

Divided Differences

nalib

We begin with the class Maple library.

```
> restart;
> libname:="c:/nalib",libname;
libname := "/nalib", "/Library/Frameworks/Maple.framework/Versions/15/lib",
"/Library/Frameworks/Maple.framework/Versions/15/toolbox/NAG/lib"
> with(numanal);
[SOR, SOR_dir, adaptq, adaptq_dir, bezier, bezier_dir, bisection, bisection_dir, chop, chop_dir,
clamped_spline, clamped_spline_dir, divided_diff, divided_diff_dir, extrap, extrap_dir,
falseposition, falseposition_dir, fixedpoint, fixedpoint_dir, gaussseidel, gaussseidel_dir, hermite,
hermite_dd, hermite_dd_dir, hermite_dir, horner, horner_dir, jacobi, jacobi_dir, muller,
muller_dir, natural_spline, natural_spline_dir, newton, newton_dir, romberg, romberg_dir,
secant, secant_dir, steffensen, steffensen_dir]
```

We are going to study interpolation with Newton's divided differences with the data points generated by Runge's function

$$f(x) = \frac{1}{1 + 25x^2}$$

to four decimal places on the interval $[-1, 1]$. We first enter Runge's function.

```
> f:=1/(1+25*x^2);
```

$$f := \frac{1}{1 + 25x^2}$$

We will interpolate at 5 equally spaced points, so we enter the x and y lists.

```
> X:=[-1,-.5,0,.5,1];
```

$$X := [-1, -0.5, 0, 0.5, 1]$$

```
> Y:=[.0385,.1379,1.0000,.1379,.0385];
```

$$Y := [0.0385, 0.1379, 1.0000, 0.1379, 0.0385]$$

We will use the command **divided_diff** to prepare the divided difference table and form the polynomial. Let's first check the directions.

```
> divided_diff_dir();
divided_diff returns the divided-differences table
and the interpolating polynomial
with 7 decimal place rounding.
```

The arguments for `divided_diff` are:

- (1) the list of x-values
- (2) the list of $f(x)$ or y-values
- (3) the variable for returning the polynomial

If assigning the result to a variable, have the variable and the 3rd argument the same.

If `p` is the variable for returning the polynomial and has already been given a value, the procedure should be preceded by the statement:
`p:='p'`

```
> p:=divided_diff(x,y,p);
```

i	x[i]	y[i]				
0	-1.000	0.0385000				
			0.1988000			
1	-0.500	0.1379000		1.5254000		
			1.7242000		-3.3158667	
2	0.000	1.0000000		-3.4484000		3.3158667
			-1.7242000		3.3158667	
3	0.500	0.1379000		1.5254000		
			-0.1988000			
4	1.000	0.0385000				

The interpolating polynomial is $.2373000000+.1988000000*x+1.525400000*(x+1)*(x+.5)-3.315866700*(x+1)*(x+.5)*x+3.315866700*(x+1)*(x+.5)*x*(x-.5)$

$$p := 1.000000000 - 4.277366700 x^2 + 3.315866700 x^4$$

Let's graph both `p` and `f` along with the points of interpolation. To do the latter, we need to first read in the `plots` package so that we can use the `pointplot` command. Assigning plots to variables keeps them from displaying.

```
> with(plots);
```

```
[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot, display, dualaxisplot, fieldplot, fieldplot3d, gradplot, gradplot3d, implicitplot, implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions, setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot]
```

