1 Problem 1

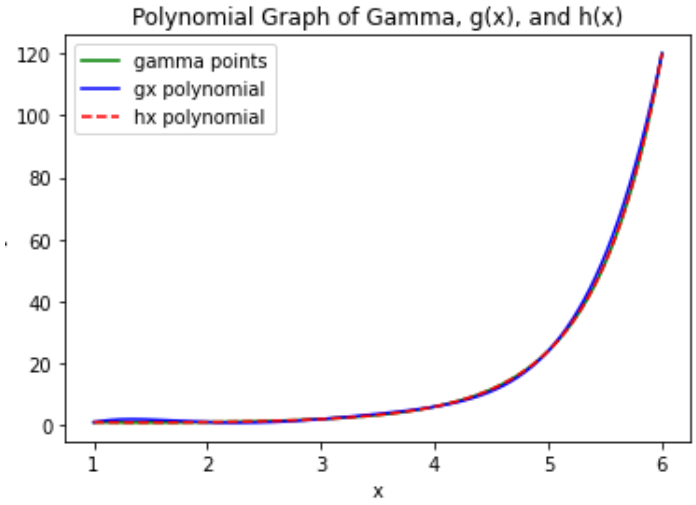
1. The polynomial interpolant and its coefficients were found using a Vandermonde matrix. I used python to solve for $Vy = b$ with numpy.Vander and numpy.linalg.solve

$$g(x) = -35 + 83.883x^1 - 70.875x^2 + 27.75x^3 - 5.125x^4 + 0.367x^5 \tag{1}$$

2. I constructed a fifth order polynomial $p(x)$ that went through the points of $(n, log(\Gamma(n)))$. I took an exponential of all the original points and then use numpy.linalg.solve to find the new coefficients.

$$h(x) = e^{1.267 - 2.187x^1 - 1.101x^2 - 0.201x^3 - 0.0217x^4 + 0.0009721x^5} \tag{2}$$

3. I constructed a plot that shows $\Gamma(x)$, $g(x)$ and $h(x)$ functions that were found above.



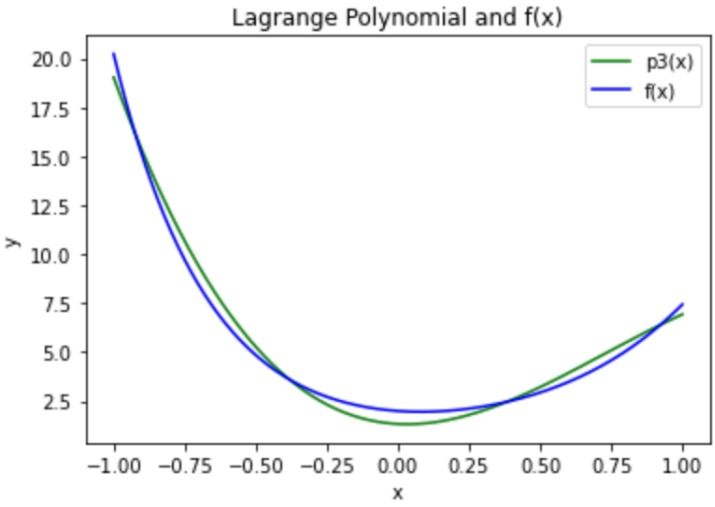
4. Calculate the maximum relative error between $\Gamma(x)$ and $g(x)$ as well as $\Gamma(x)$ and $h(x)$. I solved this by the following equation:

$$max_{x[1,6]}|\Gamma(x) - g(x)|/\Gamma(x) = 1.21084 \tag{3}$$

$$max_{x[1,6]}|\Gamma(x) - h(x)|/\Gamma(x) = 0.0080211 \tag{4}$$

2 Problem 2

1. I created a program to find the Lagrange Polynomial using the Chevyshev points over the range [-1,1]. I then plotted $f(x)$ and $p_3(x)$ on the following graph using 1000 points to smooth out the graphs.



