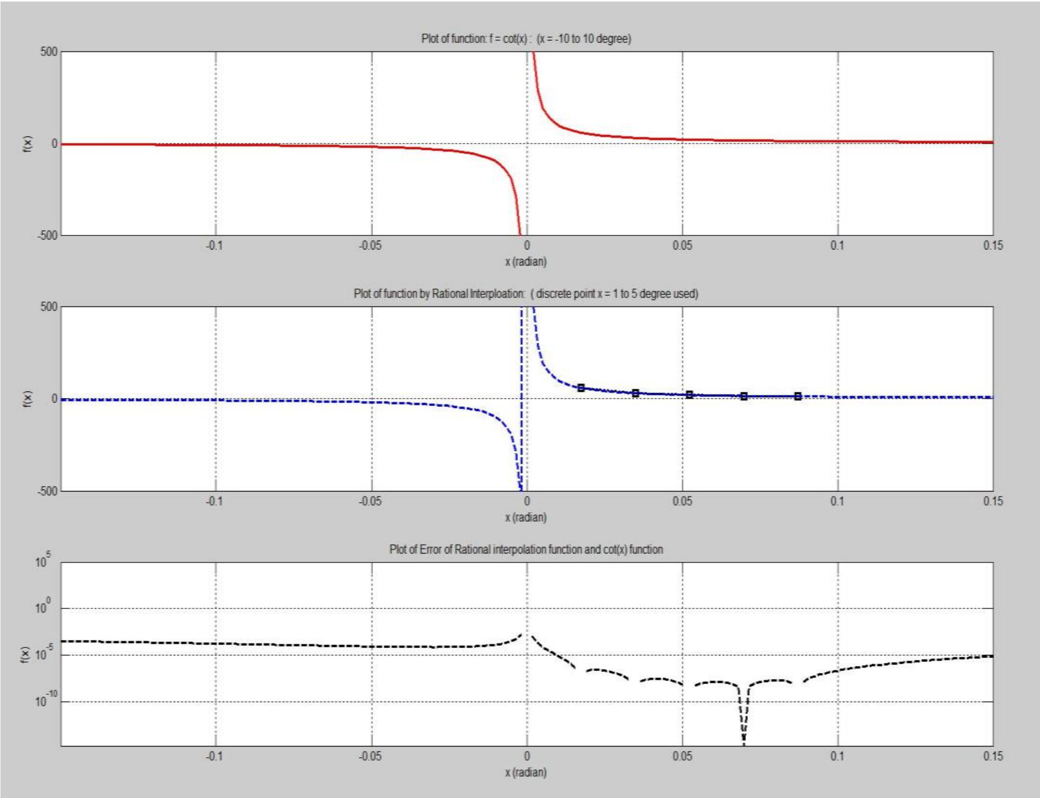

Figure2. Rational Interpolation of function

Top: Plot of f(x) = cot(x), (x range is selected as -10 to 10 degree)

Middle: Plot of Rational Interpolation p(x) by given 5 points

Bot: Plot of Error:|f(x)-p(x)|: (log plot displayed of y axis).

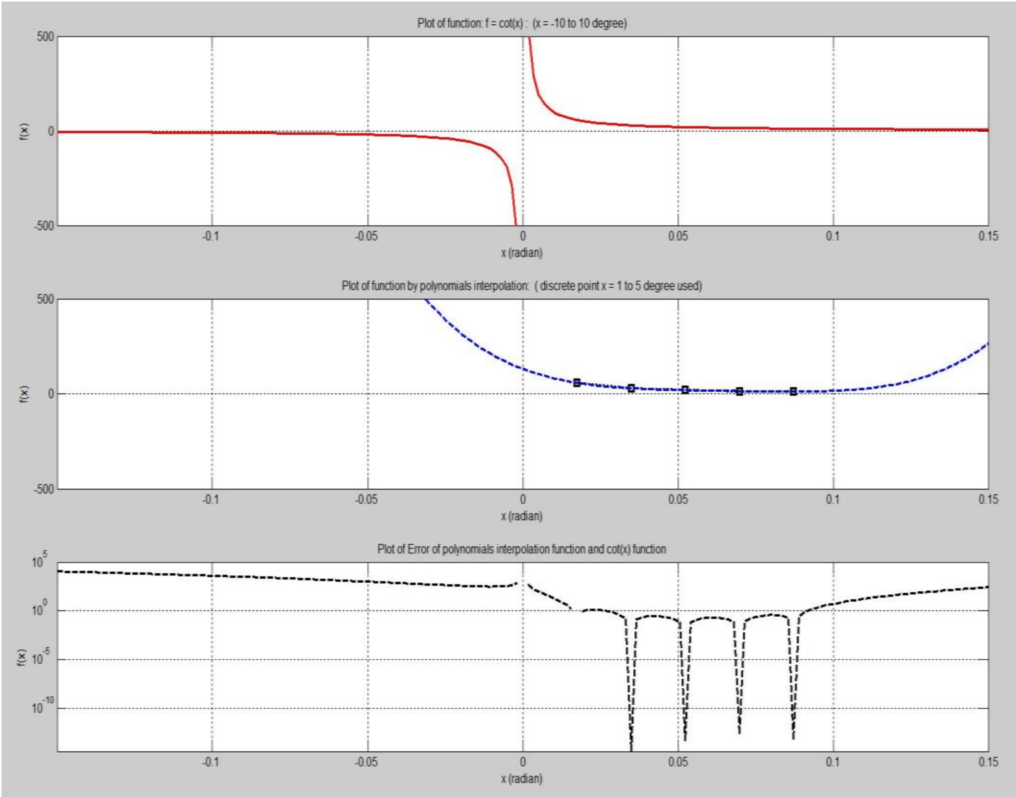

Figure3. Interpolation of function by Polynomial

Top: Plot of f(x) = cot(x), (x range is selected as -10 to 10 degree)

Middle: Plot of Polynomial Interpolation p(x) by given 5 points

Bot: Plot of Error:|f(x)-p(x)|: (log plot displayed of y axis).

Derived Rational Interpolation function:

R(x) = (2.316674e+08 + -4.046230e+06 *x + -7.708923e+07 *x^2)/(1 + 2.316673e+08 *x + -4.040076e+06 *x^2)

For x = 2.5 deg, cot(x) is 2.290377e+01, F(x) from rational interpolation is: 2.290377e+01, error is: 2.660314e-08

Derived Polynomial Interpolation function:

p(x) = (1.308254e+02 + -6.155595e+03 *x + 1.332311e+05 *x^2)/(1 + -1.347102e+06 *x + 5.145551e+06 *x^2)

For x = 2.5 deg, cot(x) is 2.290377e+01, F(x) from polynomials interpolation is: 2.263519e+01 , error is: 2.685740e-01