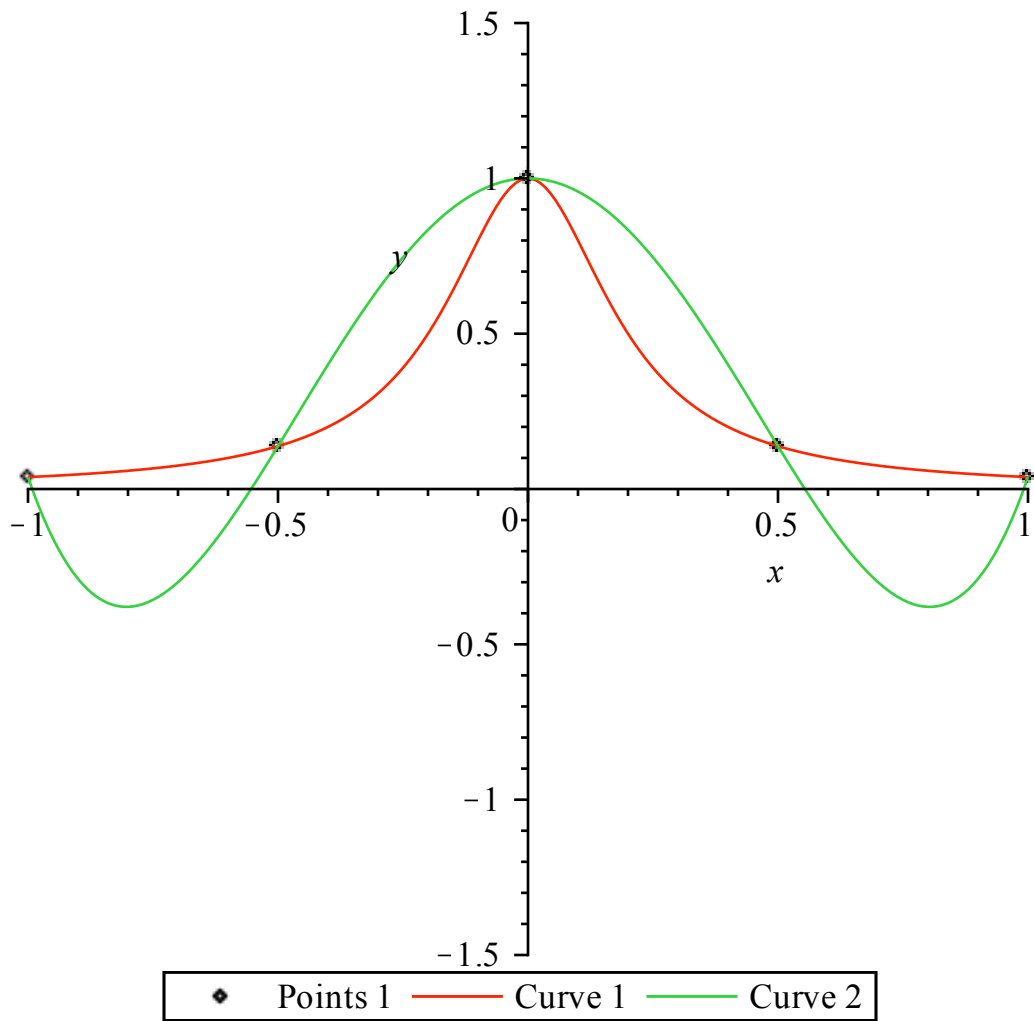
```
> plot1:=pointplot({seq( [X[i],Y[i]], i=1..5)});
plot1 := PLOT(...)
```

We see that we nevertheless get output, namely the Maple plotting information. We suppress this with a colon in the future.

```
> plot2:=plot({f,p},x=-1..1,y=-1.5..1.5):
```

We use the `display` command to view the plots on a single axis system.

```
> display(plot1,plot2);
```



We see that the polynomial does not approximate the function very well. The graph of the function is the graph entirely above the x -axis. Let's see what happens if we add four more points. Notice how `op` is used to avoid rewriting the previous data.

```
> X:=[op(X),-.75,-.25,.25,.75];
      X:=[-1, -0.5, 0, 0.5, 1, -0.75, -0.25, 0.25, 0.75]
> Y:=[op(Y),.0664,.3902,.3902,.0664];
      Y:=[0.0385, 0.1379, 1.0000, 0.1379, 0.0385, 0.0664, 0.3902, 0.3902, 0.0664]
```

```
> p1:=divided_diff(X,Y,p1);
i   x[i]   y[i]
--   ---   ---
0  -1.000  0.0385000
1  -0.500  0.1379000   0.1988000   1.5254000
2   0.000  1.0000000   1.7242000   -3.4484000   -3.3158667   3.3158667
3   0.500  0.1379000  -1.7242000   1.5254000   3.3158667   4.3478096
4   1.000  0.0385000  -0.1988000  -0.1462857   2.2289143   6.8647312
5  -0.750  0.0664000  -0.0159429  -0.5308343   0.5127315   1.4281756
      0.6476000   0.1556876
```

