Lab 09: MATLAB Interpolation Routines & their Derivatives

The Matlab functions dealing with polynomials store the coefficients in *descending* order. In other words the values of the vector of coefficients `p` yield the polynomial

$$y = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + p_{n-1}$$

In most math textbooks, including the notes for this course, discuss polynomials in *ascending* order, meaning for a set of coefficients `p` we would have the polynomial

$$y = p_0 + p_1 x + p_2 x^2 + \cdots + p_{n-1} x^{n-1} + p_n x^n$$

Built-in Interpolation Functions

Table 1

Command	Description
`polyfit(x,y,n)`	`x`, `y` set of data you want to fit, `n` degree of polynomial If you want polynomial to be guaranteed to pass through all points then need to use n points and polynomial of degree $n-1$
`polyval(p,xp)`	inputs are `p` a vector of length $n+1$ whose elements are the coefficients in descending powers of the polynomial to be evaluated and xp the set of values to evaluate the polynomial. Returns the value of a polynomial of degree n evaluated at `xp`.
`polyder(p)`	Finds the derivative of the polynomial with the coefficients in `p`.
`spline(x,y,xp)`	inputs `x`, `y` is set of data you want to fit and `xp` the set of values to evaluate spline. Returns a vector of function values of the spline `s` corresponding to points in `xp`.
`spline(x,y)`	creates the structure of the spline using inputs `x`, `y`. Components of the structure can then be used to extract particular information like the coefficients of the spline.
`pchip(x,y,xp)`	inputs `x`, `y` is set of data you want to fit and `xp` the set of values to evaluate spline. Returns a vector of function values of the Piecewise Cubic Hermite Interpolating Polynomial `s` corresponding to points in `xp`.

Additonal Functions and Commands

The following is a list of commands used in this lab along with a brief description of what they do.

Commands that do not require inputs

Table 2

Command	Description
`find(x==n)`	finds the index of an element of `x` that is equal to `n`.
`saveas(f1,'lab09plot','epsc')`	saves the figure `f1` with the name `'lab09plot'` and `'epsc'` specifies the figure as a `.eps` file with color

Lab Exercises

Download the `lab09files.zip`. This contains `lab09.m` and the function file `evalSplineDerivs.m`. In the script file `lab09.m` you will see where to enter the specified Matlab commands to perform the required interpolation routines for the given functions. The last sections of the script file contain all the commands to generate the relevant plots. Before beginning this lab you will need to move a copy of your function file(s) that calculate the Lagrange interpolation polynomial that you wrote in Homework 4.

I. Polynomial Interpolation Routines

1. Use `polyfit` and `polyval` functions to find the function values of the interpolation polynomial.
2. Use the `spline` command to find the values of the piecewise cubic spline polynomial.
3. Now use the `pchip` command to find the values of the piecewise cubic Hermite interpolation polynomial.
4. Use your function for the Lagrange interpolating polynomial to find the function values of the Lagrange interpolation polynomial. If your function does not work, you may need to adjust how your function works. The most common issue is that students evaluate their polynomial in the script file instead of the function itself. Edit your function so that all calculations are performed in a single function file. This is to demonstrate that ideally, all the code you write should be as usable as possible.
5. Uncomment the figure commands at the bottom of the script file. These will create the figure comparing each of the polynomial interpolations. If you cannot get your Lagrange interpolation polynomial to work, comment the relevant lines of code that plot that figure. You cannot get full points on this lab unless you are able to include this figure properly!

II. Derivatives of Interpolation Polynomials

1. In the next section of the script file, use `polyder` to calculate the coefficients of the first derivative of the interpolation polynomial given by `polyfit` that you constructed in Part I.
2. Apply `polyder` again to the coefficients found in II.1 to calculate the second derivative of the interpoloting polynomial that you constructed in Part I.
3. Now store the *structure* of the cubic spline interpolation polynomial. We will need this to extract coefficient information.
4. Using the colon notation for indexing a matrix, call the column of values corresponding to each coefficient of the piecewise cubic spline using `sp_struc.coefs`. Store each of these columns in the vectors `b, c, d` respectively.
5. Use these coefficients along with `xdata, ydata, xp` to evaluate the first and second derivatives of the spline using the provided function file `evalSplineDerivs.m`. This function was written because built in Matlab functions have a problem when a coefficient is equal to zero which we have in this case. This is why it's important to have your own coding skills and not rely solely on built in functions!
6. Uncomment the given lines of code to produce the plots for Part II.

III. Figures and Tables in LaTeX

1. Click on the LaTeX Lab Report file link in the assignment on Wyocourses.
2. Upload your code and plot files to the project.
3. Add the necessary code to ensure that all of your code files are included in the report. Also, be sure to edit the `title` option so that it is appropriate for the code it is presenting.
4. Adjust the scale on the figure so that it fits on the page. This can be done using the `scale` option. This is a decimal value. Alternatively, you can use the `width` option. To auto scale to the width of the page use `width=\textwidth`. Make sure the section title appears on the same page as the plot. Both lines of code are given in the LaTeX file. See what the difference is between these options, but be sure to use one version otherwise you will display two images in your figure!
5. Replicate the following table in your report file. You create a table in LaTeX using either a `tabular` environment or an `array` environment. Tabular is good for tables that contain more text than math text. Array environments work well for tables that contain more math text than text since they can be placed inside a centered math environment like `\[ \]`. This table can then be used in your write up for problem 2 in Homework 4.

i	x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
0	x_0	$f(x_0)$			
1	x_1	$f(x_1)$	$f[x_0, x_1]$		
2	x_2	$f(x_2)$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$	
3	x_3	$f(x_3)$	$f[x_2, x_3]$	$f[x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3]$