Table of Contents

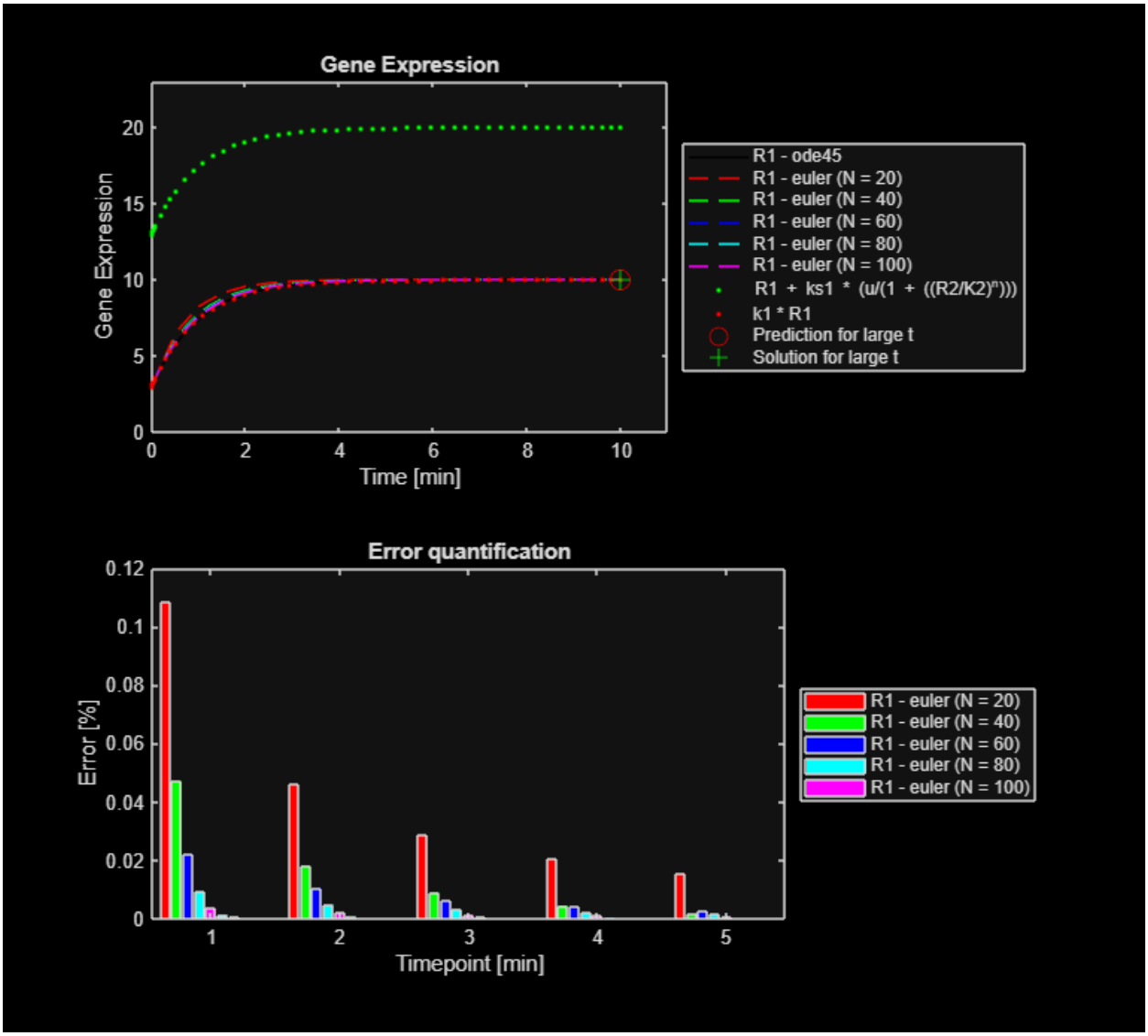

```
clear all
close all
clc
```

Task 1.1.

```
clc
% load parameters

load('p.mat', 'p')

time = [0 10];

% definition of only R1_0 as we need the initCond vector and the output
% vector to be the same length because of the ode function

y0 = [3];

% calculation using ODE function

[time, R1_ode] = ode45(@(t,initCond) model1(t,initCond,p), time, y0);

% calculation using Euler_method
% new timeRange needed to make sure R1_euler and time are the same length
% [R1_euler, timeRange] = euler_ode_solv(time, 40, y0, p); --> included in
% following test

% testing if higher N improve the result

for i = 1:5
    [R1_euler{i}, timeRange{i}] = euler_ode_solv(time, (i*20), y0, p);
end

% quantifying the error
% scanning of different cell elemtents (different N)

for i = 1:5
    % scanning timeRange for timepoints 1 to 10 and saving as idx_euler

    for j = 1:10
```

```
        idx_euler = find(timeRange{i} == j);
        for m = 1:49
            %scanning time for timepoints

            if time(m)> timeRange{i}(idx_euler)
                % calculating error

                error(i,j) = (R1_euler{i}(idx_euler) -
((R1_ode(m-1)+R1_ode(m))/2))/((R1_ode(m-1)+R1_ode(m))/2);
                break
            end
        end
    end
end

% testing for large values of t

% numerical approach

largeTimeRange = linspace(0,1000,5000);
[largeR_euler, largeTime] = euler_ode_solv(largeTimeRange, 1000, y0, p);
% Equilibrium calculation

R1Eq = equilibriumCalc(1,y0,p);

% seperate plotting

R1_1 = R1_ode + 10;
R1_2 = R1_ode;

% plotting

figure(1)
subplot(2,1,1)
plot(time, R1_ode(:,1), 'k-', ...
    timeRange{1}, R1_euler{1}, 'r--', ...
    timeRange{2}, R1_euler{2}, 'g--', ...
    timeRange{3}, R1_euler{3}, 'b--', ...
    timeRange{4}, R1_euler{4}, 'c--', ...
    timeRange{5}, R1_euler{5}, 'm--', ...
    time, R1_1, 'g.', ...
    time, R1_2, 'r.')
hold on
plot(10, largeR_euler(end), "ro", MarkerSize=10)
plot(10, R1Eq, "g+", MarkerSize=10)
title('Gene Expression')
xlabel('Time [min]')
ylabel('Gene Expression')
xlim([0 11])
ylim([0 23])
legend('R1 - ode45', ...
    'R1 - euler (N = 20)', ...
    'R1 - euler (N = 40)', ...
    'R1 - euler (N = 60)', ...
```

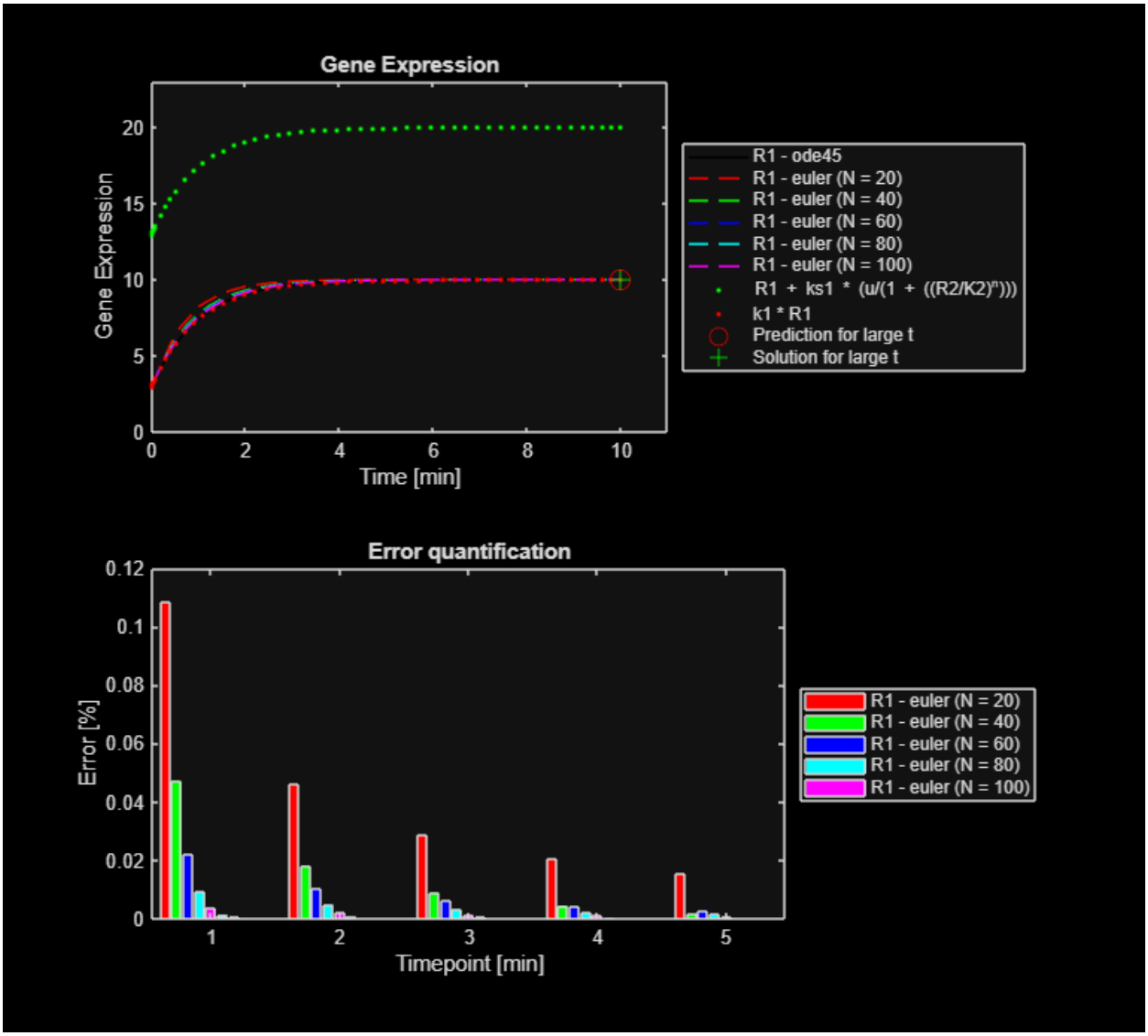
```
    'R1 - euler (N = 80)', ...
