Description of Python Code.

Parag M Ahmedabadi

October 11, 2023

Contents

1 Fit functions

`dosfit_func.py`: The provided code defines a collection of mathematical functions in Python using the NumPy library. These functions include various exponential functions, their derivatives, and other mathematical expressions. The functions are parameterized with coefficients such as a, b, c, d, e, f, and g, and they operate on variables T and t. The code is organized into separate functions for clarity and maintainability. Additionally, there is a list `fx_list` that contains references to some of these functions. This code serves as a mathematical toolkit for evaluating different mathematical equations involving exponentials and other mathematical operations.

1.1 Functions

The following functions are used for fitting as well as for derivative calculations. Here, x is $1/T$ and y is t in seconds; normalization of variables is done elsewhere.

1.1.1 Function F0 (`F0`)

Input: X, a, b, c, e, f **Output**: value

Function F0 computes a mathematical expression using the given inputs. It takes two variables, T and t from input X, and five parameters (a, b, c, e, f). The function evaluates the expression using these inputs, which involves exponential and polynomial terms. The result is stored in the variable 'value,' which is returned as the output.

1.1.2 Function dT_F0 (`dT_F0`)

Input: X, a, b, c, e, f **Output**: value

Function dT_F0 computes the derivative of the function F0 with respect to variable T. It takes the same input variables as F0 and computes the derivative based on these inputs. The result is stored in the variable 'value' and returned as the output.

1.1.3 Function dt__F0 (`dt__F0`)

Input: X, a, b, c, e, f **Output**: value

Function dt__F0 computes the derivative of the function F0 with respect to variable t. Like the previous function, it takes the same input variables and calculates the derivative based on these inputs. The result is stored in the variable 'value' and returned as the output.

1.1.4 Function F1 (`F1`)

Input: X, a, b, c, d, e, f **Output**: value

Function F1 computes a mathematical expression using the given inputs. It takes two variables, T and t from input X, and six parameters (a, b, c, d, e, f). The function evaluates the expression using these inputs, which involves exponential and polynomial terms. The result is stored in the variable 'value' and returned as the output.

1.1.5 Function F2 (F2)

Input: X, a, b, c, d, e, f, g **Output**: value

Function F2 computes a mathematical expression using the given inputs. It takes two variables, T and t from input X, and seven parameters (a, b, c, d, e, f, g). The function evaluates the expression using these inputs, which involves exponential and polynomial terms. The result is stored in the variable 'value' and returned as the output.

1.1.6 Function F3 (F3)

Input: X, a, b, c, d, e, f, g **Output**: value

Function F3 computes a mathematical expression using the given inputs. It takes two variables, T and t from input X, and seven parameters (a, b, c, d, e, f, g). The function evaluates the expression using these inputs, which involves exponential and polynomial terms. The result is stored in the variable 'value' and returned as the output.

1.1.7 Function dT_F1 (dT_F1)

Input: X, a, b, c, d, e, f **Output**: value

Function dT_F1 computes the derivative of the function F1 with respect to variable T. It takes the same input variables as F1 and computes the derivative based on these inputs. The result is stored in the variable 'value' and returned as the output.

1.1.8 Function dt__F1 (dt__F1)

Input: X, a, b, c, d, e, f **Output**: value

Function dt__F1 computes the derivative of the function F1 with respect to variable t. Like the previous function, it takes the same input variables and calculates the derivative based on these inputs. The result is stored in the variable 'value' and returned as the output.

1.1.9 Function G0 (G0)

Input: X, A, a, b, c, d, e, f **Output**: value

Function G0 computes a mathematical expression involving several parameters (A, a, b, c, d, e, f) and variables T and t from input X. It calculates a value based on an exponential and polynomial expression, and then normalizes it using a specific formula. The normalized value is returned as the output.

1.1.10 Function G1 (G1)

Input: X, A, a, b, c, k, j **Output**: value

Function G1 computes a mathematical expression involving several parameters (A, a, b, c, k, j) and variables T and t from input X. It calculates a value based on exponential expressions with different coefficients. The final value is returned as the output.

1.1.11 Function G2 (G2)

Input: X, A, w, xc, v, yc, u, k **Output**: z

Function G2 computes a complex expression with multiple parameters (A, w, xc, v, yc, u, k) and variables T and t from input X. It involves several terms with specific mathematical functions. The final result is stored in the variable 'z' and returned as the output.

1.1.12 fx_list

Description: `fx_list` is a Python list that contains references to the previously defined functions, such as F0, F1, G0, and G1. It can be used to select and call specific functions for further calculations.

These functions are used for various mathematical calculations and modeling tasks, and they play a crucial role in the overall analysis performed by the code.

2 DOS Model

`dosmodel.py`: This code is a Python program designed for curve fitting and plotting in the context of Density of States (DOS) modeling. It leverages libraries like NumPy, SciPy, and Matplotlib for data manipulation, fitting functions to experimental data, and generating various plots. The main functionalities of this code include loading and normalizing data, fitting curves to the normalized data, smoothing the fitted data, and creating 2D and 3D visualizations. It can calculate derivatives, integrate curves, and determine the ranges of specific DOS values. The code is particularly useful for materials science research involving DOS modeling.