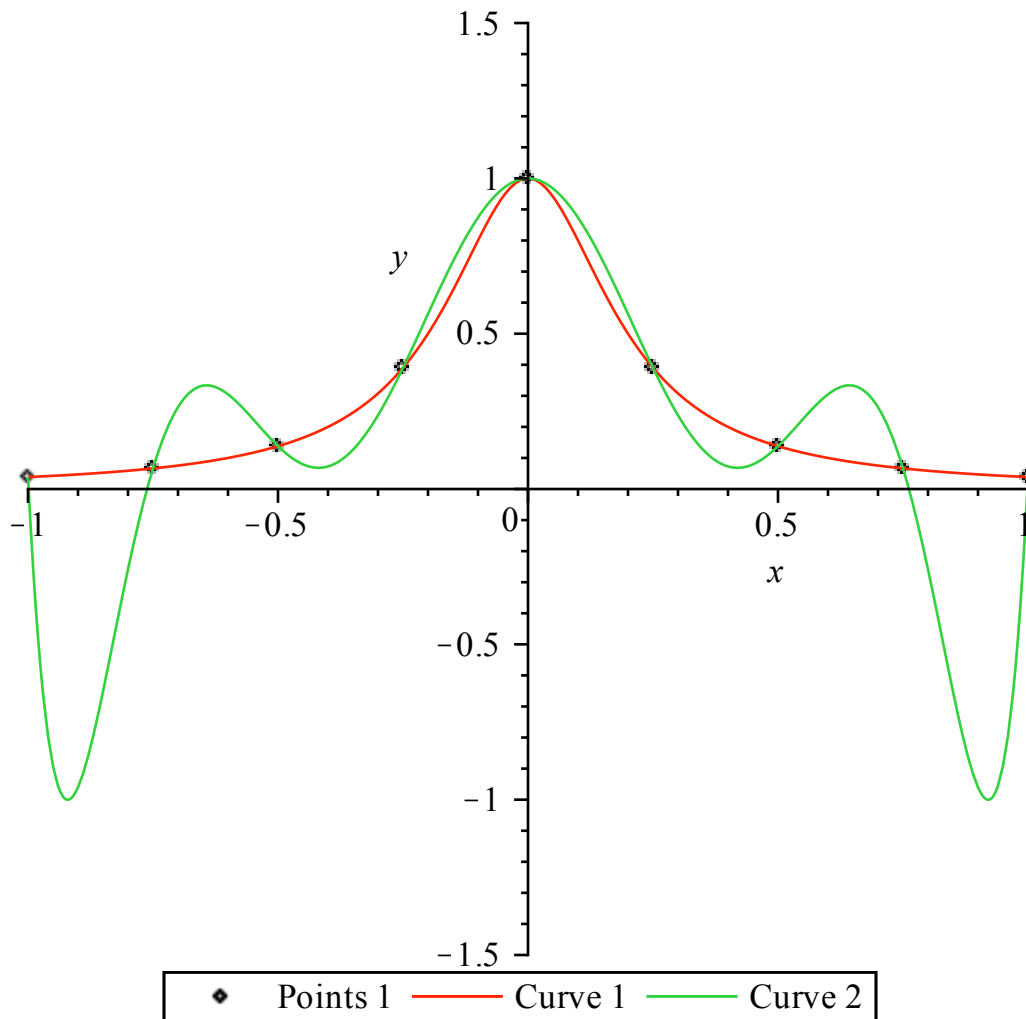
6	-0.250	0.3902000		-0.6476000		0.6227504
7	0.250	0.3902000	0.0000000	-0.6476000	0.0000000	
8	0.750	0.0664000	-0.6476000			
	4.1277716					
		7.9198864				
	10.0676864		-40.2707452			
		-42.4185451		53.6943263		
	-21.7462224		53.6943258			
		24.6993621				
	-3.2217008					