2. I used the infinity norm to calculate $||f - p_3||_\infty$ at 1000 equally spaced points over [-1,1]. I did this by creating a new polynomial function with coefficients that came from solving a vandermonde matrix using the chevyshev points. I then took the maximum value of the difference between the function and polynomial function.

$$||f - p_3||_\infty = max_{x[-1,1]}|f(x) - p(x)| = 1.192 \tag{5}$$

3. I was trying to derive an upper bound for $||f - p_{n-1}(x)||_\infty$. The problem gives us $f(x) = e^{-3x} + e^{2x}$. The proof then goes as follows:

$$||f(x) - p_{n-1}(x)||_\infty = max_{x[-1,1]}|f(x) - p_{n-1}(x)| \tag{6}$$

$$f(x) - p_{n-1}(x) = \frac{f^n(\theta)}{n!}(x - x_1)(x - x_2)...(x - x_n) \tag{7}$$

$$f^n(\theta) = (-3)^n e^{-3\theta} + 2^n e^{2\theta} \tag{8}$$

If the max value of θ between [-1,1] the chevyshev points can be used for the interpolation.

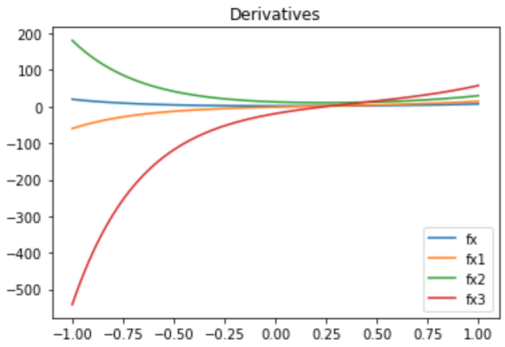
$$||(x - x_1)...(x - x_n)||_\infty = T_n(x)/2^{n-1} \tag{9}$$

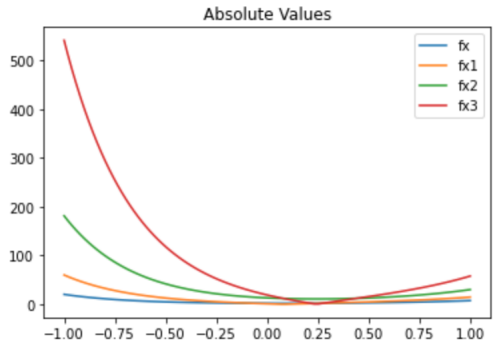
I can also replace $T_n(x)$ with the $cos(cos^{-1}x)$ since due to the chevyshev points. The equation can even further be reduced down to the value 1 because the max value of a *cos* function is 1.

This then gives the following equation.

$$||f - p_{n-1}(x)||_\infty = max_{\theta[-1,1]}|\frac{-3^n e^{-3\theta} + 2^n e^{2\theta}}{n!}|\frac{1}{2^{n-1}} \tag{10}$$

Now, there is a need to find the max value of θ for all n values. I plotted the $f(x)$, $f'(x)$, $f''(x)$, $f'''(x)$ as well as the absolute values of each polynomial. From the below plots, it can be seen that the max of each of the polynomial, as well as all polynomials to the nth derivative, occurs at the boundary point -1, which can then be plugged in.





The final solution for the upper bound of $||f - p_{n-1}(x)||_\infty$ is as follows:

$$||f - p_{n-1}(x)||_\infty = |\frac{-3^n e^3 + 2^n e^{-2}}{n!}|\frac{1}{2^{n-1}} \tag{11}$$

4. To find a cubic polynomial $||p_3^+||_\infty$ such that it gives a less error than $||f - p_3||_\infty$, I found the coefficients from the lagrange polynomial p_3 as a cubic polynomial plotted over a 1000 points. I then just manipulated my new cubic polynomial until it gave an error that was less than $||f - p_3||_\infty$.

