A screenshot of a computer

Description automatically generatedA black screen with white text

Description automatically generated

A screenshot of a black and white screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

% Load data from Excel

data = readtable('defect\_data.xlsx');

% Extract data

T = data.Temperature; % Column for temperature

t = data.Time; % Column for time

N = data.DefectConcentration; % Column for defect concentration

% Boltzmann constant

k = 8.617333262145e-5; % eV/K

% Define the defect concentration function for a single temperature

defect\_concentration = @(params, t) params(1) \* (params(2) + params(3) \* exp(-t / params(4)));

% Extract relaxation times for each temperature

unique\_T = unique(T);

tau\_T = zeros(size(unique\_T));

for i = 1:length(unique\_T)

% Filter data for the current temperature

T\_current = unique\_T(i);

indices = T == T\_current;

t\_current = t(indices);

N\_current = N(indices);

% Initial guess for parameters [Neq, c0, ci, tau]

initial\_guess = [1, 0.1, 0.1, 1];

% Fit the data to the defect concentration model

fit\_params = lsqcurvefit(@(p, x) defect\_concentration(p, x), initial\_guess, t\_current, N\_current);

% Extract the fitted relaxation time

tau\_T(i) = fit\_params(4);

end

% Display extracted relaxation times

disp('Extracted relaxation times tau(T):');

disp(table(unique\_T, tau\_T));

% Now use the Arrhenius equation to fit tau(T) to estimate tau\_0 and Ea

arrhenius\_model = @(params, T) params(1) \* exp(params(2) ./ (k \* T));

% Objective function for fitting the Arrhenius model

arrhenius\_objective = @(params) sum((tau\_T - arrhenius\_model(params, unique\_T)).^2);

% Initial guess for tau\_0 and Ea

initial\_arrhenius\_params = [1e-12, 1.0]; % [tau\_0\_initial, Ea\_initial]

% Fit the Arrhenius model to the extracted relaxation times

arrhenius\_params = fminsearch(arrhenius\_objective, initial\_arrhenius\_params);

% Extract fitted tau\_0 and Ea

tau\_0\_fit = arrhenius\_params(1);

Ea\_fit = arrhenius\_params(2);

% Display fitted parameters

fprintf('Fitted tau\_0: %.2e s\n', tau\_0\_fit);

fprintf('Fitted Ea: %.2f eV\n', Ea\_fit);

% Define the annealing curve function using the fitted tau\_0 and Ea

annealing\_curve = @(params, t, T) params(1) \* (params(2) + ...

params(3) \* exp(-t ./ (tau\_0\_fit \* exp(Ea\_fit ./ (k \* T)))) + ...

params(4) \* exp(-t ./ (tau\_0\_fit \* exp(params(5) ./ (k \* T)))));

% Objective function for fitting the annealing curve

objective\_function = @(params) sum((N - annealing\_curve(params, t, T)).^2);

% Initial guess for the fitting parameters [Neq, c0, ci1, ci2, Ea1, Ea2]

initial\_guesses = [1, 0.1, 0.1, 0.1, 1.0];

% Perform the fitting using fminsearch

options = optimset('Display', 'iter');

fitted\_params = fminsearch(objective\_function, initial\_guesses, options);

% Extract fitted parameters

Neq\_fit = fitted\_params(1);

c0\_fit = fitted\_params(2);

ci\_fit = fitted\_params(3:4);

Ei\_fit = fitted\_params(5);

% Display fitted parameters

fprintf('Fitted Neq: %.2f\n', Neq\_fit);

fprintf('Fitted c0: %.2f\n', c0\_fit);

fprintf('Fitted ci: %.2f %.2f\n', ci\_fit);

fprintf('Fitted Ea: %.2f\n', Ei\_fit);

% Plot results

figure;

hold on;

for i = 1:length(unique\_T)

T\_current = unique\_T(i);

indices = T == T\_current;

t\_current = t(indices);

N\_current = N(indices);

plot(t\_current, N\_current, 'o', 'DisplayName', ['Experimental data T = ' num2str(T\_current)]);

plot(t\_current, annealing\_curve(fitted\_params, t\_current, T\_current), '-', 'DisplayName', ['Fitted curve T = ' num2str(T\_current)]);

end

xlabel('Time (s)');

ylabel('Relative Defect Concentration');

legend show;

grid on;

A screenshot of a black background

Description automatically generated