MATH 315, Fall 2020
Homework 5

NAME: Diego Gonzalez

Important Information: This will **not** be a team assignment. You must work on this on your own, and turn in your own solutions.

You should not ask friends or anyone but me for help before Friday's lab. You will have an opportunity to work with a small group in the lab on Friday, and you can then ask that small group questions if you are having trouble. Your lab group can provide help and guidance to each other, but they **should not give** solutions to each other.

I am happy to provide individual guidance during office hours, so please do not hesitate to ask me for help.

This assignment is associated with Chapter 4, Curve Fitting – we experiment with MATLAB's `polyval` and `polyfit` functions. You should write one script m-file to perform all of the experiments and produces all of the plots outlined in the assignment.

1. Consider the function

$$f(x) = \sin(x + \sin(2 * x)) \,, \quad -\frac{\pi}{2} \le x \le \frac{3\pi}{2} \,.$$

In your script, define this function using MATLAB's function handle approach:

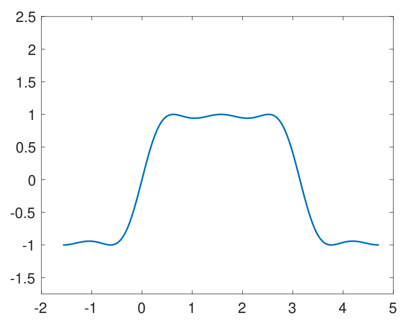
```
f = @(x) sin(x + sin(2*x));
```

Now add commands in your script to plot this function for 1000 equally spaced points in the interval $-\frac{\pi}{2} \le x \le \frac{3\pi}{2}$. You should:

- Save the 1000 points in a vector called `xvals` (you will need these later).
- Use the `axes` command to set the font size of the plot to 18 point.
- Use the `LineWidth` option to increase the line thickness by a factor of 2.
- Use the `axis` command so that the plot displays the x-axis from -2 to 5, and the y-axis from -1.75 to 2.5.

That is, the plot should look like:

Answer: I replaced the plot in the instructions with the one I created in the script called hw5part1.m. Here is the plot:

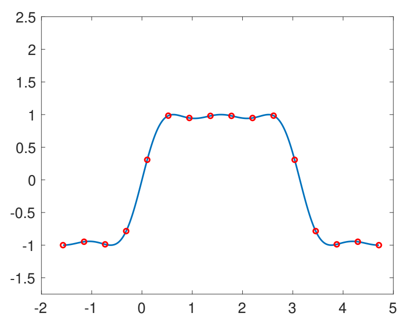


2. Next create 16 equally spaced points in the interval $-\frac{\pi}{2} \leq x \leq \frac{3\pi}{2}$, and compute the corresponding values of $f(x)$.

   - Store the x data in a vector called `x_eq`.
   - Store the corresponding $f(x)$ values in a vector called `f_eq`.
   - Plot these 16 points on top of your plot as red circles.
   - The line thickness for your red circles should be the same as the previous problem.
   - Your axis should display the same as in the previous problem.

   Your plot should now look as follows:

   Answer: I replaced the plot in the instructions with the one I created in the script called hw5part1.m. Here is the plot:

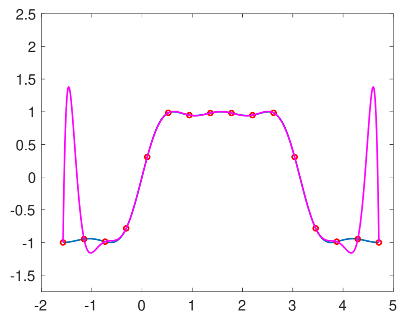


3. Use MATLAB's `polyfit` function to construct the coefficients of the interpolating polynomial through the 16 points. Then use `polyval` to evaluate the polynomial at the `xvals` from problem 1, and plot using magenta color, on the same plot as the previous problem. Make sure:

   - Your line thickness should be the same as in the previous two problems.
   - Your axis should display the same as in the previous problem.