$$max_{x[-1,1]}|f - p_3| = 1.192 \tag{12}$$

coefficients of p_3 = [1.34 -0.75 11.643 -5.325]

$$max_{x[-1,1]}|f - p_3^+| = 1.165 \tag{13}$$

coefficients of p_3^+ = [1.34 -0.712 11.643 -5.325]
This gives us the polynomial: p_3^+ = 1.34 - 0.75x + 11.643 x^2 - 5.325 x^3

3 Problem 3

1. I have shown two 2x2 invertible matrices B and C such that k(B+C) < k(B) + k(C).
$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $B^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ k(B) = $||B||||B^{-1}||$= 1*1 = 1
$C = \begin{bmatrix} 3 & 0 \\ 0 & 6 \end{bmatrix}$, $C^{-1} = \begin{bmatrix} 1/3 & 0 \\ 0 & 1/6 \end{bmatrix}$ k(C) =$||C||||C^{-1}||$= 6*1/3 = 2

$(B+C) = \begin{bmatrix} 4 & 0 \\ 0 & 7 \end{bmatrix}$, $(B+C)^{-1} = \begin{bmatrix} 1/4 & 0 \\ 0 & 1/7 \end{bmatrix}$ k(B+C) = $||B+C||||(B+C)^{-1}||$= 7*1/4 = 7/4

k(B+C) < k(B) + k(C) = 7/4 < 2 + 1 = 7/4 < 3

2. I have shown two 2x2 invertible matrices B and C such that k(B+C) > k(B) + k(C).

$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $B^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ k(B) =$||B||||B^{-1}||$ =1*1 = 1

$C = \begin{bmatrix} 3 & 0 \\ 0 & -3/4 \end{bmatrix}$, $C^{-1} = \begin{bmatrix} 1/3 & 0 \\ 0 & -4/3 \end{bmatrix}$ k(C) =$||C||||C^{-1}||$= 3*4/3 = 6

$(B+C) = \begin{bmatrix} 4 & 0 \\ 0 & 1/4 \end{bmatrix}$, $(B+C)^{-1} = \begin{bmatrix} 1/4 & 0 \\ 0 & 4 \end{bmatrix}$ k(B+C) = $||(B+C)||||(B+C)^{-1}||$ = 4 * 4 = 16

k(B+C) > k(B) + k(C) = 16 > 6 + 1 = 16 > 7

4 Problem 4

1. The goal is to find 4 polynomials with 16 coefficients that intersect points (t,x) There are 8 boundary conditions and 8 derivative conditions to help solve for the polynomials. The Cubic Spline function from scipy.interpolate was used to plot the graphs and to figure out the polynomials. I also used https://stackoverflow.com/questions/31543775/how-to-perform-cubic-spline-interpolation-in-python for help with the code.

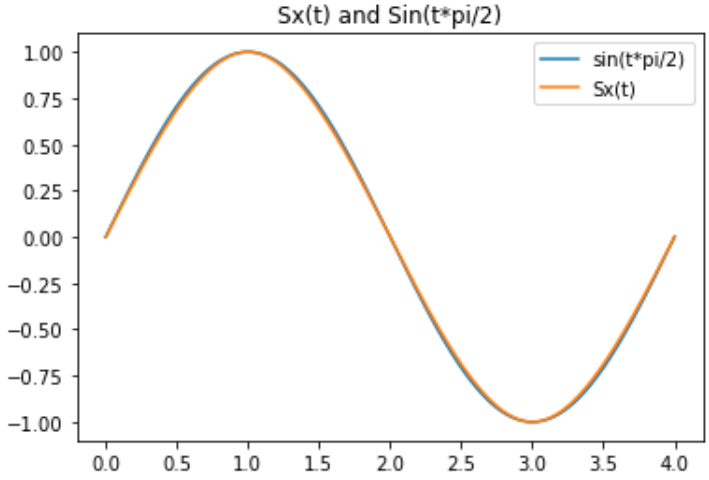
   The 8 boundary conditions:
   $S_1(0) = 0$, $S_1(1) = 1$
   $S_2(1) = 0$, $S_2(2) = 0$
   $S_3(2) = 0$, $S_3(3) = -1$
   $S_4(3) = -1$, $S_4(4) = 0$

