Symbolic Polynomial Interpolation Using Mathematica

Ali Yazici[1], Irfan Altas[2], and Tanil Ergenc[3]

[1] Computer Engineering Department, Atilim University, Ankara - Turkey
aliyazici@atilim.edu.tr
[2] School of Information Studies, Wagga Wagga - Australia
ialtas@csu.edu.au
[3] Mathematics Department, Middle East Technical University, Ankara - Turkey
tanil@metu.edu.tr

Abstract. This paper discusses teaching polynomial interpolation with the help of Mathematica. The symbolic power of Mathematica is utilized to prove a theorem for the error term in Lagrange interpolating formula. Derivation of the Lagrange formula is provided symbolically and numerically. Runge phenomenon is also illustrated. A simple and efficient symbolic derivation of cubic splines is also provided.

1 Introduction

Students use software tools such as Mathematica or Maple to test mathematical ideas and access the Internet to develop an awareness of the wider learning environment. Kaput [1] has suggested that the mathematical thinking ability to recognize translation from one representation of a function to another can be assisted by the use of computers. The use of a symbolic algebra package in combination with the Internet can develop new strategies and a deeper understanding of many mathematical concepts.

A numerical treatment of Hermite interpolation is studied in [2] using Mathematica. In this paper, we demonstrate the symbolic use of Mathematica in teaching interpolation concepts in a numerical computing course offered to sophomore Engineering students at Atilim University. In addition to 3 hours/week theoretical lectures, practical laboratory sessions are held (2 hours/week) for a group of 20 students to support the theory.

In Section 2 polynomial interpolation and Lagrange interpolating polynomials are discussed. Mathematica instructions are used to derive polynomials and the error formula for the linear case in Sections 2.2 and 2.3 respectively. Runge's phenomenon is demonstrated numerically with equally spaced nodes in Section 2.4. Use of the Chebyshev nodes as interpolation points are displayed in Section 2.5. Finally, Section 3 is devoted to the derivation of piecewise cubic spline interpolation symbolically.

2 Polynomial Interpolation

Let $x_o, x_1, ..., x_n$ be a set of $n+1$ distinct real or complex numbers on $[a, b]$, and let $y_o, y_1, ..., y_n$ be associated function values. Then the polynomial $p(x)$ is said to interpolate $f(x)$ at these points if $p(x_k) = f(x_k)$, $k = 0, 1, ..., n$.

2.1 Lagrange Form of the Interpolating Polynomial

Lagrange form of interpolating polynomial [3] is based on the polynomials

$$L_{n,j}(x) = \prod_{i=0}^{j} \frac{(x - x_i)}{(x_j - x_i)}, \quad i \neq j, \quad j = 0, 1, \dots n \tag{1}$$

where x_i's are interpolation nodes in $[a, b]$. For each j, $L_{n,j}(x)$ is a polynomial of degree n and has the property

$$L_{n,j}(x_i) = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

We will write $L_{n,j}(x)$ simply as $L_j(x)$ when there is no confusion as to its degree. It is easy to see that polynomial $p(x) = \sum_{k=0}^{n} L_k(x) f(x_k)$ has degree n and satisfies the interpolation condition $p(x_j) = f(x_j), j = 0, 1, \dots n$.

If $f^{n+1}(x)$ is continuous on $[a, b]$, then interpolating polynomial $p(x)$approximates $f(x)$ for each x in $[a, b]$ with an error

$$e(x) = f(x) - p(x) = \frac{f^{n+1}(c)}{(n+1)!} W(x) \tag{3}$$

where $W(x) = (x - x_o)(x - x_1) \dots (x - x_n)$ and c is a number in (a, b).

The function $W(x)$ plays an important role in determining the size of the error bound. Among all possible choice for distinct x_j's $j = 0, 1, \dots, n$ in $[a, b] = [-1, 1]$, maximum of $W(x)$ is minimized if x_i's are the roots of the $(n + 1)st$ degree Chebyshev polynomial.

2.2 Session: Lagrange Interpolation with Equally Spaced Points

A Mathematica session is set up to demonstrate the theoretical and practical aspects of interpolation in a step wise manner. In this paper only a part of the experiments are discussed. Complex coding and programming are avoided for educational purposes at the cost of computational efficiency. Firstly, the error formula (3) given above will be proven for the linear case using Mathematica.

