HW 9 - Scipy

Due April 8th, 2026 at 11:59 PM

Overview

In this homework assignment, you'll practice using `scipy.optimize.curve_fit` to model noisy data, calculate the goodness of fit using reduced chi-squared and analyze data involving Kepler's Third Law.

1 Branching with GitHub

Before starting this assignment, create a new Git branch and folder for this assignment:

Steps:

1. Create a branch called `homework9` in your `yourname` repository.

2. Inside that branch, create a folder named `homework9`.

3. Take a screenshot showing that you:
   - Created that branch
   - Deleted all the current files/folders
   - Committed and push your changes to GitHub

4. Save the screenshot as `hw9_branch` inside your `homework9` folder.

2 Curve Fit

A **scientific model** is a simplified representation of a system. For example, how the atom is often visualized with a cluster of protons and neutrons surrounded by orbiting electrons. While this model may not be perfectly accurate, they allow us to understand real-world behavior.

When taking real-world measurements, it's common to use **mathematical models** (functions/equations) to explain current and predict future data. However, no collected data comes without noise. **Noise** is random variation in a measurement which was intended to be captured. Let's explore how noise affects the process of fitting a model.

2.1 Collect the Data

Pull the latest version of the `main` branch of the `course_assignment` repository. You will find a folder called `homework9/` which contains a data files called:

- `noisy_curvefit_data.csv`
- `solar_system.csv`

In this problem, we will only be working with `noisy_curvefit_data.csv`. It contains synthetic data with different amounts of noise.

Steps:

1. Create a new Jupyter Notebook in your `homework9/` folder. Name it `homework9.ipynb`.

2. Import the following libraries:
   - `numpy as np`
   - `pandas as pd`
   - `matplotlib.pyplot as plt`
   - `from scipy.optimize import curve_fit`

3. Load the `noisy_curvefit_data.csv` file as a `pandas` Dataframe.

4. Inspect the DataFrame:
   - What is the shape of your dataset?
   - How many rows and columns are there?
   - Print the first five and last five rows.
   - What are the column names?

5. Plot the `x` vs. `y_noise_1` column:
   - Use `plt.scatter()` to show the noisy data as gray dots.
   - Add axis labels, a title and a legend.

Note: The column `y_noise_1` corresponds to the first noise level. So, `y_noise_2` corresponds to the second, and so on.

2.2 Fit the Data

When you analyze data, sometimes you know exactly what kind of model describes your data. Other times, you don't. In this case, we will give you the model so you can just focus on the process of fitting:

$$y = A \cdot e^{-k \cdot x}$$

You'll also calculate **the reduced chi-squared statistic**, which tells you how well your model fits the data:

$$\chi^2_{\text{reduced}} = \frac{\sum_{i=1}^{n} \left(y_{\text{noise},i} - f(x_i; A_{\text{fit}}, k_{\text{fit}})\right)^2}{n - p}$$

A value close to **one** is a good fit. That's all you need to know for this problem but you are welcome to do more research about reduced chi-squared.

Steps:

1. Define the model:

```
def model(x, A, k):
    return A * np.exp(-k * x)
```

2. Fit the model to the `y_noise_1` data:
   - Use and initial guess: `p0 = [1, 1]`
   - Call `curve_fit`:

```
popt, pcov = curve_fit(model, x, y_noise_1, p0=p0)
```

3. Extract the best-fit parameters:

```
A_fit, k_fit = popt
```

4. Print the fitted model with two decimal places. Example:

```
f"Fit: y = {A_fit:.2f} * e^(-{k_fit:.2f} * x)"
```

5. Calculate the reduced chi-squared:

```
residuals = y_noise_1 - model(x, A_fit, k_fit)
chi2 = np.sum(residuals**2)
dof = len(x) - len(popt)
chi2_reduced = chi2 / dof
```

6. Plot the result:
   - Scatter: noisy data
   - Red line: fitted model
   - Black dotted line: true model (use `model(x, 5, 0.8)`)
     Hint: Use `linestyle="--"` as an argument.
   - Include the noise level (i.e., 1 since y_noise_1) and reduced chi-squared in the title. Round the reduced chi-squared to two decimal places.

- Add axis labels and a legend

2.3 Compare Different Noise Levels

To evaluate the rest of the noise levels, let's automate the process since we will be performing the same fit over and over again.

Steps:

1. Write a function called `fit_model(x, y)` that:
   - Fits the model using `curve_fit`
   - Returns: best-fit parameters and reduced chi-squared
2. Create a 2x2 subplot grid.
3. Loop through `y_noise_2`, `y_noise_3`, `y_noise_4`, `y_noise_5`. In each subplot:
   Hint: Use a `for` loop with `enumerate()` and/or `zip()`.
   - Plot the noisy data (gray dots)
   - Plot the fitted model (red line)
   - Plot the true model (black dotted line)
   - Add title with noise level and reduced chi-squared
   - Add axis labels and a legend
4. Answer the following questions:
   - What happens to the **red and black dotted lines** as you increase the noise level? Why?
   - What happens to the **reduced chi-squared** as you increase the noise level? Why?

2.4 Save Your Results

Steps:

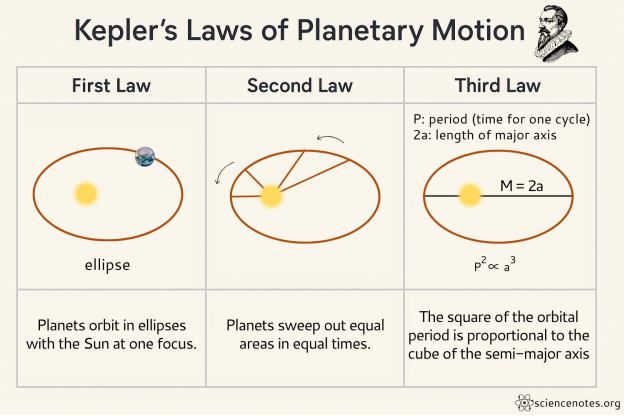
1. Create a new `pandas` DataFrame with:
   - Noise level
   - A fit
   - k fit
   - Reduced chi-squared

   There should be **five** rows and **four** columns.
2. Save the DataFrame as `noisy_parameters.csv`:

```
df.to_csv("noisy_parameters.csv", index=False)
```

3. Save this file in your `homework9/` folder in your `homework9` branch.

3 Kepler's Third Law



Kepler's Third Law relates a planet's orbital period to the size of its orbit:

$$T^2 = a^3$$

Where T is the orbital period (how long it takes an object to orbit) and a is the semi-major axis (half the length of the longest diameter of an ellipse).

The **perihelion** is the point in the object's orbit when it is *closet* to the Sun. The **aphelion** is the point when it is *farthest* from the Sun. The semi-major axis is half of the longest diameter. Therefore:

$$\text{Semi-major axis} = \frac{\text{Perihelion} + \text{Aphelion}}{2}$$

Kepler's Third Law helps us understand planetary motion. The dataset you will analyze contains information from our solar system which you will use to test Kepler's Third Law.

Steps:

1. Create a new Jupyter Notebook in your `homework9/` folder. Name it `kepler.ipynb`..

2. Import necessary libraries.

3. Import the `solar_system.csv` file as a `pandas` DataFrame.

4. Inspect the data. Answer the following questions:
   - What is the shape of your dataset?

- How many rows and columns are there?
- Print the first five and last five rows.
- Print the column names.
- Print the rows names.

5. Extract the following rows as variables:
   - Orbital period
   - Perihelion
   - Aphelion

   Hint: Use `.iloc` to extract a row and `.astype(float)` to convert strings to floats.

6. Define **perihelion**, **aphelion** and **semi-major axis** in your own words.

7. Compute the semi-major axis:

   ```
   semi_major_axis = (perihelion + aphelion) / 2
   ```

8. Add the semi-major axis as a new row to your DataFrame. Don't forget units!
   Hint: Create a `pandas` Series and use `pd.concat()` to add it to your DataFrame.

9. Print the new shape of your DataFrame. How many rows? How many columns?

10. Plot the `semi_major_axis` vs. `orbital_period` using `plt.scatter()`. Include:
    - Axis labels with units (don't forget units!)
    - Title and legend

11. Define the model function:

    ```
    def keplers_third_law(a, m):
        return m * a ** 1.5
    ```

12. Use `curve_fit` to fit the model to the data.

13. Extract the best-fit parameter.

14. Print the fitted model with two decimal places.

15. Calculate the reduced chi-squared statistic.

16. Plot the result, including:

- Scatter points
- Fitted model
- Chi-squared in the title
- Labels and legend (don't forget units!)

17. Save the new DataFrame as `kepler.csv`:

18. Save this file in your `homework9/` folder in your `homework9` branch.

4 Submitting Your Homework

Make sure before you submit your repository that you have fully run your Jupyter Notebooks, we want to be able to see the output when we are grading!

Steps:

1. In your `yourname/` repository, run:

```
git add .
git commit -m "done with hw9"
git push origin homework9
```

2. Take a screenshot of the terminal output.
3. Name the screenshot: `hw9_changes`.
4. Place it inside your `homework9/` folder.

5. Ensure all screenshots are saved correctly and your code runs without errors. Your `homework9/` folder should now contain:

```
homework9/
|--- kepler.csv
|--- noisy_parameters.csv
|--- homework9.ipynb
|--- kepler.ipynb
|--- hw9_branch.png
|--- hw9_changes.png
```

6. Go to Gradescope and find the **Homework 9: Scipy** assignment.
7. Select the option to upload a GitHub repository.
8. Make sure to upload your `homework9` branch.
9. Submit your `yourname` repository.

Great job!