Your plot should now look as follows:

Answer: I replaced the plot in the instructions with the one I created in the script called hw5part1.m. Here is the plot:

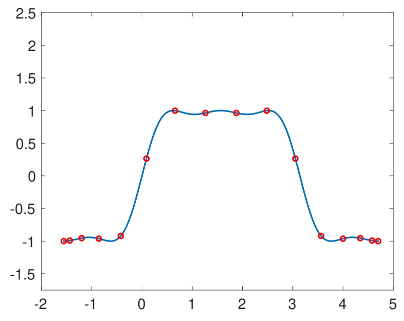


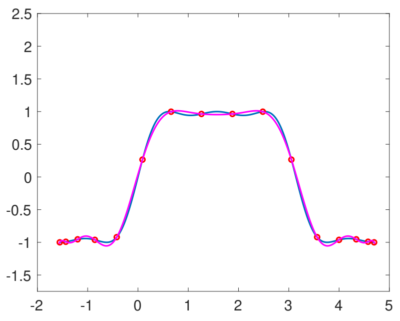
4. Now repeat exercises 1-3, but in exercises 2 and 3, instead of using 16 equally spaced points for interpolation, use the 16 Chebyshev points:

$$x_i = \frac{b+a}{2} - \frac{b-a}{2}\cos\left(\frac{2i+1}{2N+2}\pi\right) \qquad i = 0, 1, \ldots, N$$

where $N = 15$, $a = -\frac{\pi}{2}$, and $b = \frac{3\pi}{2}$.
You should use a new figure to generate this plot, and you should use the same line thickness and axis limits as in the previous problems.

Answer: I created these plots in another script called hw5part2.m. Here is the plot:





5. Suppose we are given data points (x_i, f_i), $i = 0, 1, \ldots, N$, and we want to construct and plot the Lagrange form of the polynomial that interpolates this data. That is, we want to construct $p(x)$ such that $p(x_i) = f_i$, $i = 0, 1, 2, \ldots, N$, where $p(x)$ is written in the Lagrange form:

$$p(x) = f_0 L_0(x) + f_1 L_1(x) + \cdots + f_N L_N(x) = \sum_{j=0}^{N} f_j L_j(x)$$

and

$$L_j(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_N)}{(x_j - x_0)(x_j - x_1)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_N)}$$

There is no work to be done to "construct" the polynomial, but to plot $p(x)$, we need to be able to efficiently evaluate it at x-values. This problem outlines the process.

(a) First the define weights from the denominators in $L_j(x)$:

$$w_j = \frac{1}{(x_j - x_0)(x_j - x_1)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_N)}$$

Also define a new polynomial, $L(x)$ so that

$$L_j(x) = L(x)\frac{w_j}{x - x_j}\,.$$

What is $L(x)$?

Answer: $L(x)$ should be given by the product of all $(x - x_i)$, such that:

$$L(x) = (x - x_0)(x - x_1)\cdots(x - x_{j-1})(x - x_j)(x - x_{j+1})\cdots(x - x_N)$$

If we replace this $L(x)$ into our expression above for $L_j(x)$, then we get the correct answer.

(b) Show that $w_j = 1/L'(x_j)$.

Answer: We start by taking finding $L'(x)$:

$$L'(x) = \frac{d}{dx}[(x - x_0)(x - x_1)\cdots(x - x_N)]$$

By the product rule, we get:

$$L'(x) = \frac{d}{dx}[(x-x_0)](x-x_1)\cdots(x-x_N)+$$
$$(x-x_0)\frac{d}{dx}[(x-x_1)]\cdots(x-x_N)+\cdots+$$
$$(x-x_0)(x-x_1)\cdots\frac{d}{dx}[(x-x_N)]$$

Which equals to:

$$L'(x) = [(x-x_1)(x-x_2)\cdots(x-x_N)]+$$
$$[(x-x_0)(x-x_2)\cdots(x-x_N)]+\cdots$$
$$[(x-x_0)(x-x_2)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_N)]+\cdots$$
$$[(x-x_0)(x-x_1)\cdots(x-x_{N-1})]$$

Finally, if we evaluate this expression at $x = x_j$, all of the terms becomes zero except for one:

$$L'(x_j) = [(x_j-x_0)(x_j-x_1\cdots(x_j-x_{j-1})(x_j-x_{j+1}))\cdots(x_j-x_N)]$$

Which indeed is equal to $1/w_j$.

(c) Show that the Lagrange form of $p(x)$ can be rewritten as:

$$p(x) = L(x)\sum_{j=0}^{N}\frac{w_j}{x-x_j}f_j$$

Answer: We know that:

$$p(x) = \sum_{j=0}^{N} f_j L_j(x)$$

Replacing $L_j(x)$ with the expression found in part (a), we have:

$$p(x) = \sum_{j=0}^{N} f_j L(x)\frac{w_j}{x-x_j}f_j$$

Here, we can factor $L(x)$ as it is the same for all j. Thus, we are left with:

$$p(x) = L(x)\sum_{j=0}^{N}\frac{w_j}{x-x_j}f_j$$

(d) Show that

$$1 = L(x) \sum_{j=0}^{N} \frac{w_j}{x - x_j}$$

Hint: In part (c), if $f_0 = f_1 = \cdots = f_N = 1$, what does that say about the interpolating polynomial?