The Mathematica instructions to prove the result is given below:

- Define the point set X for x_o and x_1, and compute length of X = m = n+1
  `In[1]:=` $X = \{xo, x1\}$, `In[2]:=` $m = Length[X]$ `Out[2]= 2`

- Define a general function F and define L_j's symbolically in product form
  `In[3]:=` $F[x\_] := f[x]$
  `In[4]:=`$L[j\_, x\_] := Product[If[i == j, 1, (x - X[[i]])/(X[[j]] - X[[i]]), \{i, 1, m\}]$
- Compute, say, $L[2, x]$ and display the interpolating polynomial $p_1[x]$
  `In[5]:=` $L[2, x]$ `Out[5]=` $\frac{x-x_o}{x_1-x_o}$
  `In[6]:=` $p_1[x] := F[x_0]L[1, x] + F[x_1]L[2, x]$
- Define a special function g[t]. Here, x, x_o, and x_1 are constants with respect to t, and g[t] is zero at these points. Also, assume that $x_0 < x < x_1$
  `In[7]:=` $g[t\_] := f[t] - p_1[t] - e1[x]\frac{(t-x_0)(t-x_1)}{(x-x_0)(x-x_1)}$
- Compute the first derivative of g with respect to t
  `In[8]:=` $g\prime[t]$
  `Out[8]=` $-\frac{(t-x_0))e1[x]}{(x-x_0)(x-x_1)} - \frac{(t-x_1)e1[x]}{(x-x_0)(x-x_1)} - \frac{f[x_o]}{x_o-x_1} - \frac{f[x_1}{-x_o+x_1} + f\prime[t] - p_1\prime[t]$
- Compute the second derivative of g with respect to t at c_1
  `In[9]:=` $g\prime\prime[z]$ `Out[9]=` $-\frac{2e1[x]}{(x-x_o)(x-x_1)} + f\prime\prime[z]$
- Applying Rolle's theorem to g[t] on $[x_o, x]$ to find a value d_o in (x_0, x) so that $g\prime[d_o] = 0$. A second application of Rolle's theorem to g[t] on $[x, x_1]$ produces a value d_1 in (x, x_1) so that $g\prime[d_1] = 0$. Observe that $g\prime[t]$ is zero at d_o, and d_1. Therefore, again by Rolle's theorem (applied to $g\prime[t]$ on $[d_o, d_1]$) we find the value c_1 for which $g\prime\prime[c_1] = 0$ which ends the proof.
  `In[10]:=` $Solve[g\prime\prime[c_1] == 0, e1[x]]$
  `Out[10]=` $\{\{e1[x] \to \frac{1}{2}(x - x_o)(x - x_1)f\prime\prime[c_1]\}\}$

This experiment encourages the students to utilize Mathematica for some constructive proofs of simple theorems and for deriving required identites in the method.

2.3 Session: Deriving Lagrange Interpolation Symbolically with Mathematica

In this section, Lagrange interpolating polynomial p2 will be derived symbolically for n=2 using 3 interpolation points.

- Define the point set X for the three (m=n+1) points x_o, x_1, and x_2
  `In[1]:=` $X = \{xo, x1, x2\}$ `Out[1]=` $\{xo, x1, x2\}$
  `In[2]:= m=Length[X]  Out[2]= m=3`
- Define a general function F and L_j's symbolically in product form
  `In[3]:=` $F[x\_] := f[x]$
  `In[4]:=` $L[j\_, x\_] := Product[If[i == j, 1, (x - X[[i]])/(X[[j]] - X[[i]]), \{i, 1, m\}]$
- Display L[1,x] symbolically
  `In[5]:= L[1,x]  Out[5]=` $\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}$
- Form the quadratic interpolating polynomial p2 symbolically
  `In[6]:=` $p2[x\_, X\_] = Sum[F[X[[i]]]L[i, x], \{i, 1, m\}]$
  `Out[6]=` $\frac{(x-x_1)(x-x_2)f[x0]}{(x_0-x_1)(x_0-x_2)} + \frac{(x-x_0)(x-x_2)f[x1]}{(-x_0+x_1)(x_1-x_2)} + \frac{(x-x_0)(x-x_1)f[x2]}{(-x_0+x_2)(-x_1+x_2)}$

Out[6] is the quadratic interpolating polynomial as an approximation to f.

2.4 Session: Runge's Function Illustrated

The polynomial interpolation problem has a unique solution. However, use of equally spaced interpolation points with polynomials of high degree may cause divergence. This is known as Runge's phenomenon and illustrated below.

