EGR 141: Polynomials, Curve Fitting, and Symbolic Computations

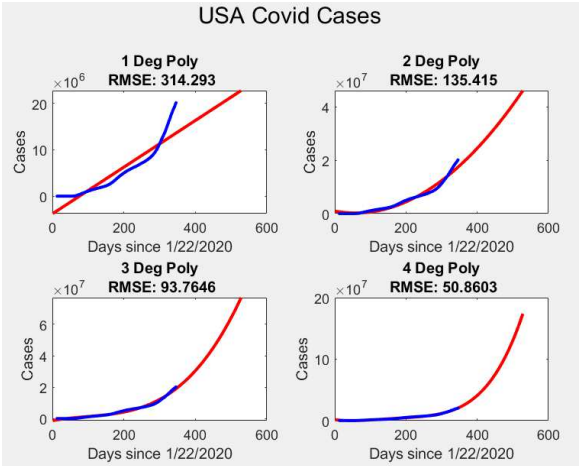
Summary: The goal of this lab is to help figure learn how fit data with functions and perform symbolic mathematics. Each problem should use the concepts and functions (or related functions) discussed in class. If you find a MATLAB function that seems to directly solve the problem, check with me before using it.

- Each of the following problems should have a script and, possibly, a function associated with them.
- For each problem, the script file should be called something appropriate, such as *Lab8_1_yourName.m*.
- Include any functions that you needed to create in order to complete the problem. Name them whatever is indicated in the problem.
- Inside your script, solve each of the given problems. In between each problem, type *pause;* Clearly indicate where the code for each problem begins by using a comment block. Start each new problem with a *clear .*
- If my example output "lines up nicely" then your output should as well.
- All output statements involving variables should output variable values, not pre-computed constants. For example, if I ask you to output $r/2$ when $r = 3$, then you should set r to be three then output as *fprintf(' r/2 = %f ',r/2);* and not *fprintf('r/2 =1.5')* or *fprintf(r/2 = %f ',3/2)*.
- Note that example output for each problem is not necessarily correct output (I intentionally do different output than what you will do).
- Each computation that can be done symbolically should be done so.

1. Revisit your Covid dataset/country from Project 1. If your professor asked you to correct anything with your dataset (or use a different dataset), please do so for this problem. Instead of using your own regression functions, we will use MATLAB's built in ones. We define the *root mean squared error* between a polynomial, $p(x)$, and a data set $(x_i, y_i)_{i=1}^n$ to be:

$$e = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - p(x_i))^2}$$

Using subplot, create four plots containing the first through fourth order polynomial least-square fits. On each figure, include the given data as single points along with your polynomial graphed from the starting day of your dataset until 7/4/2021. Label and title as in the Project 1, but include the error of each fit as part of your title.



Output each polynomial's estimate for the number of Covid cases on 7/4/2021.

```
Lab 8 - New Covid Est
Estimate of cases on 7/4/2021 for
 Poly of order 1 is 22745203 cases
Estimate of cases on 7/4/2021 for
 Poly of order 2 is 46098015 cases
Estimate of cases on 7/4/2021 for
 Poly of order 3 is 77009100 cases
Estimate of cases on 7/4/2021 for
 Poly of order 4 is 173691031 cases
```

Note: Depending on your dataset, you might get a warning about badly condition data. Ignore it (it does create a big mathematical issue, but we're going to smile and nod).

2. Create a MATLAB function $r = nm(f,x0,tol)$ which applies Newton's method to an anonymous function $f(x)$.

   - Look up Newton's Method (Wikipedia will work fine)
   - Start with a guess of $x0$ (input into the function)
   - Run until the absolute value of$f(x)$ at the current iteration is less than tol (just like Bisection method back in Lab 6)
   - Automatically compute the derivative of $f(x)$ using symbolic mathematics for use in the algorithm
   - Output the final estimate for the root as the variable r

   In your calling script file, test by finding the roots of each function

   (a) $f(x) = x^2 - 7$
   (b) $g(x) = \sin x - \frac{1}{2}$
   (c) $h(x) = \ln x - 1$

   For each algorithm, use a reasonable initial guess that is not already the solution to the problem (close to zero should work for all) and a tolerance of 10^{-7}. Output the final result nicely to the screen using at least 7 decimal places.

```
Lab 8 - Newton's Method
 f(x)=cos(x)+x has a root at approximately x=-0.7390851
```

3. Create a function, named something appropriate, which finds all local min/max of a function using the *first derivative test*. The function should

   - takes in a single function handle/anonymous function
   - return two vectors, $minX$ and $maxX$ which represent the x values of each local minimum and local maximum of the passed-in function. Note that $minX$ and/or $maxX$ could be empty vectors.