2.1 Import Statements

In this section, various Python libraries and modules are imported to provide the necessary tools for data analysis, plotting, and mathematical operations. Notable libraries include `matplotlib`, `math`, and several functions from other files like `tools.py` and `dos_fitfunc.py`.

2.2 Custom Style Sheet Definition

This section defines a custom style sheet for Matplotlib. It specifies various visual properties like font size, label sizes, and legend font size to be used in plots created with Matplotlib.

2.3 Results Folder Path Definition

The `dir_path` variable is set to a folder path where the results of the analysis will be saved.

2.4 Class Definition (DOSModel)

This is the core of the code, where a Python class named `DOSModel` is defined. The class encapsulates methods and attributes related to a specific data analysis and modeling task. Here's a breakdown of the class methods and attributes:

2.4.1 Initialization Method (`__init__`)

This method initializes an instance of the `DOSModel` class. It takes one argument, `xfile`, which is the path to a CSV file containing data. The method initializes various data and parameters needed for further analysis.

2.4.2 Save Plot Method (`save_plot`)

This method is used to save plots to a specified file path. The `file_str` argument is used to define the filename of the saved plot.

2.4.3 Load Data Method (`load_data`)

This method reads data from the CSV file specified in the `xfile` attribute and processes it. It normalizes the data and stores it in the `norm_df` attribute.

2.4.4 Fit Curve Method (`fit_curve`)

This method fits a mathematical function (`fx`) to the normalized data using the `curve_fit` function from the `scipy.optimize` library. It calculates the optimized parameters (`popt`) and covariance matrix (`pcov`) for the chosen function.

2.4.5 Do Fitting Method (`do_fitting`)

This method orchestrates the entire fitting process, from loading data to fitting the curve, smoothing, and preparing data. It returns None but updates the object's attributes with the fitted results.

2.4.6 Smooth Data Method (`smooth_df`)

This method generates a meshgrid of values and evaluates the fitted function on this grid. It then rescales the results to match the original data's scales.

2.4.7 Construct Data Method (`construct_df`)

This method constructs a DataFrame containing values computed from the fitted curve. It transforms values from the meshgrid back to the original data scale.

2.4.8 Plot Experimental vs. Predicted Method (`plot_exp_pred`)

This method creates a scatter plot that compares the actual experimental data with the data predicted by the fitted curve. It also computes and displays standard deviations for the predictions.

2.4.9 Plot 2D Method (`plot_2d`)

This method generates a 2D plot that visualizes the data, both the actual experimental data and the data predicted by the fitted curve, for various temperature values.

2.4.10 Prepare Data Method (`prepare_df`)

This method prepares the data for 2D visualization, rescaling it to the original scale.

2.4.11 Plot 2D Local Method (`plot_2d_local`)

Similar to the `plot_2d` method, this method generates a 2D plot but colors different lines based on temperature values.

2.4.12 Plot 3D Method (`plot_3d`)

This method creates a 3D surface plot, visualizing the data in three dimensions (Temperature, time, and DOS). It uses Matplotlib's 3D plotting capabilities.

2.4.13 Plot Contour Method (`plot_contour`)

This method generates a contour plot that shows how the DOS varies with time and temperature. Different contour levels are used to represent different values of DOS.

2.4.14 One Derivative Method (`one_deriv`)

This method calculates and plots the derivative of the fitted function with respect to either temperature or time. It provides insight into how DOS changes concerning these parameters.

2.4.15 One Derivative Smooth Method (`one_deriv_smooth`)

Similar to the `one_deriv` method, this method calculates and plots the smoothed derivative of the fitted function.

2.4.16 Calculate Derivatives Method (`calc_deriv`)

This method calculates derivatives of the fitted function with respect to temperature and time and plots them.

2.4.17 Calculate Smooth Derivatives Method (`calc_deriv_smooth`)

This method calculates smoothed derivatives of the fitted function with respect to temperature and time and plots them.

2.4.18 Calculate Ranges Method (`calculate_ranges`)

This method calculates the temperature and time ranges corresponding to a given DOS value. It is used to determine the intervals where DOS matches a specific value.

3 Main file

This code appears to do the following:

> `fitdos.py`: The code makes use of the ’dosmodel’ module to analyze a series of CSV files found within the current directory. For each CSV file, it performs several tasks, including extracting the file name (excluding the ".csv" extension), initializing a ’DOSModel’ object with the current CSV file, and iterating through a predefined list of functions named `fx_list`. This iteration involves executing various operations such as fitting, plotting, and derivative calculations.

1. It imports the necessary functions and modules from the "dosmodel" module.
2. It retrieves a list of CSV files in the current working directory using the `get_csv_files` function.
3. For each CSV file found:
   a) It extracts the file name without the ".csv" extension and prints it.
   b) It creates a `DOSModel` object with the current CSV file.
   c) It iterates over a list of functions called `fx_list`.
   d) For each function in `fx_list`, it prints the function name.
   e) It performs various operations on the `dos_model` object, including fitting, plotting, calculating derivatives, and more.

The code essentially processes multiple CSV files using the `DOSModel` class, performing various operations for different functions specified in `fx_list` and displaying progress messages along the way.