Consider the function $F(x) = \frac{1}{1+25x^2}$ over the interval [-1,1]. The interpolating polynomials $p2[x]$, and $p8[x]$ with 3, and 9 equally spaced points respectively, in $[-1, 1]$ are formed.

- Define the function and the interpolation points
  `In[1]:=` $F[x\_] := \frac{1}{1+25x^2}$, `In[2]:=` $xo = -1; x1 = 0; x2 = 1$
  `In[3]:=` $X = \{xo, x1, x2\}$ `Out[3]=` $\{-1, 0, 1\}$
  `In[4]:=` $m = Length[X]$ `Out[4]= 3`
- Form L_j's symbolically in product form and compute $p2[x]$.
  `In[5]:=` $L[j\_, x\_] := Product[If[i == j, 1, (x - X[[i]])/(X[[j]] - X[[i]]), \{i, 1, m\}]$ `In[6]:=` $p2[x\_, X\_] = Simplify[Sum[F[X[[k]]]L[k, x], \{k, 1, m\}]$
  `Out[6]=` $1 - \frac{25x^2}{26}$
- Display $p8[x]$. Intermediate steps are similar and omitted.
  `Out[7]=` $1.-1.77636\text{x}10^{-15}x-13.20303x^2-2.13163\text{x}10^{-14}x^3+61.36721x^4+7.10543\text{x}10^{-14}x^5 - 102.81501x^6 + 4.973799\text{x}10^{-14}x^7 + 53.68930x^8$

Now, students can obtain a plot of these polynomials to observe the divergence with higher degree polynomials.

2.5 Session: Lagrange Interpolation with Chebyshev Points

A remedy to Runge's phenomenon is to employ unequally spaced Chebyshev nodes as interpolation points as discussed above. Now, Lagrange interpolating polynomials will be formed at the Chebyshev nodes in a similar fashion.

- Compute the quadratic chebp2 at the Chebyshev nodes, c_k.
  `In[10]:=` $m = 3$
  `In[11]:=` $For[k = 1, k \leq m, k++, c[k] = -Cos[Pi(2k - 1)/(2m)]];$
  `In[12]:=` $X = Table[c[k], k, 1, m]$ `Out[12]=` $\{-0.866025, 0., 0.866025\}$
  `In[13]:=` $L[j\_, x\_] := Product[If[i == j, 1, (x - X[[i]])/(X[[j]] - X[[i]]), \{i, 1, m\}]$
  `In[14]:=` $chebp2[x\_, X\_] = Simplify[Sum[F[X[[k]]]L[k, x], \{k, 1, m\}]]$
  `Out[14]=` $1 + 0.x - 1.26582x^2$
- chebp8 is computed in a similar fashion.
  `Out[15]=`$1.+0.x-9.51343^2-8.88178\text{x}10^{-16}x^3+31.3482x^4-1.95399\text{x}10^{-14}x^5 - 40.3504x^6 - 6.21725\text{x}10^{-15}x^7 + 17.6203x^8$
- Finally, a plot the graph of F, p4, and chebp8 is given to illustrate the Runge's phenomenon (Fig.1) and use of Chebyshev nodes
  `In[17]:=`$Plot[F[x], p4[x, X], chebp8[x, X], x, -1, 1, PlotRange- > All, AxesLabel- > x, y, Ticks- > Automatic, AxesOrigin- > 0, 0]$

Observe that, Chebyshev nodes produces a good approximation and Lagrange method with equally spaced points seems to diverge.

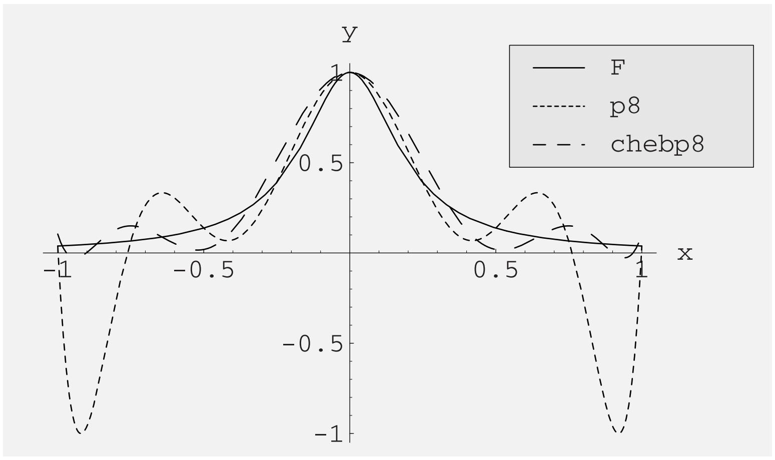

Fig. 1. Plot of Runge function together with p8, and chebp8

3 Cubic Splines

3.1 Background

As shown above, high degree interpolating polynomials may produce divergent approximations. To alleviate these difficulties, piecewise polynomial interpolation is provided [4], [5].