    'R1 - euler (N = 100)', ...
    'R1 + ks1 * (u/(1 + ((R2/K2)^n)))', ...
    'k1 * R1', ...
    'Prediction for large t', ...
    'Solution for large t', ...
    'Location', 'eastoutside');
subplot(2,1,2)
bars = bar(error);
title('Error quantification')
xlabel('Timepoint [min]')
ylabel('Error [%]')
legend('R1 - euler (N = 20)', ...
    'R1 - euler (N = 40)', ...
    'R1 - euler (N = 60)', ...
    'R1 - euler (N = 80)', ...
    'R1 - euler (N = 100)', ...
    'Location', 'eastoutside')

% making sure plots and bars have the same colors

colors = [1 0 0;
    0 1 0;
    0 0 1;
    0 1 1;
    1 0 1];

for i = 1:height(colors)
    bars(i).FaceColor = colors(i,:);
end
```

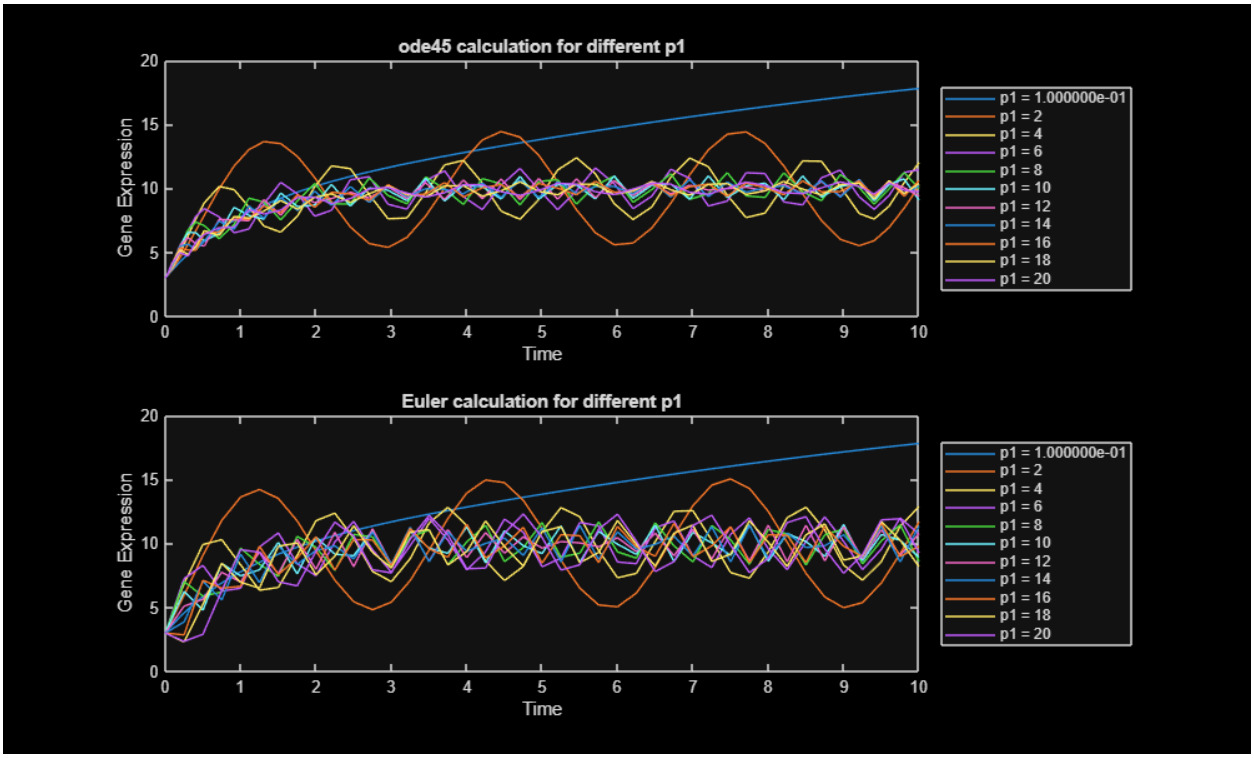


Task 1.2.