The interpolating polynomial is $.2373000000+.1988000000*x+1.525400000*(x+1)*(x+.5)-3.315866700*(x+1)*(x+.5)*x+3.315866700*(x+1)*(x+.5)*x*(x-.5)+4.127771600*(x+1)*(x+.5)*x*(x-.5)*(x-1)+7.919886400*(x+1)*(x+.5)*x*(x-.5)*(x-1)*(x+.75)-40.27074520*(x+1)*(x+.5)*x*(x-.5)*(x-1)*(x+.75)*(x+.25)+53.69432630*(x+1)*(x+.5)*x*(x-.5)*(x-1)*(x+.75)*(x+.25)*(x-.25)$

$$p1 := 1.000000000 - 13.20405530 x^2 - 3.000000000 10^{-7} x^3 + 61.37289100 x^4 + 7.000000000 10^{-7} x^5 - 102.8246621 x^6 - 5.000000000 10^{-7} x^7 + 53.69432630 x^8$$

Let's check this new interpolating polynomial versus the function.

```
> plot3:=pointplot({seq( [X[i],Y[i]], i=1..9)}):
> plot4:=plot({f,p1},x=-1..1,y=-1.5..1.5):
> display(plot3,plot4);
```



We get an approximation that is even worse in some cases. This is a famous example of a function where you do not get a good approximation from an interpolating polynomial and adding more points just makes the situation worse.

NumericalAnalysis

The command [PolynomialInterpolation](#) in the [NumericalAnalysis](#) subpackage can also be used for Newton's divided differences.

```
> with(Student[NumericalAnalysis]):
```

We need to provide the data as a list of XY data pairs. We start with 5 points.

```
> XY:= [seq( [X[i],Y[i]], i=1..5)];
      XY := [ [-1, 0.0385], [-0.5, 0.1379], [0, 1.0000], [0.5, 0.1379], [1, 0.0385] ]
```

For Newton's divided-difference method, we use the argument **method=newton**.

```
> p1:=PolynomialInterpolation(XY,method=newton,function=f(x),
  independentvar=x);
```

```
p1 := POLYINTERP( [ [-1, 0.0385], [-0.5, 0.1379], [0, 1.0000], [0.5, 0.1379], [1, 0.0385] ], method
= newton, function =  $\frac{1}{1 + 25x(x)^2}$ , independentvar = x, INFO )
```

The procedure **POLYERP** is where all the computed information is stored. We find the divided-difference table.

```
> DividedDifferenceTable(p1);
```

0.0385	0	0	0	0
0.1379	0.1988000000	0	0	0
1.0000	1.724200000	1.525400000	0	0
0.1379	-1.724200000	-3.448400000	-3.315866667	0
0.0385	-0.1988000000	1.525400000	3.315866667	3.315866667

We get a triangular table that is slightly different than the one above, but can be easily related to it. Notice that the i and x_i columns are not included here. We find the interpolating polynomial.

```
> p:=Interpolant(p1);
```

$$p := 0.2373000000 + 0.1988000000 x + 1.525400000 (x + 1.) (x + 0.5) - 3.315866667 (x + 1.) (x + 0.5) x + 3.315866667 (x + 1.) (x + 0.5) x (x - 0.5)$$

Notice the **basis functions** it is given in terms of.

```
> b:=BasisFunctions(p1);
```

$$b := [1, x + 1., (x + 1.) (x + 0.5), (x + 1.) (x + 0.5) x, (x + 1.) (x + 0.5) x (x - 0.5)]$$

We simplify our polynomial.

