ESC 113 Project Group 3 Code file
By Euler's method

```
% Initial concentration of hydrogen peroxide
h2o2_0 = 2.0 ;
%Temperature = 333 K
T=333;
% Rate constant at T = 333 K
k =rateconstant(T); % s^-1
% Time interval i.e of the 40s
tspan = [0, 40]; % seconds
dt = 0.01; % s
% Number of time steps
n = ((tspan(2) - tspan(1)) / dt);
% Initialize arrays to store concentrations and time
t = zeros(n + 1, 1);
h2o2 = zeros(n + 1, 1);
h2o = zeros(n + 1, 1);
o2 = zeros(n + 1, 1);
% Set initial concentration for the reaction
t(1) = tspan(1);
h2o2(1) = h2o2_0;
% Performing Euler's explicit method
for i = 1:n
t(i+1) = t(i) + dt;
% Calculate the derivatives
dh2o2_dt = -2 * k * h2o2(i);
dh2o_dt = 2 * k * h2o2(i);
do2_dt = k * h2o2(i);
% Update the concentrations
h2o2(i+1) = h2o2(i) + dh2o2_dt*dt;
h2o(i+1) = h2o(i) + dh2o_dt*dt;
o2(i+1) = o2(i) + do2_dt*dt;
% Ensuring concentrations of the reactant, as well as the product, don't go
below zero
if h2o2(i+1) < 0
h2o2(i+1) = 0;
end
if h2o(i+1) < 0
h2o(i+1) = 0;
end
if o2(i+1) < 0
o2(i+1) = 0;
end
end
% Plot the concentrations over time
figure;
plot(t, h2o2, 'r-', 'LineWidth', 2);
```

```
hold on;
plot(t, h2o, 'b-', 'LineWidth', 2);
plot(t, o2, 'g-', 'LineWidth', 2);
hold off;
xlabel('Time (s)');
ylabel('Concentration (M)');
legend('H2O2', 'H2O', 'O2');
title('Decomposition of Hydrogen Peroxide Simulation');
grid on;
function k=rateconstant(T)
Ea=75000;
A=1.2*(10^11);
R=8.314;
k=zeros(size(T,1),1);
for i=1:size(T,1)
k(i)=A*(10^(-Ea/(2.303*R*T(i))));
end
end
```

By Runge-Kutta method

```
% Initial concentration of components
h2o2_0 = 2.0; % M
% Rate constant calculated at 333 K as per function given at the end
t = [0, 40];
dt = 0.01; % s
% Number of time steps in the simulation
Steps = ceil((t(2) - t(1)) / dt);
t = zeros(Steps + 1, 1);
h2o2 = zeros(Steps + 1, 1);
h2o = zeros(Steps + 1, 1);
o2 = zeros(Steps + 1, 1);
% Setting initial concentration of components
t(1) = t(1);
h2o2(1) = h2o2_0;
% Perform Runge-Kutta method
for i = 1:Steps
t(i+1) = t(i) + dt;
% Calculate k1, k2, k3, k4 for H2O2 for Runge-Kutta
k1_h2o2 = -2 * k * h2o2(i);
k2_h2o2 = -2 * k * (h2o2(i) + k1_h2o2*dt/2);
k3_h2o2 = -2 * k * (h2o2(i) + k2_h2o2*dt/2);
k4_h2o2 = -2 * k * (h2o2(i) + k3_h2o2*dt);
% Update the concentration of H2O2
h2o2(i+1) = h2o2(i) + (k1_h2o2 + 2*k2_h2o2 + 2*k3_h2o2 + k4_h2o2) * dt / 6;
% Calculate k1, k2, k3, k4 for H2O for Runge-Kutta
k1_h2o = 2 * k * h2o2(i);
```

```
k2_h2o = 2 * k * (h2o2(i) + k1_h2o*dt/2);
k3_h2o = 2 * k * (h2o2(i) + k2_h2o*dt/2);
k4_h2o = 2 * k * (h2o2(i) + k3_h2o*dt);
% Update the concentration of H2O water
h2o(i+1) = h2o(i) + (k1_h2o + 2*k2_h2o + 2*k3_h2o + k4_h2o) * dt / 6;
% Calculate k1, k2, k3, k4 for O2 for Runge-Kutta
k1_o2 = k * h2o2(i);
k2_o2 = k * (h2o2(i) + k1_o2*dt/2);
k3_o2 = k * (h2o2(i) + k2_o2*dt/2);
k4_o2 = k * (h2o2(i) + k3_o2*dt);
% Update the concentration of O2 dioxygen
o2(i+1) = o2(i) + (k1_o2 + 2*k2_o2 + 2*k3_o2 + k4_o2) * dt / 6;
% Ensuring concentration to be in range; positive
if h2o2(i+1) < 0
h2o2(i+1) = 0;
end
if h2o(i+1) < 0
h2o(i+1) = 0;
end
if o2(i+1) < 0
o2(i+1) = 0;
end
end
% Plotting the concentration of compounds versus time
figure;
plot(t, h2o2, 'r-', 'LineWidth', 2);
hold on;
plot(t, h2o, 'b-', 'LineWidth', 2);
plot(t, o2, 'g-', 'LineWidth', 2);
hold off;
xlabel('Time (s)');
ylabel('Concentration (M)');
legend('H2O2', 'H2O', 'O2');
title('Decomposition of H2O2 Simulation (Runge-Kutta)');
grid on;
%function for calculating rate constant k at a particular temperature
T=[333];
k=rateconstant(T);
disp(k);
function k=rateconstant(T)
Ea=75000;
A=1.2*(10^11);
R=8.314;
k=zeros(size(T,1),1);
for i=1:size(T,1)
k(i)=A*(10^(-Ea/(2.303*R*T(i))));
end
end
```