   The 8 derivative conditions:
   $S_1(0)' = S_4(4)'$
   $S_1(0)'' = S_4(4)''$
   $S_1(1)' = S_2(1)'$
   $S_1(1)'' = S_2(1)''$
   $S_2(2)' = S_3(2)'$
   $S_2(2)'' = S_3(2)''$
   $S_3(3)' = S_4(3)'$
   $S_3(3)'' = S_4(3)''$

   The polynomials found are:
   $Sx1 = 1.5t - 0.5t^3$ for t between 0 and 1

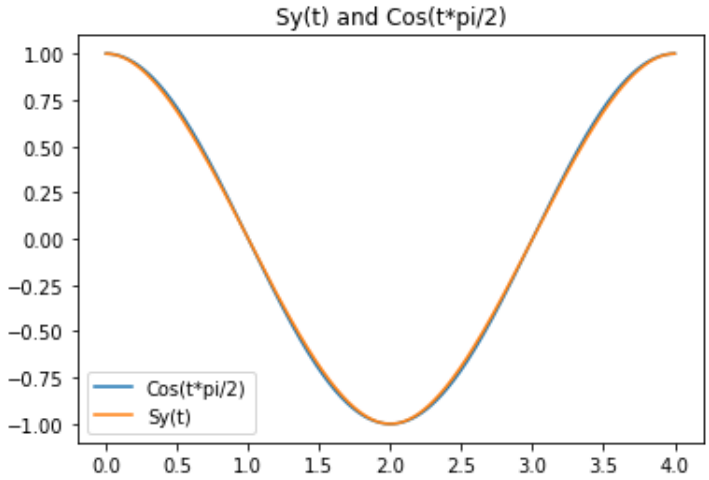
$Sx2 = 1.0 - 6.846e - 17t_1 - 1.5t_1^2 + 0.5t_1^3$ for t between 1 and 2
$Sx3 = -1.5t_2 - 4.441e - 16t_2^2 + 0.5t_2^3$ for t between 2 and 3
$Sx4 = -1.0 - 6.476e - 17t_3 + 1.5t_3^2 - 0.5t_3^3$ for t between 3 and 4

2. When you plot $sin(t\pi/2)$ against $S_x(t)$ from t = [0,4] you can see they are quite similar. I used a thousand points between [0,4] and Cubic spline to graph $S_x(t)$.

