# HW 9 - Curve Fitting

Due April 15th, 2025 at 11:59pm

Curve fitting is a super important skill. These problems should help you get more comfortable with:

- curve fitting (using `scipy.optimize.curve fit`, `pandas`, and `matplotlib`) (you'll need this for your final project!)
- covariance matrices and error
- data cleaning, processing, and visualization

## 1 Curve Fitting Guided Problem

The following problem will walk you through how to fit a tricky curve to a set of data.

This problem looks super long, but it's not :)

### 1.1

Use pandas to do the following:

1. Read in the file "GlobalLandTemperaturesByState.csv".
2. Filter the table to include only the columns for the date, temperature, and state.
3. Filter the table to include only years after 2000.
4. Filter the table to include only the rows corresponding to Wyoming, Nebraska, or South Dakota. *Check: your table should be 495 rows and 3 columns.*

### 1.2

Modify the table such that it contains the **average** temperature over all three states for each date. It should have two columns: date and average temperature.

- Look into the `pandas` function `groupby`.

## 1.3

Use `matplotlib.pyplot` to plot the data from the table you created above. You can pass `pandas` columns directly into `matplotlib.pyplot` without needing to turn them into arrays.

1. Date on the x axis, average temperature on the y axis.
2. Label the axis and give the graph a title.

Now that the data is imported and plotted so you have an idea of what it looks like, let's get into the curve fitting.

## 1.4

The function `scipy.optimize`, unsurprisingly, can only do math with numbers. The date column of the table is currently composed of strings.

- Fix this and convert the string date into numerical values however you see fit, and make it a column in the dataframe. Your numerical values should capture all parts of the date provided in the string (don't use just the year, etc).

## 1.5

The function `scipy.optimize` requires: a model equation, and an initial guess of parameters. For this section:

1. Define an appropriate model equation. Use a generic form like $mx + b$: there should be four parameters.
2. Make an initial guess at the parameters and save them in an array.
    - This part is really important. A dataset with a non polynomial pattern was chosen for a reason: your initial guesses matter, particularly the period.
    - If you're stuck, eyeball the length of the period (it should make physical sense, and remember that the units are in years) and keep this in mind: $\cos(2x)$ means the function covers two periods in the space of $2\pi.2 * \text{period} = 2\pi$, so each period is $\pi$ long.

## 1.6

Run `scipy.optimize`'s curve fit function! Remember that it outputs a tuple containing two arrays: the parameter array and the covariance matrix.

- If while attempting this, you get one of the following errors:
    - Something about maximum depth

  - Something about not being able to estimate the covariance
  - A line that does not fit the data at all

  You may need to re-examine your guesses for the initial parameters (particularly the period). This is why plotting the data before fitting it is critical.

### 1.7

Do the following:

1. Re-plot the data.
2. Plot the line outputted by curve fitter on the same graph as the data.
3. Make sure they mostly match up!

### 1.8

Do they match? If so, you made it! Your curve fitting code looks great. Non polynomial functions are tricky to fit, so congratulations on curve-fitting sinusoidal temperature variations! One last step (okay, two):

- Use the covariance matrix to calculate the errors for each parameter. Recall that the errors are located down the diagonal of the matrix.
  - E.g: Parameter 1's error is the square root of location [0,0] in the matrix, 2's error is at location [1,1], etc.

### 1.9

The final step! Print out your results!

1. Print out each parameter AND its corresponding error with format: parameter $\pm$ error.
2. Print out the final equation!

Congrats :) you did it!

### 1.10

Goal: complete a linear regression fit to a dataset and evaluate the appropriateness of the model. Do the following:

1. Read in one of the two .dat files: 'global CCl4 MM.dat' or 'global SF6 MM.dat'. Don't modify the file directly. Use `astropy.table`'s Table library.

2. Convert the astropy Table into a pandas dataframe. Include columns: date, global mean concentration, and global mean concentration sd.

3. Plot the data from the file with error. Matplotlib has a function that can do this - look it up!

    - Include axis labels and a title.

4. Fit an appropriate linear model. Use whatever package makes sense to you (`scipy.optimize`, `numpy.polyfit`, etc)

5. Calculate the reduced chi-squared value.

6. Print out the parameters, their errors, the final equation, and the reduced chi-squared value.

7. Write if a linear model seems appropriate. A residual plot is encouraged but it is not required.

8. Bonus (0 points): Write a sentence or two about the molecule you're looking at and how that background may relate to the trend you're seeing.