

# Interpolation Polynomials

Mth 351 Spring 2001

Bent E. Petersen

Filename: 351s2001\_interp\_polys.mws

In this worksheet I present a few procedures that are useful in experimenting with interpolation polynomials.

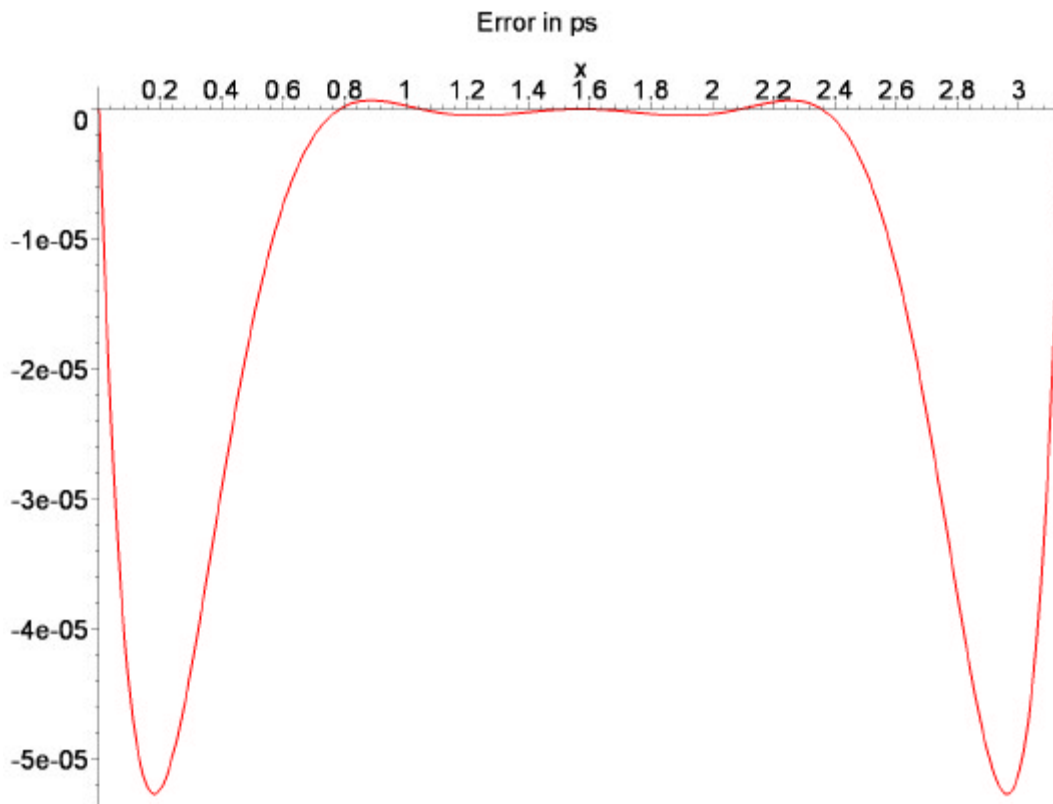
```
[ > restart;
```

finterp() interpolates a function, given the (distinct) nodes, by computing the function values at the nodes and submitting the resulting lists of abscissas and ordinates to Maple interp() function. Note that Maple allows sending any number of unspecified arguments to a procedure, so it is a simple matter to pass lists of data of arbitrary length to a procedure. All the variables passed to the procedure may be accessed through the list args - the nth entry in the list is args[n] - the sublist consisting of the kth to the nth entry is args[k..n]. Note args is not an array, in spite of appearances. The number of entries in the list of parameters is given by nargs.

```
[ > finterp:=proc(f,x)
>   if nargs < 3 then ERROR(FAIL); fi;
>   interp([args[3..nargs]],map(f,[args[3..nargs]]),x);
> end;
```

Here's a quick test of finterp().

```
[ > ps:=finterp(sin,x,0,Pi/4,Pi/3,Pi/2,2*Pi/3,3*Pi/4,Pi):
> plot(sin(x)-ps,x=0..Pi, axes=NORMAL,thickness=3,title="Error in
ps",numpoints=100,resolution=300);
```



We see here the typical error distribution for uniformly distributed nodes, so we have confidence in `finterp()`.

If the coefficients in `ps` are evaluated in floating point above, the accuracy will be reduced quite a bit because the values of `ps` are quite sensitive to roundoff (and other) errors in the coefficients. Thus I have written `finterp()`, and the other routines below to use symbolic calculations (unless floating point data is received). This is inefficient, of course, but allows us to maintain accuracy.

Here's a simple routine, `Unodes()`, which returns a list (not a formal list in the sense of data types) of  $n+1$  uniformly spaced nodes, in order, for an interval  $[a,b]$ . Thus the endpoints determine  $n$  equal subintervals.

```
> Unodes:=proc(n,a,b)
>   local L, k;
>   if n < 1 then ERROR(FAIL); fi;
>   L:=[];
>   for k from 0 to n do L:=[op(L),a+k*(b-a)/n]; od;
>   op(L);
> end;
```

Note `Unodes()` returns the nodes symbolically if possible. We return `op(L)` in `Unodes()` so we get the actual nodes. Returning `L` would return a single data item, a list of the nodes. This is not just a matter

of taste - it allows nargs in other routines to contains the number of nodes plus other parameters, and saves us the trouble of counting the number of entries in a formal list.