```
% define initial conditions

p1 = 0.1:0.1:20;

% calculation using ode45

for i = 1:length(p1);
    [time_odeU{i}, R1_odeU{i}] = ode45(@(t, y) submodel1(t, y, p, p1(i)),
time, y0);
end

% calculation using Euler's method

for i = 1:length(p1);
    [R1_eulerU{i}, timeRangeU{i}] = subtask1(time, 40, y0, p, p1(i));
```

```
end

% creating a dynamic plot with dynamic labels

figure(2)
set(gcf, 'Position', [100, 100, 1000, 600]);
subplot(2,1,1)
plot(time_odeU{1}, R1_odeU{1});
label_ode{1} = sprintf('p1 = %d', p1(1));
hold on
for i = 1:length(p1)/20;
    plot(time_odeU{i*20}, R1_odeU{i*20});
    hold on
    label_ode{i+1} = sprintf('p1 = %d', p1(i*20));
end
legend(label_ode, 'Location', 'eastoutside');
xlabel('Time');
ylabel('Gene Expression');
title('ode45 calculation for different p1')

subplot(2,1,2)
plot(time_odeU{1}, R1_odeU{1});
label_euler{1} = sprintf('p1 = %d', p1(1));
hold on
for i = 1:length(p1)/20;
    plot(timeRangeU{i*20}, R1_eulerU{i*20});
    label_euler{i+1} = sprintf('p1 = %d', p1(i*20));
end
legend(label_euler, 'Location', 'eastoutside');
xlabel('Time');
ylabel('Gene Expression');
title('Euler calculation for different p1')
```

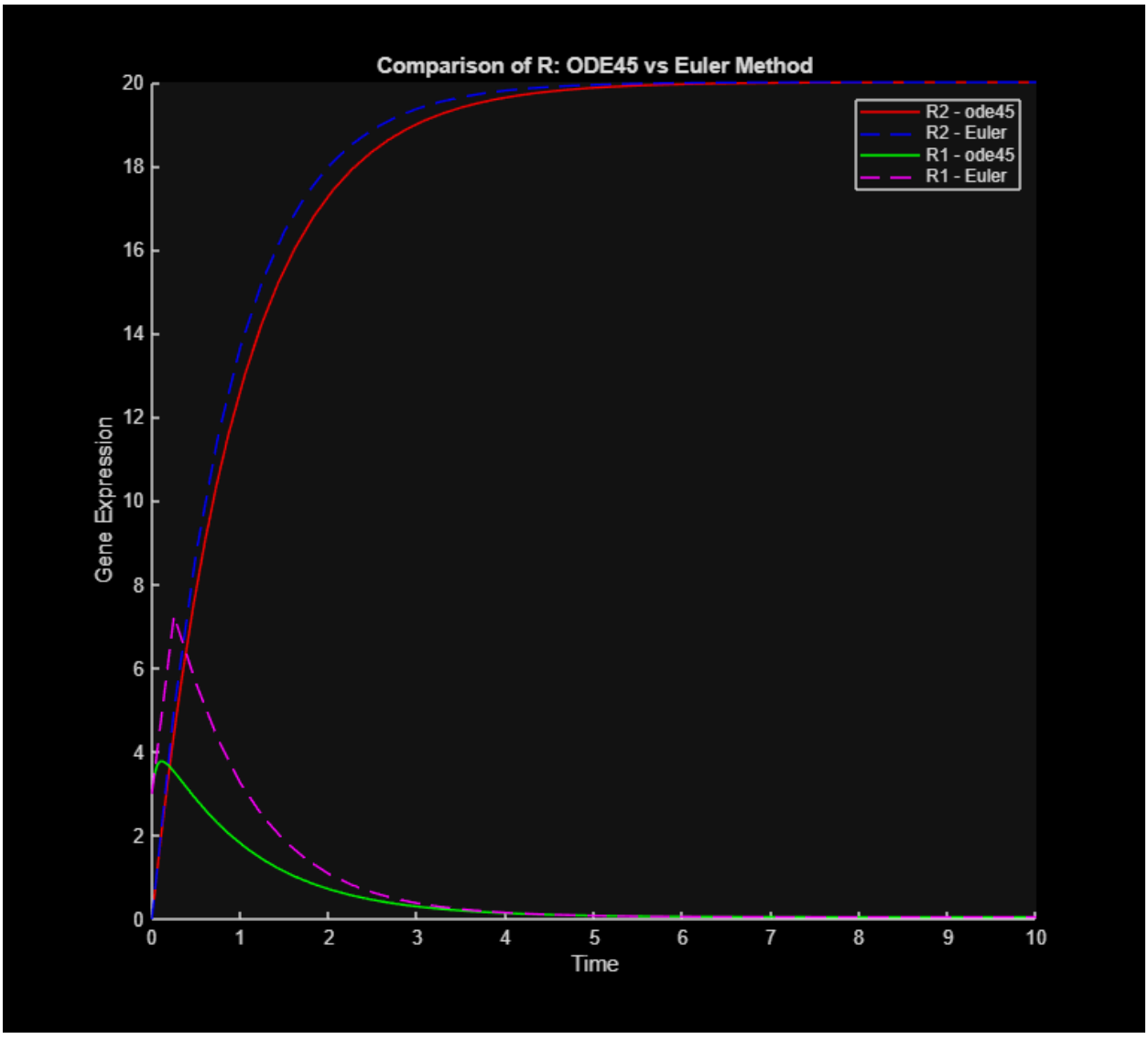

Gene Expression
Gene Expression
Time [min]
R1 - ode45
R1 - euler (N = 20)
R1 - euler (N = 40)
R1 - euler (N = 60)
R1 - euler (N = 80)
R1 - euler (N = 100)
R1 + ks1 * (u/(1 + ((R2/K2)^n)))
k1 * R1
Prediction for large t
Solution for large t
Error quantification
Error [%]
Timepoint [min]
R1 - euler (N = 20)
R1 - euler (N = 40)
R1 - euler (N = 60)
R1 - euler (N = 80)
R1 - euler (N = 100)



Task 2.1.