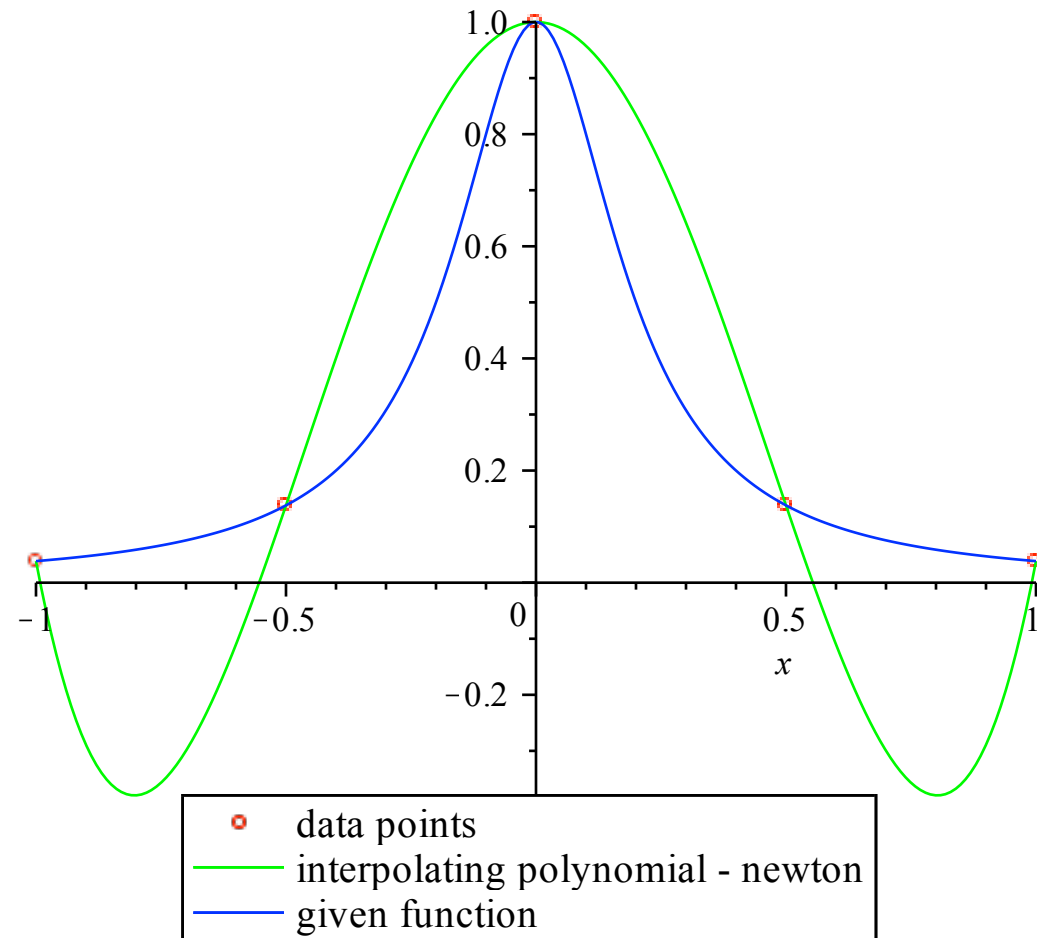
```
> p:=expand(p);
```

$$p := 1.000000000 - 1. 10^{-9} x - 4.277366667 x^2 + 3.315866667 x^4$$

The small differences from what we got above are due to different rounding schemes. We use the [Draw](#) command to plot the function and polynomial interpolant.

```
> Draw(p1);
```

Polynomial interpolation



Now we add four more points to our data list. Again, `op` is used to avoid rewriting the previous data.

```
> XY1:=op(XY),seq( [X[i],Y[i]], i=6..9)];
XY1 := [[ -1, 0.0385], [ -0.5, 0.1379], [ 0, 1.0000], [ 0.5, 0.1379], [ 1, 0.0385], [ -0.75, 0.0664], [
-0.25, 0.3902], [ 0.25, 0.3902], [ 0.75, 0.0664]]
```

We compute the new **POLYINTERP** structure.

```
> p2:=PolynomialInterpolation(XY1,method=newton,function=f(x),
independentvar=x);
p2 := POLYINTERP ( [[ -1, 0.0385], [ -0.5, 0.1379], [ 0, 1.0000], [ 0.5, 0.1379], [ 1, 0.0385], [ -0.75,
0.0664], [ -0.25, 0.3902], [ 0.25, 0.3902], [ 0.75, 0.0664]], method = newton, function
=  $\frac{1}{1 + 25 x(x)^2}$ , independentvar = x, INFO )
```

We view our new divided-difference table.

```
> DividedDifferenceTable(p2);
[[0.0385, 0, 0, 0, 0, 0, 0, 0, 0],
[0.1379, 0.1988000000, 0, 0, 0, 0, 0, 0, 0],
```

```
[1.0000, 1.724200000, 1.525400000, 0, 0, 0, 0, 0, 0],
[0.1379, -1.724200000, -3.448400000, -3.315866667, 0, 0, 0, 0, 0],
[0.0385, -0.1988000000, 1.525400000, 3.315866667, 3.315866667, 0, 0, 0, 0],
[0.0664, -0.01594285714, -0.1462857143, 2.228914285, 4.347809528, 4.127771444, 0, 0, 0],
[0.3902, 0.6476000000, -0.5308342857, 0.5127314285, 6.864731424, 10.06768758,
7.919888181, 0, 0],
[0.3902, 0., -0.6476000000, 0.1556876191, 1.428175238, -21.74622474, -42.41854976,
-40.27075035, 0],
[0.0664, -0.6476000000, -0.6476000000, 0., 0.6227504764, -3.221699046, 24.69936759,
53.69433388, 53.69433385]]
```

We find the interpolating polynomial.

```
> p:=Interpolant(p2);
```

```
p := 0.2373000000 + 0.1988000000 x + 1.525400000 (x + 1.) (x + 0.5) - 3.315866667 (x
+ 1.) (x + 0.5) x + 3.315866667 (x + 1.) (x + 0.5) x (x - 0.5) + 4.127771444 (x + 1.) (x
+ 0.5) x (x - 0.5) (x - 1.) + 7.919888181 (x + 1.) (x + 0.5) x (x - 0.5) (x - 1.) (x
+ 0.75) - 40.27075035 (x + 1.) (x + 0.5) x (x - 0.5) (x - 1.) (x + 0.75) (x + 0.25)
+ 53.69433385 (x + 1.) (x + 0.5) x (x - 0.5) (x - 1.) (x + 0.75) (x + 0.25) (x - 0.25)
```

We list the basis functions.

```
> b:=BasisFunctions(p2);
```

```
b := [1, x + 1., (x + 1.) (x + 0.5), (x + 1.) (x + 0.5) x, (x + 1.) (x + 0.5) x (x - 0.5), (x
+ 1.) (x + 0.5) x (x - 0.5) (x - 1.), (x + 1.) (x + 0.5) x (x - 0.5) (x - 1.) (x + 0.75), (x
+ 1.) (x + 0.5) x (x - 0.5) (x - 1.) (x + 0.75) (x + 0.25), (x + 1.) (x + 0.5) x (x
- 0.5) (x - 1.) (x + 0.75) (x + 0.25) (x - 0.25)]
```

We simplify our polynomial.

```
> p:=expand(p);
```

```
p := -4. 10-9 x + 1.000000000 + 2. 10-8 x3 + 61.37289767 x4 - 13.20405618 x2 - 5.0 10-8 x5
- 102.8246754 x6 + 4. 10-8 x7 + 53.69433385 x8
```

We plot the function and polynomial interpolant.

```
> Draw(p2);
```

Polynomial interpolation

