calculate_complex_impedances(file):

Purpose: Calculates complex impedances from a file containing impedance magnitude and phase data.

Usage:

```
complex_impedances, frequencies = 
calculate_complex_impedances('your_file.csv')
```

plot_nyquists(file):

plot_nyquists(file):

Purpose: Plots a Nyquist plot for the impedance data in the given file.

Usage:

plot_nyquists('your_file.csv')

plot_multi_nyquist(files,phase_shift=True):

Purpose: Plots multiple Nyquist plots on the same graph for comparison.

Usage:

files = ['file1.csv', 'file2.csv', 'file3.csv']

plot_multi_nyquist(files, phase_shift=True)

calculate_and_plot_bode_multi(files, phase_shift=True)

Purpose: Calculates and plots Bode plots for multiple files.

Usage:

files = ['file1.csv', 'file2.csv', 'file3.csv']

calculate_and_plot_bode_multi(files, phase_shift=True)

plot_impedance_analysis(file, time_point=None, start_time=None, end_time=None, num_plots=10, phase_shift=True):

plot_impedance_analysis(file, time_point=None, start_time=None, end_time=None, num_plots=10, phase_shift=True):
Purpose: Plots detailed impedance analysis including Bode and Nyquist plots for a specific file.

Usage:

plot_impedance_analysis('your_file.csv', time_point=15) # For a specific time point

plot_impedance_analysis('your_file.csv', start_time=10, end_time=20, num_plots=5) # For a time range

fit_circuit(frequencies, complex_impedances, order=1):

Purpose: Fits an equivalent circuit model to the impedance data.
Usage:

circuit, fitted_params = fit_circuit(frequencies, complex_impedances, order=2)

plot_and_fit_impedance(file, order=2, time_point=None, phase_shift=True, smooth=False, window_length=15, polyorder=3, optimize=True):

Purpose: Plots impedance data and fits an equivalent circuit model.

Usage:

circuit, data, frequencies = plot_and_fit_impedance('your_file.csv', order=2, smooth=True)

write_to_excel(data_list, output_file="impedance_analysis_results.xlsx"):

Purpose: Writes analysis results to an Excel file.

Usage:

write_to_excel(all_data, "your_output_file.xlsx")

analyze_cutoff_frequency_evolution(files, parameter_name):

Purpose: Analyzes how the cutoff frequency changes with different concentrations of a parameter.

Usage:

concentrations, cutoff_freqs = analyze_cutoff_frequency_evolution(['file1.csv', 'file2.csv'], 'Sugar')

plot_element_impact(files, element_name, concentrations):

Purpose: Plots the impact of different concentrations of an element on impedance spectra.

Usage:

files = ['file1.csv', 'file2.csv', 'file3.csv']

concentrations = [0, 150, 300]

plot_element_impact(files, 'Sugar', concentrations)

load_fermentation_data(file):

Purpose: Loads impedance data from a fermentation experiment file

Usage:

frequencies, time_points, impedance_data, phase_data = load_fermentation_data('fermentation_data.csv')

load_and_process_data(file, phase_shift=True):

Purpose: Loads and processes impedance data, including optional phase shift.

Usage:

frequencies, mean_impedance, mean_phase, complex_impedances = load_and_process_data('your_file.csv', phase_shift=True)

plot_impedance_spectra(ax, frequencies, complex_impedances, label, color):

Purpose: Plots impedance spectra on given axes.

Usage:

fig, ax = plt.subplots(1, 3, figsize=(18, 6))

plot_impedance_spectra(ax, frequencies, complex_impedances, 'Sample 1', 'red')

analyze_single_elements(files, concentrations, element_name):

Purpose: Analyzes and plots impedance data for a single element at different concentrations.

Usage:

files = ['sugar_0.csv', 'sugar_150.csv', 'sugar_300.csv']

concentrations = [0, 150, 300]

analyze_single_elements(files, concentrations, 'Sugar')

analyze_impedance_variation(files, concentrations, element_name, frequencies):

Purpose: Analyzes how impedance varies with concentration for a specific element.

Usage:

analyze_impedance_variation(files, concentrations, 'Sugar', frequencies)

analyze_combinations(files, conditions):

Purpose: Analyzes impedance data for combinations of different elements.

Usage:

files = ['comb1.csv', 'comb2.csv', 'comb3.csv']

conditions = [{'S': 150, 'N': 0.25, 'A': 71}, {'S': 300, 'N': 0.25, 'A': 71}, {'S': 0, 'N': 0.25, 'A': 71}]

analyze_combinations(files, conditions)

sensitivity_analysis():

Purpose: Performs a sensitivity analysis on impedance data for different parameters.
Usage:

sensitivity_analysis()

Remark: you have to change the file used in the function

calculate_equivalent_impedance(*impedances):

Purpose: Calculates the equivalent impedance for parallel connection of two or more impedances.
Usage:

Z_equivalent = calculate_equivalent_impedance(Z1, Z2, Z3)

get_complex_impedance(file, time_point=0):

Purpose: Extracts complex impedance data from a file for a specific time point.
Usage:

frequencies, Z = get_complex_impedance('your_file.csv', time_point=5)

compare_rc_circuits(circuit1, circuit2, circuit_combined, freq_range, labels):

Purpose: Compares the impedance of two parallel RC circuits with a single RC circuit.

Usage:

compare_rc_circuits(circuit1, circuit2, circuit_combined, freq_range, {'file1': 'Sample1', 'file2': 'Sample2', 'file_combined': 'Combined'})

compare_impedances(file1, file2, file_combined, labels):

Purpose: Compares impedances of individual components with a combined solution.

Usage:

compare_impedances('sample1.csv', 'sample2.csv', 'combined.csv', {'measured': 'Measured', 'calculated': 'Calculated'})