Data Modeling Lab

In this little, in class project, you're going to generate two types of functions to approximate the US Population throughout the 19th century:

- Using a spline and
- Using a linear, least square approximation

Of course, we'll use a few libraries:

```
In [ ]: %matplotlib inline
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        from numpy.linalg import lstsq
        from scipy.interpolate import interp1d
```

Here's the data:

```
In [ ]: pop_df = pd.read_csv('https://www.marksmath.org/data/us_population.csv')
        pop_df = pop_df.loc[pop_df.loc[:,'year']<1900]
        year = pop_df.loc[:,'year']
        pop = pop_df.loc[:,'pop']
```

If you don't have internet, you can execute the following:

```
In [ ]: year = np.array([
          1790, 1800, 1810, 1820, 1830, 1840, 1850, 1860, 1870, 1880, 1890
        ])
        pop = np.array([
          3929214, 5236631, 7239881, 9638453, 12866020, 17069453,
          23191876, 31443321, 38558371, 49371340, 62979766
        ])
```

Either way, the data should look like so:

```
In [ ]: plt.plot(year,pop,'ok')
```

Exercise 1 Use SciPy's `interp1d` command, as described in our recent class notebook, to find and plot a cubic spline that passes through this data. Let's denote this function by f_3 and let's denote the piecewise linear spline that interpolates the function by f_1. Looking at the graph you just made of f_3 passing through the data, do you think that

$$f_1(1855) < f_3(1855) \quad \text{or} \quad f_1(1855) > f_3(1855)?$$

Then, compute $f_1(1855)$ and $f_3(1855)$ to check your answer.

0.1 Least squares

Often, we might suspect that a particular type of function can be used to model given data. Populations, for example, are well known to grow approximately exponentially when the population is not too big. Thus, it might make sense to model the data above with an exponential funciton. We write:

$$y_i \sim ae^{rx_i}.$$

Since, the model is not likely to be perfect, we expect that we won't be able to fit the data exactly; thus, we minimize the residuals in the least square sense. Thus, we might try to minimize

$$\sum_{i=1}^{n}(ae^{rx_i} - y_i)^2.$$

Unfortunately, the resulting gradient system is non-linear in the parameters a and r. While there are techniques to solve these types of systems, linear solvers tend to be faster, simpler, and more reliable. For exponential models, we can translate to the equivalent model

$$\log(y_i) \sim \log(a) + rx_i.$$

The resulting gradient system is linear in $\log(a)$ and r and it's easy to translate from there to the exponential model of the original data.

Exercise 2 Use the scheme described above to find an exponential model for the US Population during the 19$^{\text{th}}$ century. Plot your model along with the data. What does the model predict for the population in 1855?

I'll get you started! The following code solves the linear system that arises by applying the logarithm to both sides:

```
In [ ]: log_pop = np.log(pop)
        A = np.vstack([year**1,year**0]).transpose()
        coeffs = np.linalg.lstsq(A, log_pop)[0]
        coeffs
```

Now, I guess you'll need to interpret that to find your exponential model. Then you'll define a function g that looks something like so:

```
In [ ]: a = ...
        r = ...
        def g(year): return a*np.exp(r*year)
```

Then you'll use that function to answer the questions.