In the sequel, a natural cubic spline over two subintervals using 3 data points will be derived symbolically using Mathematica.

3.2 Session: Deriving Cubic Splines with Mathematica

- Consider a set of three points $(t_i, y_i), i = 1, 2, 3$. The required natural cubic spline is defined by two separate cubic polynomials p1 and p2 in $[t_1, t_2]$ and $[t_2, t_3]$.
  `In[1]:=` $p1[t\_] := a + bt + ct^2 + dt^3$ `,` `In[2]:=` $p2[t\_] := e + ft + gt^2 + ht^3$
- A total of 8 parameters (a,b,c,d,e,f,g, and h) are to be determined. Using the interpolation condition at the end points, we obtain 4 equations
  `In[3]:=` $eq1 = y1 == (a + bt[1] + ct[1]^2 + dt[1]^3)$
  `In[4]:=` $eq2 = y2 == (a + bt[2] + ct[2]^2 + dt[2]^3)$
  `In[5]:=` $eq3 = y3 == (e + ft[2] + gt[2]^2 + ht[2]^3)$
  `In[6]:=` $eq4 = y3 == (e + ft[3] + gt[3]^2 + ht[3]^3)$
- Requiring the first derivative to be continuous at t_2 gives an additional equation as follows:
  `In[7]:=` $eq5 = p1'[t[2]] == p2'[t[2]]$ `Out[7]=` $b + 2ct[2] + 3dt[2]^2 == f + 2gt[2] + 3ht[2]^2$
- Requiring the second derivative to be continuous at t_2 gives
  `In[8]:=` $eq6 = p1''[t[2]] == p2''[t[2]]$ `Out[8]=` $2c + 12d == 2g + 12h$

- Finally, by the conditions of natural cubic spline
  `In[9]:=` $eq7 = p1''[t[1]] == 0$ `Out[9]=` $2c + 6d$
  `In[10]:=` $eq8 = p2''[t[3]] == 0$ `Out[10]=` $2g + 18h$
- Cubic splines can be easily determined by solving the linear system of 8 equations in 8 unknowns using Mathematica's Solve function
  `In[11]:=` $sol = Simplify[Solve[eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8, \{a, b, c, d, e, f, g, h\}]]$
  `Out[11]=`$\{\{a-> 2y1-y2, b-> \frac{1}{2}(-y1+y3), c-> -\frac{3}{4}(y1-2y2+y3), d-> \frac{1}{4}(y1-2y2+y3), e-> 6y1-9y2+4y3, f-> \frac{1}{2}(-13y1+24y2-11y3), g-> \frac{9}{4}(y1-2y2+y3), h-> \frac{1}{4}(-y1+2y2-y3)\}\}$

4 Conclusions

In this paper, use of Mathematica software is demonstrated for teaching polynomial interpolation problem in an educational setting. It has been shown that, symbolic packages are quite effective in deriving the required formula, and even proving some simple constructive theorems as shown to prove the error term of the Lagrange interpolation for the linear case.

Our experience reveals that students learned the concepts better and deeper. At the same time, this approach and the symbolic power of Mathematica provided an interactive research environment for the students to develop new skills and ideas.

With this approach, students may grasp important and difficult concepts easily in a laboratory environment.

The symbolic power provided by Mathematica, has provided a platform to discuss the fundamental and difficult issues related to the interpolation problem and the cubic splines.

The authors are involved in the design of an interactive tool to put all the ideas together in an integrated way to support teaching of numerical methods.

References

1. Kaput, J.: Technology and Mathematics Education, in Handbooks of Research on Mathematics Teaching and Learning (Ed. Grouws, D.A.), MacMillan, New York (1992) 515-556.
2. Reiter, C.A.: Exploring Hermite Interpolation with Mathematica, Primus, 2, 2(1992) 173-182.
3. Mathews, J.H.: Numerical Methods For Computer Science, and Mathematics, Prentice-Hall International (1987).
4. De Boor, C.: A Practical Guide to Splines, Springer Verlag, (1978).
5. Heath, M.T.: Scientific Computing: An Introductory Survey, McGraw-Hill International Editions (1997).