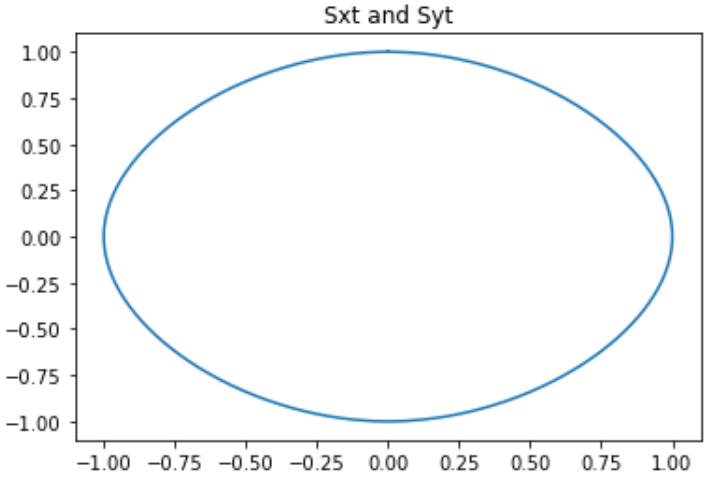


3. I did a similar process to part 4a except changed the points to be t and y[1,0,-1,0,1]. The polynomials found are:
$Sy1 = 1.0 + 1.11e - 16t - 1.5t^2 + 0.5t^3$ for t between 0 and 1
$Sy2 = -1.5t_1) + 2.22e - 16t_1^2 + 0.5t_1^3$ for t between 1 and 2
$Sy3 = -1.0 + 1.11(e - 16)t_2 + 1.5t_2^2 - 0.5t_2^3$ for t between 2 and 3
$Sy4 = 1.5t_3 + 2.226(e - 16)t_3^2 + 0.5t_3^3$ for t between 3 and 4

I also plotted the polynomial $Sy_(t)$ against $cos(t/\pi/2)$ using the same methodology as 4b.



4. Next, I plotted $Sy(t)$ against $S_x(t)$.



I then integrated under the curves to get the area of the two graphs. The area of the two graphs is 3.0499999712640764. This is close to the area when using π (3.14) in the formula $A = \pi r^2$ with r = 1.

5 Problem 5

Note: My jupyter notebook and pdf will be slightly different for the io.imread() functions because I had to go through FAS to print the pdf. I needed to reupload the jupyter notebook and the photos so they can read because they weren't in that environment.

1. In this problem, I read in four different images, which consisted of the regular photo and then the regular photo in the color red, blue, and green. I took the regular photo

in as a (400, 300, 3) matrix, with red pixels, green pixels, and blue pixels each as a layer in the image as the third dimension. I had to reshape each into an individual matrix of a size (120000, 1). Next, I took in each of the other photos and reshaped them into a (120000,3) matrix. From there, I created a (120000,9) by concatenating those three matrices next to each other. Now, I can perform a Linear Regression for each of the colored picture matrices (120000, 1) with the (120000,9) concatenated matrix. Lastly, I can recreate the image but using the Linear Regression model to predict each of the channels and concatenating them again.

$$Fb = \begin{bmatrix} 0.6 & -0.84 & -2.31 \\ -0.35 & 0.044 & -1.5 \\ -0.38 & -0.434 & 1.00 \end{bmatrix}$$

$$Fc = \begin{bmatrix} 1.98 & 0.94 & -1.49 \\ 1.2 & 2.04 & -2.02 \\ 0.87 & -0.27 & -0.68 \end{bmatrix}$$

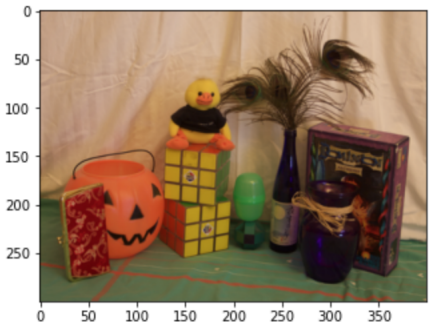
$$Fd = \begin{bmatrix} -0.31 & 0.18 & 0.33 \\ -0.47 & 0.55 & 0.304 \\ -0.13 & 0.49 & 1.03] \end{bmatrix}$$

The intercepts $p_{const} = \begin{bmatrix} 0.062 \\ 0.046 \\ -0.0062 \end{bmatrix}$

We then derive S from the following equation:

$$S = \frac{1}{MN} \sum_{k=0}^{MN-1} ||Fb * p_k^b + Fc * p_k^c + Fd * p_k^c||_2^2 = 0.0511 \tag{14}$$

Here is the regular image:



Here is the reconstructed image:

2. I used the linear model from 5b to created a predicted image of Professor Rycroft's wife's bears. I followed a similar methodology of shaping the pixels and channels in order to recreate the matrices. I also solved for the mean squared error using:

$$T = \frac{1}{MN} \sum_{k=0}^{MN-1} ||p_k^A - p_k^{A*}|| \ = 0.0054 \tag{15}$$

Here is the regular image:

Here is the reconstructed image: