Project 3: Interpolation

Due: November 8, 1999

In this session, we will learn about interpolating functions.

Suppose that one has a table of data describing the change in the U.S. population over a period of one hundred years. A plot of this data would be a finite set of points in the plane. To make this plot useful, one would like to draw a smooth curve through those points which can be used to estimate intermediate population values. This process of estimating intermediate values is called ''interpolation''. There are many ways to interpolate the intermediate values, each having its own strengths and weaknesses. A method is chosen depending on what kind of information one has about the function that one is interpolating.

We will learn about three types of interpolation: Lagrange interpolation, Hermite interpolation, and interpolation by cubic splines. .

Lagrange Interpolation

Lagrange does not rhyme with exchange. It is a French name, and it sounds more like ''The garage''.

Lagrange interpolation is a widely used method for approximating a set of data with a polynomial called ''the Lagrange polynomial''. The Lagrange polynomial for a set of data is the polynomial of least degree that passes through all data points. The degree of the Lagrange polynomial that interpolates $n+1$ data points is n. The method of Lagrange interpolation is so common that Maple has a built-in routine for computing the Lagrange polynomial. For example, suppose we want to find the polynomial of least degree that passes through the points $[[0, 0], [1, 2], [2, 0], [3, 1]]$.

```
> plot([[0,0],[1,2],[2,0],[3,1]],x=-1..4,y=-1..4,style=point,symbol=circle,color=blue);
```

Then we call the routine *interp* on the list of x-coordinates, followed by the list of y-coordinates, and the name of the variable:

```
> L:=interp([0,1,2,3],[0,2,0,1],x);
> plot([L,[[0,0],[1,2],[2,0],[3,1]]],x=-1..4,y=-1..4,style=[line,point],symbol=circle,co
  lor=[red,blue]);
```

The nice thing about Lagrange interpolation is that we only need to know the values of the function to produce the Lagrange polynomial - we don't need any information about the derivatives and higher derivatives of the function. The bad thing about Lagrange interpolation is that to interpolate a function at a large number of points requires a polynomial of high degree, and high degree polynomials have lots of wiggles in them.

Exercises

1. Interpolate the function $f(x) = x^{\left(\frac{1}{5}\right)}$ at four equally-spaced points on the interval $[0, 1]$. Plot $f(x)$ together with your approximation on the same coordinate system. Save your result to a diskette. The Lagrange polynomial from this problem will be used in a later problem.

2. The Weasel Census Bureau reports that in 1975 there were 11.3 million weasels in the U.S. Over the years the weasel population has fluctuated: in 1980 it was 10.3 million weasels, in 1985 it was 3.2 million weasels, in 1990 it was 3.6 million weasels, and in 1995 it was 3.6 million weasels. Estimate the number of weasels in the U.S. in 1982 and 1991.

3. Use Lagrange interpolation to find a polynomial whose graph above the x-axis on the interval $[-\sqrt{5}, \sqrt{5}]$ lies entirely between the two circles determined by the equations $x^2 + y^2 = 4$ and $x^2 + y^2 = 5$.

Save and print your work before going on.

Hermite Interpolation

Hermite does not rhyme with termite. It's a French guy's name, and it sounds more like ''Air-meat''.

Hermite interpolation is an extension of Lagrange interpolation which takes advantage of information about derivatives. Specifically, to interpolate the function $f(x)$ at data points $[x_0, x_1, x_2, x_3]$ we need to know the function values at these points and the values of the derivative. The Hermite polynomial is the polynomial of least degree which has the same function values and derivative values as $f(x)$. If we interpolate $f(x)$ at $n+1$ points, then the degree of the Hermite polynomial is $2n+1$.

```
> n:=3:
> x[0]:=0.0: x[1]:=1.0: x[2]:=2.0: x[3]:=3.0:
> y[0]:=3.0: y[1]:=1.0: y[2]:=0.0: y[3]:=-2.0:
> z[0]:=-2.0: z[1]:=0.0: z[2]:=3.0: z[3]:=1.0:
> P[0]:=0:
>   for j from 0 to 2*n+1 do
>      P[j+1]:= P[j]+a[j]*x^j: od:
>   Q:=unapply(P[2*(n+1)],x):
>   eqns:=NULL: vars:=NULL:
>   for i from 0 to 2*n+1 do vars:=vars, a[i]: od:
>   for i from 0 to n do
>     eqns:=eqns, Q(x[i])=y[i], D(Q)(x[i])=z[i]: od:
>   coefs:=fsolve({eqns},{vars}):
>   H:=unapply(subs(coefs,Q(x)),x);
> data:=[seq([x[i],y[i]],i=0..3)];
> plot([H(x),data],x=-.2..3.2, style=[line,point], symbol=circle,color=[red,blue]);
```

