Polynomials and Functions Wrap Up

Instructions: Work on solving each of the questions using Maple and/or Matlab. Turn in a *nice write up* of your solutions, attaching Maple/Matlab graphics and commands where appropriate. DUE: Before you leave for Spring Break, or Friday at 3PM, whichever comes first. You do not need to attend lab next Friday, we'll reserve that time in case you need it.

To get the orthogonal polynomials in Maple, you may use the `orthopoly` package. Typing `with(orthopoly)` lets you generate the Legendre polynomial of degree n by typing `P(n,x)`. See the help page for more. Below is a sample `for` loop that will compute the Legendre coefficients in Maple, assuming that an expression `f` has been previously defined.

```
for i from 0 to 6 do
  c[i]:=evalf(int(f*P(i,x),x=-1..1)/int((P(i,x))^2,x=-1..1));
end
```

You might either modify this loop, or write a second loop to construct the full approximation:

$$f(x) \approx c_0 P(0,x) + c_1 P(1,x) + \ldots + c_k P(k,x)$$

(Global) Polynomial Approximations

Consider the function $f(x) = \dfrac{1}{1+25x^2}$. We can approximate f using several techniques: Taylor, interpolation at $n+1$ points, or by projecting f into an orthogonal polynomial basis. We will be comparing these techniques.

1. Compute the degree 20 Taylor approximation in Maple and compare this (graphically) to $f(x)$. Use Maple's `taylor` command. If PT is the Taylor approximation, use `convert(PT, polynom)` to change the expression into a polynomial that you can plot.

2. Compute the degree 10 Legendre and Chebyshev (first kind) approximations to $f(x)$ and compare (graphically) each to $f(x)$. Compute the error of the approximation (Remember how the error is computed in this vector space with the new inner product!). This gives the distance from f to the subspace spanned by the first 10 basis vectors (and is therefore a 10-dimensional approximation to f). You might repeat this experiment with degree 20 for question 4.

3. Use 21 evenly spaced points and compute the degree 20 approximation to $f(x)$ using interpolation. Compute and compare the error to the previous problem. For the next question, think about this: Is it possible, by increasing the number of interpolation points, to do as well as Legendre and Chebyshev? Try it!

4. Summarize your findings from the first three problems. We have three basic techniques for performing a polynomial approximation to a function $f(x)$ on an interval from $[a,b]$. Compare and contrast the three techniques, especially about how the approximations do as the degree of the approximation increases- Why is this (an intuitive explanation will do, and think about whether or not your answer is specific to this particular $f(x)$)?

Before we get started, here are some new commands we'll use. The first is a sample script in Matlab that computes a cubic spline over some data, then evaluates the spline and plots it. The new commands are `spline` and `ppval` (for piecewise polynomial evaluation).

```
%Example:
%This generates a spline and samples it over a finer mesh:

        x = 0:10;  y = sin(x);  %Data for the spline ("knots")

        xx = 0:.25:10;          %Points where the resulting
                                % spline is to be evaluated

        PP = spline(x,y);       %Create the structure of coefs

        yy = ppval(PP,xx);      %Evaluate the spline on xx

        plot(x,y,'o',xx,yy)
```

There are $4n$ coefficients that define the spline; the $n \times 4$ matrix can be accessed by typing `PP.coefs`

(Local) Cubic Splines

You might read over Section 5.1 (and Definition 5.1) in our text. For problem 2 and 3, you can simply turn in your Matlab code and output.

1. (Problem 3, p. 99- Modified slightly) Write the equations for creating a cubic spline on the following three points, if the first derivatives of the spline at the endpoints are as given below:

   $$\text{Data: } (0,1), (1,6), (2,5)$$

   with $C'(x_0) = -1, C'(x_1) = 1$. Look in the `help` file in Matlab to see how this would be done. Use linear algebra to find the coefficients, then compare them to what `spline` outputs.

2. In statistics, we often need to evaluate

   $$\int_a^b \mathrm{e}^{-x^2}\, dx$$

   In this problem, we will estimate this using cubic splines. As usual, you will probably want to write a script file to solve this problem.

   (a) If $C_i(x)$ is the cubic spline on $[x_i, x_{i+1}]$, compute $\displaystyle\int_{x_i}^{x_{i+1}} C_i(x)\, dx$. Look up `mkpp` in Matlab to see how you would construct the antiderivative from the coefficients given in `spline`.

   (b) Estimate the integral from -1 to 1 using 10 knots (evenly spaced). You should take advantage of the commands `spline`, `mkpp` and `ppval`.

3. (Using cubic splines to estimate a planar curve) We'll assume our data represents two parameterized functions, $(x(t), y(t))$. We will get a cubic spline representation of each function, then plot them together (Note that t will not be specified from the data- You can use the indices 1, 2, etc. for time values). Download the data set `unknown.mat` from our website. To get the data into Matlab, type `load unknown`, and you will have two columns of data. (a) Find a cubic spline representation for $x(t)$, then $y(t)$ (b) Evaluate the splines at 100 evenly spaced points in time, (c) Plot the results as $(x(t), y(t))$.