```
% Time range

timeR2 = [0 10];

% Initial conditions for R1 and R2

y0 = [3, 0];

% Calculation using ODE function

[time_odeR2, R] = ode45(@(t, y) task2model(t, y, p), timeR2, y0);

% Extract R2 from the ODE solution
R1_ode = R(:, 1);
R2_ode = R(:, 2);

% Calculation using Euler method (task2euler)

[R_euler, timeRange2] = task2euler(timeR2, 40, y0, p);

% Extract R2 from Euler method results (the second column)
R1_euler = R_euler(:, 1);
R2_euler = R_euler(:, 2);

% Plot the results
```

```
figure(3);
hold on;
plot(time_odeR2, R2_ode, 'r-', timeRange2, R2_euler, 'b--');
plot(time_odeR2, R1_ode, 'g-', timeRange2, R1_euler, 'm--');
xlabel('Time');
ylabel('Gene Expression');
xlim([0 10]);
ylim([0 20]);
legend('R2 - ode45', 'R2 - Euler', 'R1 - ode45', 'R1 - Euler');
title('Comparison of R: ODE45 vs Euler Method');
```

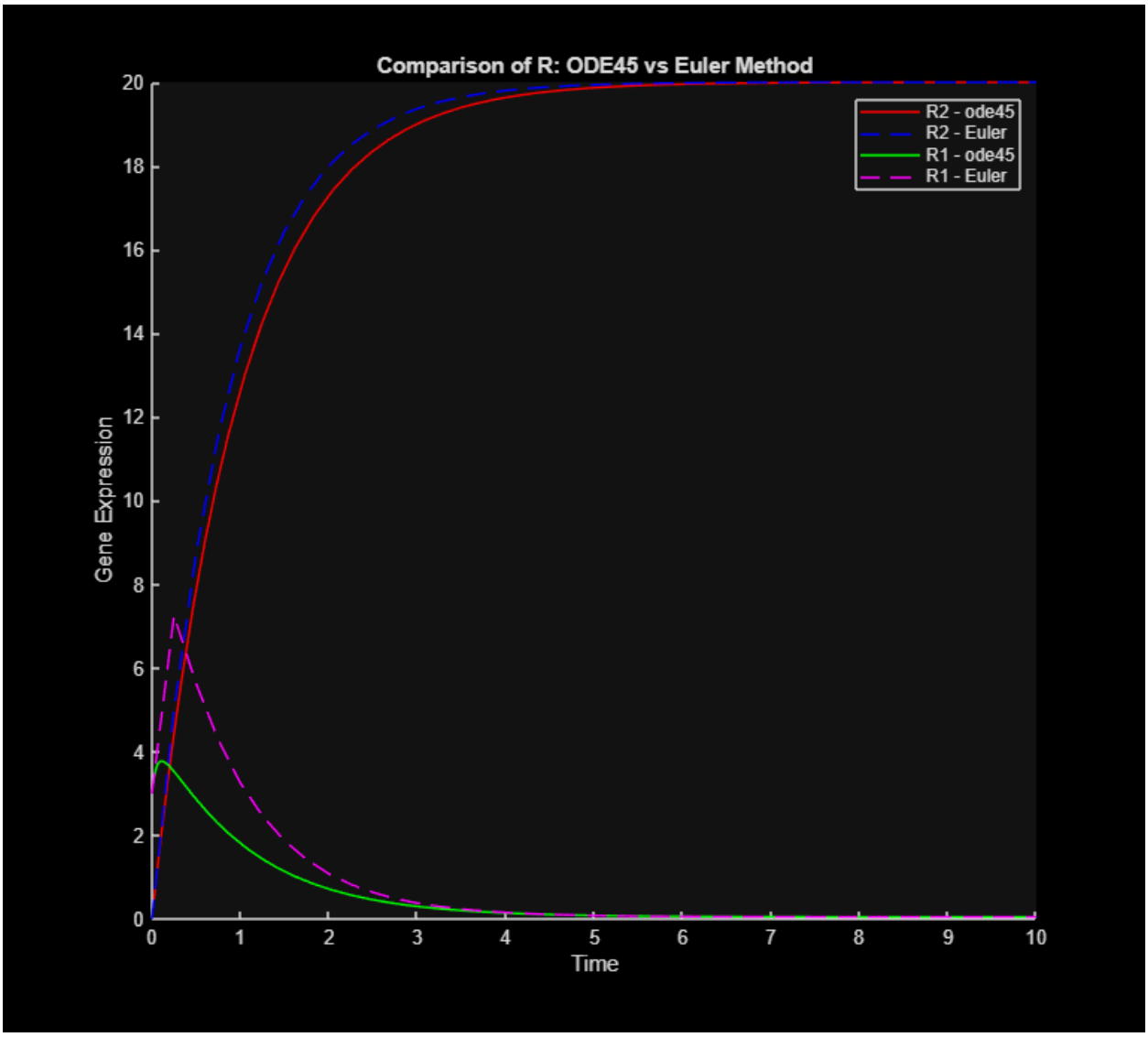


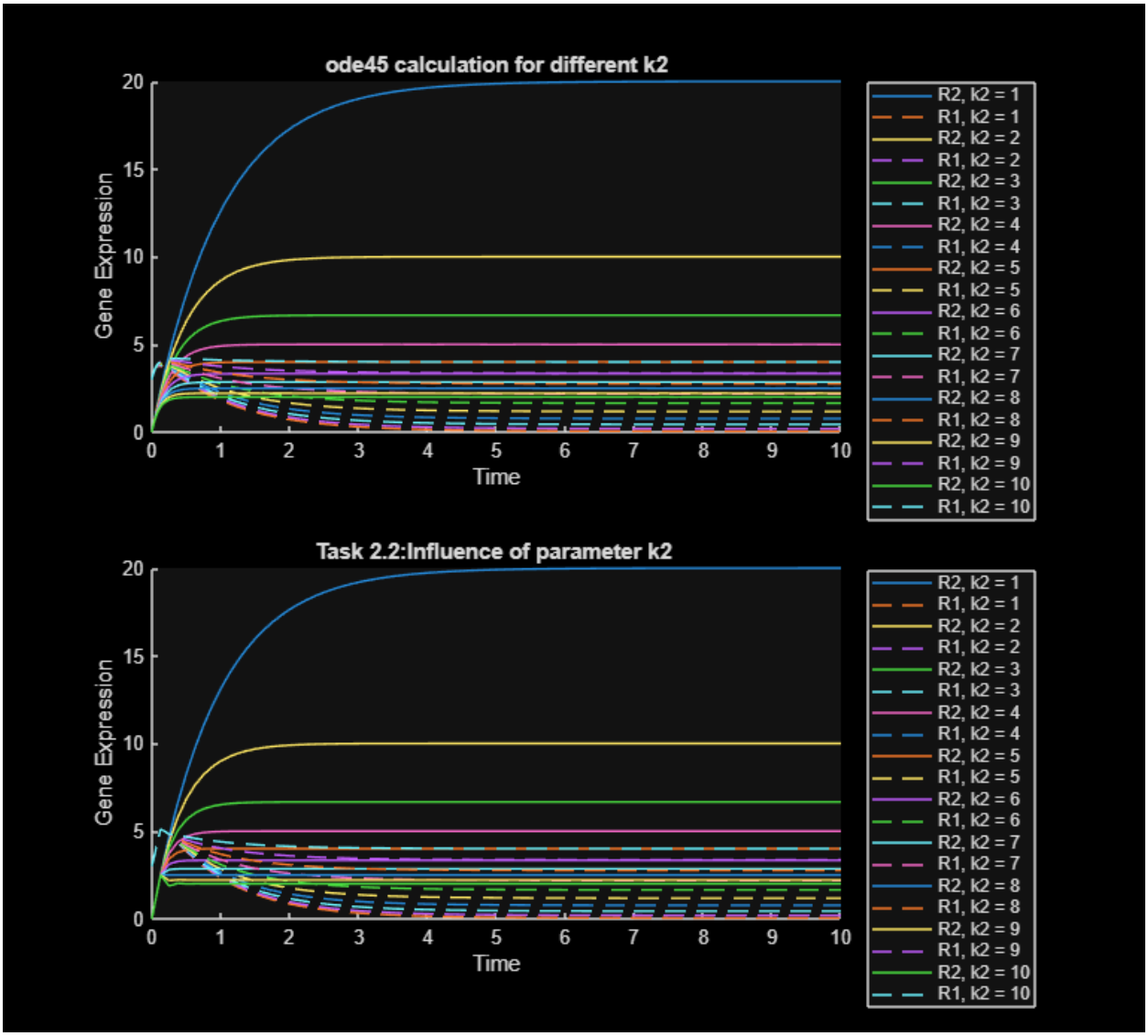
Task 2.2.