Let's test Unodes().

> **Unodes(6,a,b);**

$$a, \frac{5}{6}a + \frac{1}{6}b, \frac{2}{3}a + \frac{1}{3}b, \frac{1}{2}a + \frac{1}{2}b, \frac{1}{3}a + \frac{2}{3}b, \frac{1}{6}a + \frac{5}{6}b, b$$

Cnodes() returns the n+1 Chebyshev nodes for an interval [a,b].

> **Cnodes:=proc(n,a,b)**

> **local L, k;**

> **if n < 1 then ERROR(FAIL); fi;**

> **L:=[];**

> **for k from 0 to n do**

**L:=[op(L),a+(b-a)\*(1-cos((2\*k+1)\*Pi/(2\*n+2)))/2]; od;**

> **op(L);**

> **end;**

Here's a couple of tests of Cnodes().

> **Cnodes(4,-1,1);**

$$-\frac{1}{4}\sqrt{2}\sqrt{5+\sqrt{5}}, -\frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}}, 0, \frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}}, \frac{1}{4}\sqrt{2}\sqrt{5+\sqrt{5}}$$

> **Cnodes(4,a,b);**

$$a + \frac{1}{2}(b-a) \left( 1 - \frac{1}{4}\sqrt{2}\sqrt{5+\sqrt{5}} \right), a + \frac{1}{2}(b-a) \left( 1 - \frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}} \right), \frac{1}{2}a + \frac{1}{2}b, \\ a + \frac{1}{2}(b-a) \left( 1 + \frac{1}{4}\sqrt{2}\sqrt{5-\sqrt{5}} \right), a + \frac{1}{2}(b-a) \left( 1 + \frac{1}{4}\sqrt{2}\sqrt{5+\sqrt{5}} \right)$$

Let's interpolate the sine function on  $[0, \pi]$  with 7 nodes.

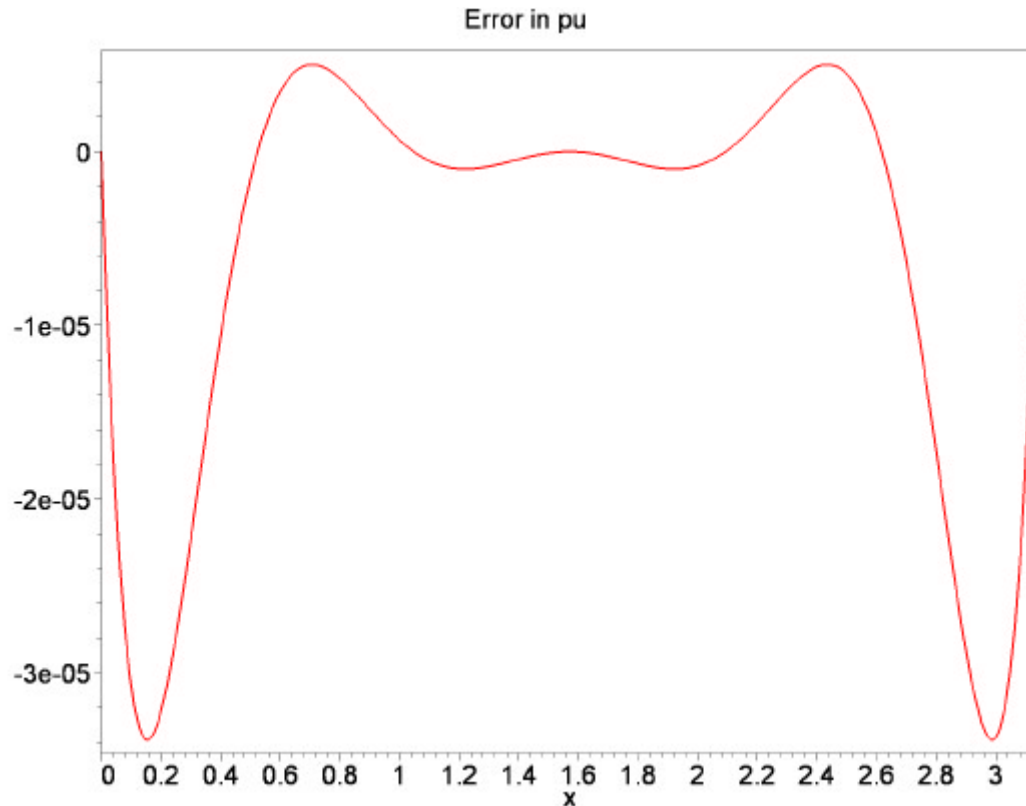
$[0, \pi]$

> **pu:=finterp(sin,x,Unodes(6,0,Pi));**

$$pu := -\frac{8424}{5} \frac{x^6}{\pi^6} + \frac{25272}{5} \frac{x^5}{\pi^5} - 5706 \frac{x^4}{\pi^4} + 2988 \frac{x^3}{\pi^3} - \frac{3566}{5} \frac{x^2}{\pi^2} + \frac{308}{5} \frac{x}{\pi} + 972 \frac{\sqrt{3} x^6}{\pi^6} \\ - 2916 \frac{\sqrt{3} x^5}{\pi^5} + 3294 \frac{\sqrt{3} x^4}{\pi^4} - 1728 \frac{\sqrt{3} x^3}{\pi^3} + \frac{1647}{4} \frac{\sqrt{3} x^2}{\pi^2} - \frac{135}{4} \frac{\sqrt{3} x}{\pi}$$