Answer: If $f_0 = f_1 = \cdots = f_N = 1$, then we know that an interpolating polynomial would have to pass through all those $f_i = 1$. If we specifically want a polynomial of degree 0, then we know that a horizontal line described by $p(x) = 1$ is the only possible fit.

(e) From parts (c) and (d), for a general $p(x)$, show that we can write:

$$p(x) = \frac{p(x)}{1} = \frac{\displaystyle\sum_{j=0}^{N} \frac{w_j}{x - x_j} f_j}{\displaystyle\sum_{j=0}^{N} \frac{w_j}{x - x_j}}$$

Answer: This just follows from replacing the expressions that we in the previous two parts and cancelling $L(x)$ in both the numerator and denominator.

(f) We are now ready to write some code to evaluate $p(x)$. You will need to write two functions. The first function should compute the weights w_j given the data $x_0, x_1, \ldots, x_N$. That is, the beginning of your function should look like:

```
function w = LagrangeWeights(x_data)
%
%  This function computes weights
%                                        1
%    w_j = ----------------------------------------------------------
%          (x_j-x_0)(x_j-x_1)...(x_j-x_{j-1})(x_j-x_{j+1})...(x_j-x_N)
%
%  which are used in the Lagrange form of an interpolating polynomial.
%
%  Input:  x_data - vector of length n=N+1 containing x_i interpolation data
%
%  Output:      w - vector of length n containing the weights
%
```

The second function evaluates the interpolating polynomial using the fraction given in part(e). The beginning of this function should look like:

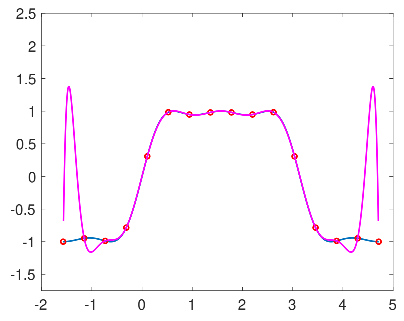
```
function pvals = LagrangeEval(w, x_data, f_data, xvals)
%
%  This function evaluates the Lagrange form of an interpolating
%  polynomoal at xvals.
%
%  Input:      w - vector containing weights, as computed by the function
%                  LagrangeWeights
%         x_data - vector of length n containing x_i interpolation data
%         f_data - vector of length n containing f_i interpolation data
%          xvals - vector containing a bunch of x-values at which we want
%                  to compute p(xvals)
%
%  Output: pvals - vector, same shape as xvals, containing p(xvals)
%
```

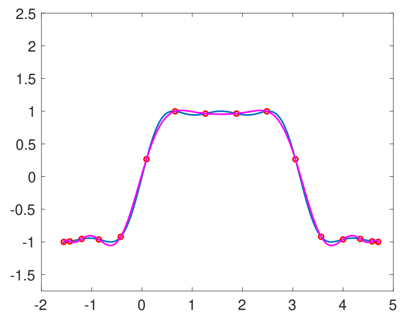
6. Using your Lagrange interpolation codes, repeat the previous experiments with equally spaced points and Chebyshev points, again with

$$f(x) = \sin(x + \sin(2 * x))\,, \quad -\frac{\pi}{2} \le x \le \frac{3\pi}{2}\,.$$

Use two new figure windows to display the two plots for this problem.

Answer: The two functions were implemented and saved with the names indicated. The test and creation of the plots was done in a separate script called hw5part3.m. Here are the plots:





7. Discuss any conclusions you can make from these experiments. For example, do you prefer equally spaced points to the Chebyshev points? Explain. Do you prefer the power series approach used by `polyfit` and `polyval` over the code you wrote for the

Lagrange form? Explain.

Answer: I think Chebyshev points are better because they minimize the error in the interpolation of a polynomial. The only way that I can think of reducing the error in an interpolation done with equally spaced points is by adding more points (thus, having a more complete curve to begin with). This, however, would mean that more computation is required. Consequently, another advantage of Chebyshev points are that they might be less computationally expensive to use in a good interpolation.
Comparing the power series approach against the one that uses Lagrange form, I prefer the power series just because we use MATLAB's built-in functions, which are already optimized. For the second approach, I am pretty sure that there are further optimizations that can be done with the code that I wrote to implement it.