```
% Time range
time = [0 10];

% Initial conditions for R1 and R2
```

```
y0 = [3, 0];
k2 = 1:1:10;

for i = 1:length(k2)
    p.k2 = k2(i);

    % Calculation using ODE function

    [time_odek2{i}, Rk2{i}] = ode45(@(t, y) task2model(t, y, p), time, y0);

    % Extract R1 and R2 from the ODE solution

    R1_odek2{i} = Rk2{i}(:, 1);
    R2_odek2{i} = Rk2{i}(:, 2);


    % Calculation using Euler method (task2euler)

    [R22{i}, timeRangek2{i}] = task2euler(time, 80, y0, p);

    % Extract R2 from Euler method results (the second column)
    R1_eulerk2{i} = R22{i}(:,1);
    R2_eulerk2{i} = R22{i}(:,2);

end

% creating a dynamic plot with dynamic labels

figure(4)
subplot(2,1,1)
hold on
for i = 1:length(k2);
    plot(time_odek2{i}, R2_odek2{i}, 'LineStyle', '-', ...
        'DisplayName', sprintf('R2, k2 = %d', k2(i)))
     plot(time_odek2{i}, R1_odek2{i}, 'LineStyle', '--', ...
         'DisplayName', sprintf('R1, k2 = %d', k2(i)))
end
legend('Location', 'eastoutside');