It's rather precise! Let's convert it to floating point so we can see what it looks like.

```
> evalf(pu,16);  
-0.001296680939813 x6 + .01222092994370 x5 - .00641254180168 x4 - .16073505149962 x3  
- .00284509716638 x2 + 1.00053876843603 x  
> plot(sin(x)-pu,x=0..Pi,axes=BOXED,thickness=3,title="Error in  
pu",numpoints=100,resolution=300);
```



The symbolic Chebyshev nodes are fairly complicated here. If we use them to compute the exact interpolation polynomial we obtain an expression that covers many pages and consumes a great deal of RAM. Maple is likely to choke on this expression, or at least be very slow. Thus we convert the Chebyshev nodes to floating point with 16 decimal digits before computing the interpolation polynomial.

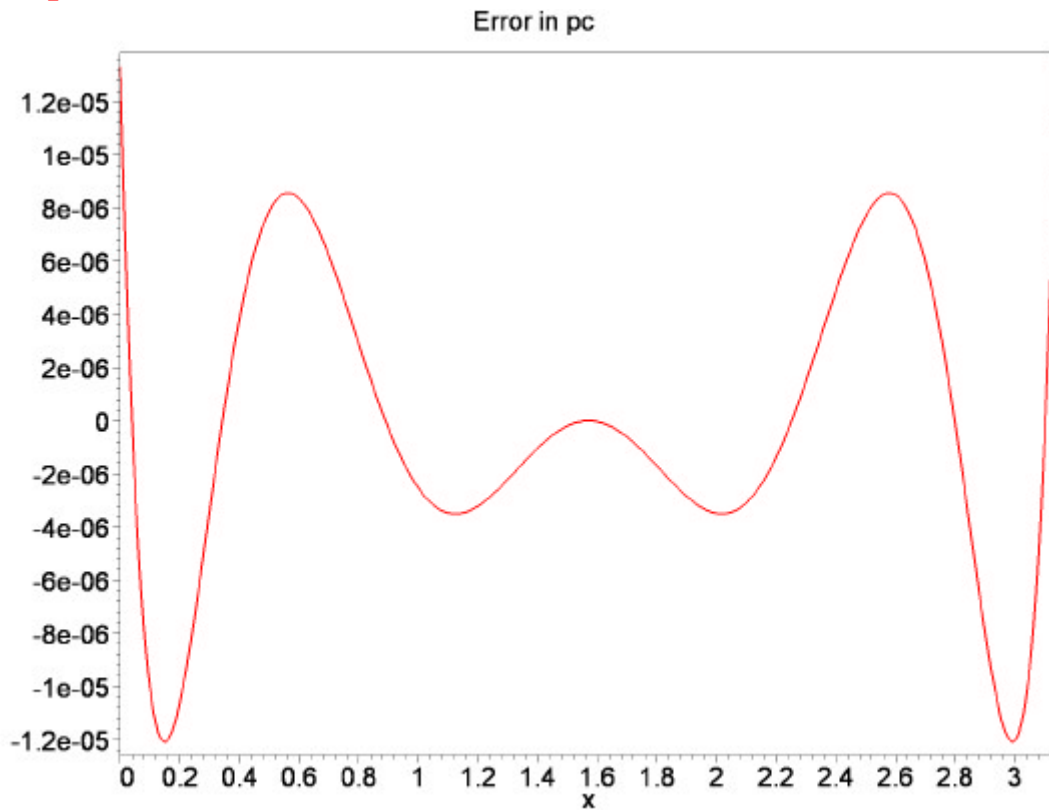
```
> pc:=finterp(sin,x,evalf(Cnodes(6,0,Pi),16));  
pc := -0.001285392572 x6 + .01211453953 x5 - .006033382849 x4 - .1613673246 x3  
- .002364997394 x2 + 1.000423400 x - .00001331620
```

Generally there is no telling in what order the terms in a polynomial will returned. We can put them in standard order by using the function sort().

```
> sort(pc,x);
```

$$-.001285392572 x^6 + .01211453953 x^5 - .006033382849 x^4 - .1613673246 x^3 \\ - .002364997394 x^2 + 1.000423400 x - .00001331620$$

```
> plot(sin(x)-pc,x=0..Pi,axes=BOXED,thickness=3,title="Error in  
pc",numpoints=100,resolution=300);
```



As we expected the Chebyshev nodes give a better distributed error. Moreover the maximum errors are smaller than in the uniform nodes case.

```
>
```