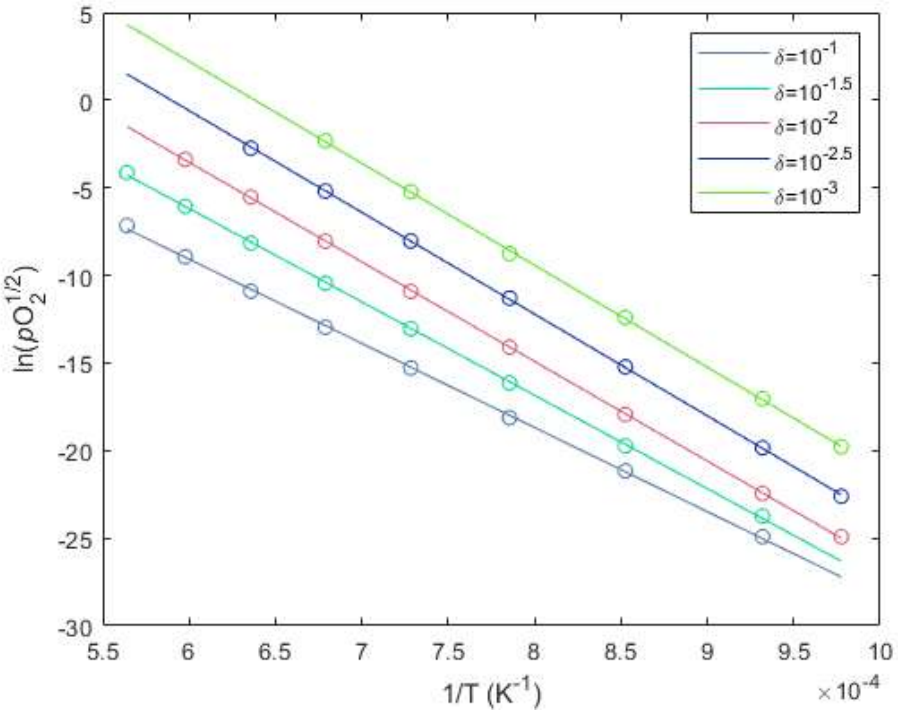
```
% Problem 3
clear;clc;

% Define constants and parameters
R = 8.314; %Universal gas constant (kJ/molK)
temp = [1773 1673 1573 1473 1373 1273 1173 1073 1023]; % Temperatures at which data was collected.
delta = [10^-1 10^-1.5 10^-2 10^-2.5 10^-3]; % Range chosen to cover a wide span of experimental data

% Import data obtained experimentally such as that seen in Figure 22.12
dataTable = readtable('Fig2212_data.csv');

% Convert data table into an array.
% Note: you can refer to the table form for column names
dataArray = table2array(dataTable);

% Organize temperature, delta, and pO2 data into a structure and assign
% polynomial fits of pO2 = f(delta)
for i = 1:length(temp)

    num = strcat('T',num2str(temp(i)));
    data.(num)(:,1) = dataArray(:,2*i-1); % pO2 (atm)
    data.(num)(:,2) = dataArray(:,2*i);   % delta

        % Use this for loop to delete NaN entries from data import caused by
        % an unequal number of data points available for each isotherm. This
        % will allow for creation of a polynomial fit.
        for j = 1:length(data.(num))
            if isnan(data.(num)(j,1)) == true
                data.(num)(j:length(data.(num)),:)=[];
                break
            end
        end

    % Create a polynomial structure and store polynomial fits.
    % Choose an order that best fits your data. Here, a fifth order
    % polynomial is used.

    % Note: the data is fitted in log10 form to help more accurately fit the
    % data.
    poly.(num) = polyfit(log10(data.(num)(:,2)),log10(data.(num)(:,1)),5);

end

% For each constant value of delta, search each set of temperature data and
% check if it contains the current constant delta value.
% If so, evaluate the polynomial for that temperature at the current constant
% delta to obtain pO2 using the polyval function.

% Store that value as 0.5*log(pO2), along with the inverse of the
% temperature at which that value was solved. Here, these values are being
% stored in a data structure called ellinghamData.

% Note:log(x) = natural logarithm in MATLAB v2020b
for i = 1:length(delta)
    k = 1;
    num1 = strcat('d',num2str(i));

    for j = 1:length(temp)
        num2 = strcat('T',num2str(temp(j)));
        if delta(i) > min(data.(num2)(:,2)) && delta(i) < max(data.(num2)(:,2))
            ellinghamData.(num1)(k,1) = 0.5*log(10^polyval(poly.(num2),log10(delta(i))));
            ellinghamData.(num1)(k,2) = 1/temp(j);
            k = k+1;
        end
    end
```

```
end

% Plot the data obtained for the ellingham diagram for each constant delta
% value. Inverse temperature will go on the x-axis, while 0.5ln(pO2) will
% go on the y-axis. Fit a line to each constant delta data set and plot.
% Scale the slope and intercept of the linear fit for each delta
% appropriately and store each in an array.
for i = 1:length(delta)

    col = rand(3,1);
    num = strcat('d',num2str(i));
    plot(ellinghamData.(num)(:,2),ellinghamData.(num)(:,1),'o','Color',col,'HandleVisibility','Off');
    hold on;
    polyProp.(num) = polyfit(ellinghamData.(num)(:,2),ellinghamData.(num)(:,1),1);
    plot(1./temp,polyval(polyProp.(num),1./temp),'-','Color',col);

    H(i) = -(polyProp.(num)(1)*R)/1000;
    S(i) = (polyProp.(num)(2)*R);

end

% Plot labels
xlabel('1/T (K^{-1})');
ylabel('ln({\itp}O_{2}^{1/2})')
legend('\delta=10^{-1}','\delta=10^{-1.5}','\delta=10^{-2}','\delta=10^{-2.5}','\delta=10^{-3}')

% Concatinate thermodynamic property arrays and output as a table
output = cat(1,H,S);
array2table(output,'VariableNames',{'delta=10^-1','delta=10^-1.5','delta=10^-2'...
            ,'d=10^-2.5','d=10^-3'},'RowNames',{'Enthalpy (kJ/mol)','Entropy (J/molK)'})
```

```
ans =

  2×5 table

                       delta=10^-1        delta=10^-1.5        delta=10^-2          d=10^-2.5            d=10^-3
                    _________________    _________________    _________________    _________________    _________________

    Enthalpy (kJ/mol)    399.411993399213    443.119281300684     472.56118350642     482.76937807391    483.917465286416
    Entropy (J/molK)     164.159749819852    214.416176126595    254.027038394741    284.674518892935    308.785108716772
```



Published with MATLAB® R2020b