   In your script, test on each of the following

   (a) $f(x) = x^5 - \frac{19}{2}x^4 + \frac{4}{3}x^3 + 124x^2 + 96x$
   (b) $g(x) = -e^x + 3x$
   (c) $h(x) = -4e^x + xe^x + 6x - x^2$

   Print out your local min values first (sorted low to high) then the local max values (sorted low to high).

```
Lab 8 - Min/Max First Deriv Test
 For f(x) = (x^2-1)*exp(-x^2)
  Local Min: x = 0
  Local Max: x = -1.41421, 1.4142
```

4. Recall that, back in Project 1, we approximated integrals using Riemann sums (based on rectangles), Trapezoidal rule (based on trapezoids) or Simpson's rule (based on quadratics). We now consider an interpolation-based approach for approximating

$$\int_a^b f(x)\,dx$$

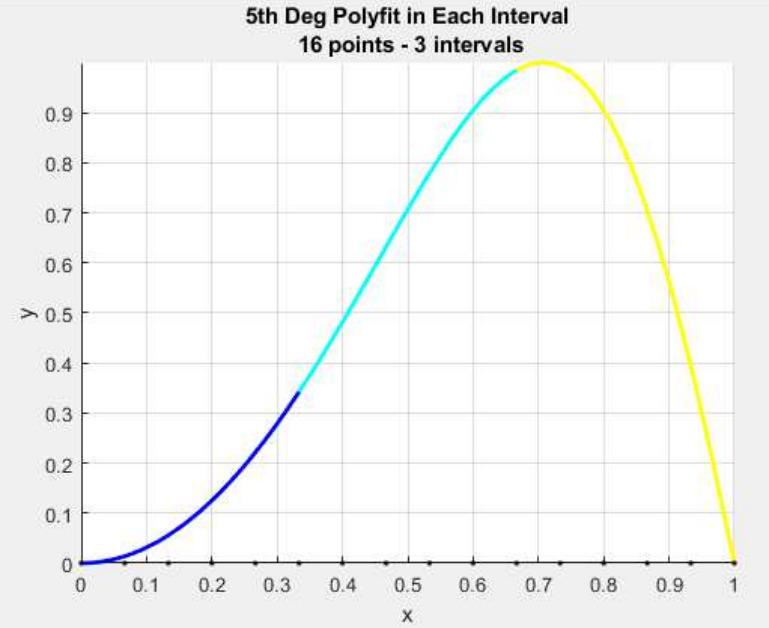
where we

- Partition the interval $[a, b]$ into $N + 1$ points.
- Break up integral bounds into intervals containing $M + 1$ points each

$$\int_a^b f(x)\,dx = \int_{a=x_1}^{x_{M+1}} f(x)\,dx + \int_{x_{M+1}}^{x_{2M+1}} f(x)\,dx + \cdots + \int_{x_{(N-1)M+1}}^{b=x_{N+1}} f(x)\,dx.$$

- Note that N must be divisible by M.
- Note that "middle interval points" will overlap.
- For each "sub-integral" (or, sub-interval!) fit $f(x)$ with a M^{th} degree polynomial (this is exact interpolation, since there are $M + 1$ points on each interval)
- Determine the value of the integral of this polynomial over the region.
- Add each integral together to obtain an esimate of $\int_a^b f(x)\,dx$.

For example, we have $N = 15$, $M = 5$ and the inverval $[0, 1]$. We split $[0, 1]$ into $N + 1 = 16$ points ($N = 15$ intervals) and use $M + 1 = 6$ points in each interval (thus 3 intervals since $N/M = 3$). Since we have $M + 1 = 6$ points in each interval, we fit each with a 5th degree polynomial.



This would represent

$$\int_0^1 \sin \pi x^2 \, dx = \int_0^{1/3} \sin \pi x^2 \, dx + \int_{1/3}^{2/3} \sin \pi x^2 \, dx + \int_{2/3}^1 \sin \pi x^2 \, dx,$$

where each integral computed using the integral of an interpolated 5th degree polynomial.

Create a function that takes in a function $f(x)$, along with a, b, N and M as defined above. The estimate of $\int_a^b f(x)\,dx$ using the above algorithm should be returned. If M does not divide N exactly, display a message to the screen and return a NaN . Test your routine on each

(a) $\int_{-2}^1 e^{-x^2} \cos 2\pi x \, dx$, with 37 points and 7 points in each "sub-integral" (so $N = 36$ and $M = 6$).

(b) $\int_0^1 \sqrt{1 - x^4}\, dx$, with 46 points and 6 points in each "sub-integral."

(c) $\int_{-1}^1 e^{e^x} \, dx$, with 61 points and 5 points in each "sub-integral."

Print at least 8 decimal places in your solution.

```
Integral of sin(pi*x^2) over
 0 to 1 is approx 0.50485363
```