Exercises

4. Interpolate the function $f(x) = x^{\left(\frac{1}{5}\right)}$ at $x_0 = .001$, $x_1 = \frac{1}{3}$, $x_2 = \frac{2}{3}$, and $x_3 = 1$ using Hermite interpolation. You will have to calculate function values and derivatives in order to do this. Plot $f(x)$ together with your approximation on the same coordinate system. Save your result to a diskette. The Hermite polynomial from this problem will be used in a later problem.

5. Interpolate $f(x) = \sin(x)$ at five equally-spaced points on the interval $[0, \pi]$ using Hermite interpolation. Plot the difference between your approximation and the function over the given interval.

6. Use Hermite interpolation to approximate the graph of the curve defined implicitly by $x\,y + y^3 = 2\sin(x)$ on the interval $[-1, 7]$. Plot the function together with your approximation on the interval $[-2, 8]$. (Hints: (a) the curve has a vertical tangent at the origin, so do not choose $x_i = 0$ for one of the data points; (b) to plot implicit functions you should use the implicitplot command. To use it you should load the plots package by typing ''with(plots):''.)

Save and print your work before going on.

Cubic Splines

Cubic splines are piecewise defined functions used in interpolation to avoid the excess wiggle that is introduced by using high-degree polynomials in Lagrange and Hermite interpolation. The basic idea is to fit a cubic polynomial P_k between successive data points x_k and x_{k+1} so that it agrees with the data at the endpoints and the polynomials P_k fit together to produce a twice differentiable function. The reason cubic polynomials are chosen is that they provide enough flexibility to produce a twice differentiable piecewise defined function, but have minimal total curvature among all functions that fit the conditions. Minimizing curvature smooths out the wiggles.

This procedure is sufficiently popular that Maple has a built-in spline routine. We will use the data from the first example above so that the syntax of the spline command is clear.

```
> readlib(spline):
```

```
> xvals:=[0,1,2,3]:
> yvals:=[0,2,0,1]:
> S:=spline(xvals,yvals,x);
> plot([S,[[0,0],[1,2],[2,0],[3,1]]],x=-1..4,y=-1..4,style=[line,point],symbol=circle,co
  lor=[red,blue]);
```

It's a fine time to spline the sine!

```
> f := x -> x*sin(x);
> n:=4:
> for i from 0 to n do x[i]:=2*Pi*i/n: y[i]:=f(x[i]): od:
> xvals:=[seq(x[i],i=0..n)]; yvals:=[seq(y[i],i=0..n)];
> sin_spline:=spline(xvals,yvals,x);
> plot([f(x),sin_spline],x=0..2*Pi,color=[red,blue]);
>
```

Exercises

7. Interpolate $f(x) = x^{\left(\frac{1}{5}\right)}$ at four equally-spaced points on the interval $[0, 1]$ using cubic splines. Plot the following three functions on the same coordinate system: $L(x) - f(x)$ where $L(x)$ is the Lagrange polynomial from Exercise 1, $H(x) - f(x)$ where $H(x)$ is the Hermite polynomial from Exercise 4, and $S(x) - f(x)$ where $S(x)$ is the cubic spline from this problem.

8. Interpolate the upper half of the circle defined by $x^2 + y^2 = 1$ using cubic splines. Use seven data points equally spaced around the circumference of the semicircle. (This does NOT mean that the seven points will have equally-spaced x-coordinates!) Plot the semicircle and your approximation on the same coordinate system. (Important note: Maple's spline routine only accepts data points whose x-values are in increasing order.)