xlabel('Time');
ylabel('Gene Expression');
title('ode45 calculation for different k2')
subplot(2,1,2)
hold on
for i = 1:length(k2);
    plot(timeRangek2{i}, R2_eulerk2{i},'LineStyle', '-', ...
        'DisplayName', sprintf('R2, k2 = %d', k2(i)));
    plot(timeRangek2{i}, R1_eulerk2{i}, 'LineStyle', '--', ...
         'DisplayName', sprintf('R1, k2 = %d', k2(i)))

end
legend('Location', 'eastoutside');
xlabel('Time');
ylabel('Gene Expression');
xlim([0 10]);
```

```
ylim([0 20]);
title('Task 2.2:Influence of parameter k2 ');
```

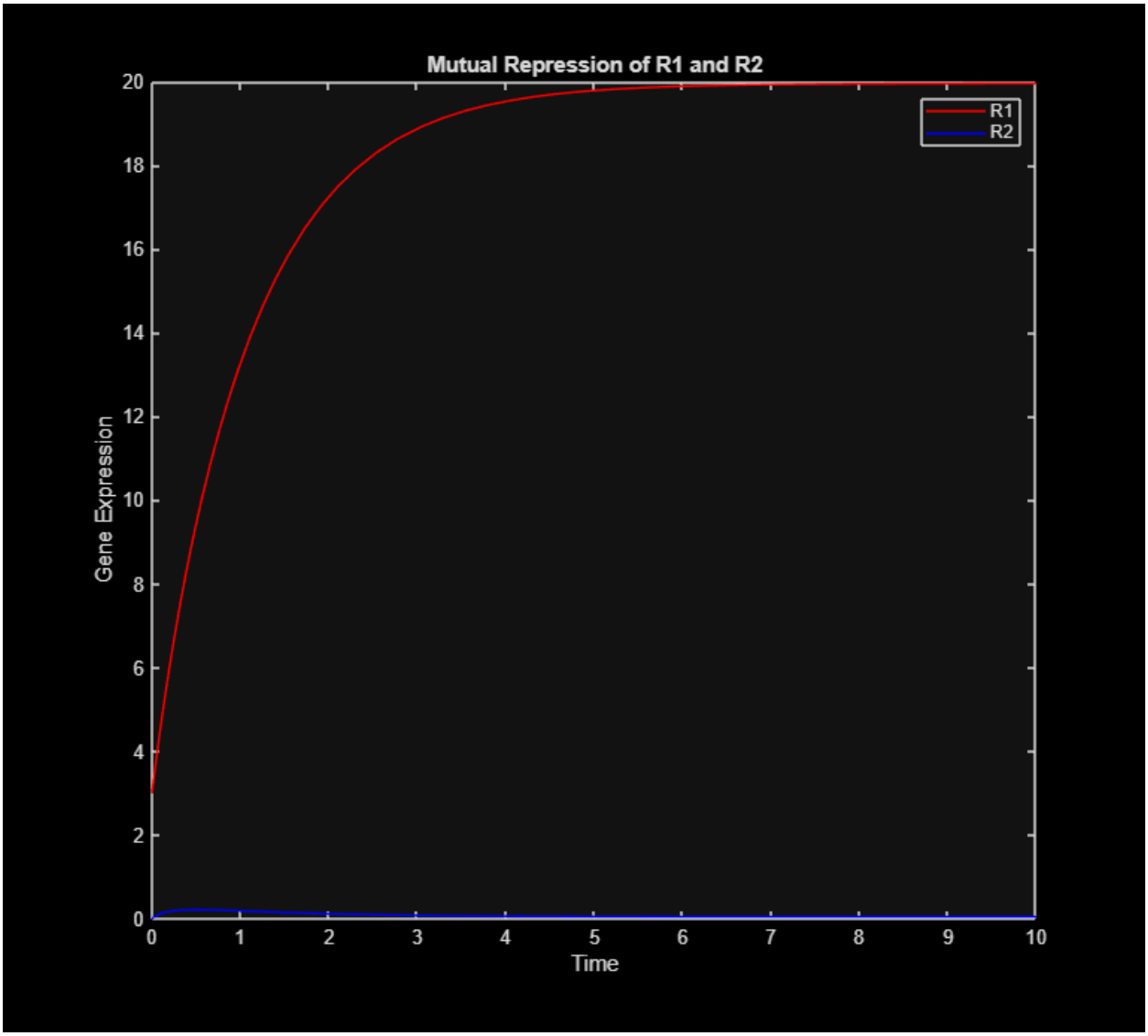


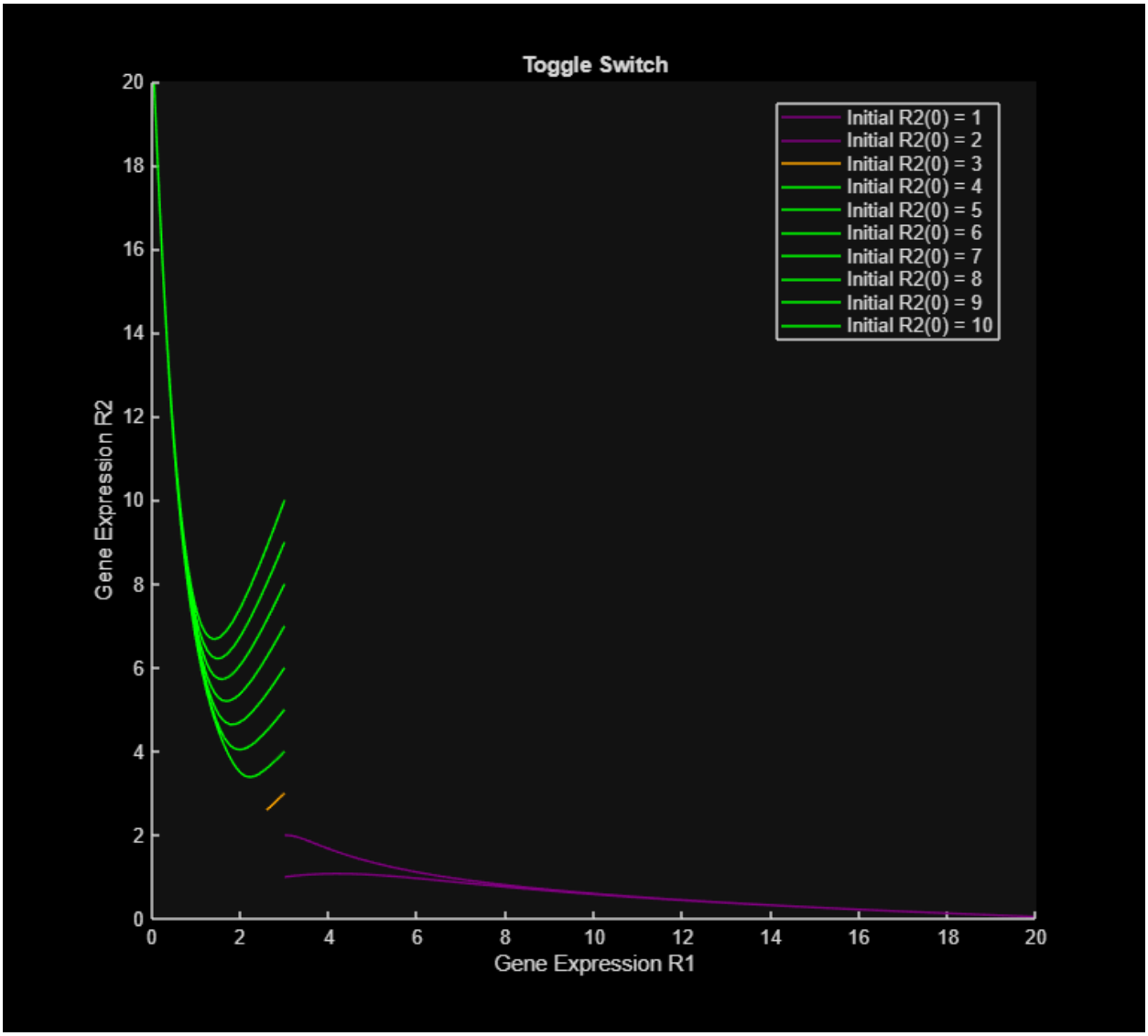


Task 3.1.

```
% Time range
load('p.mat', 'p')

timeR3 = [0 10];

% Initial conditions for R1 and R2

y0_3 = [3, 0];

% Calculation using ODE function

[time_odeR3, R_task3] = ode45(@(t, y) task3model(t, y, p), timeR3, y0_3);

% Extract R1 and R2 from the ODE solution
R1_ode_3 = R_task3(:, 1);
```

```
R2_ode_3 = R_task3(:, 2);

% Plot the results
figure(5);
plot(time_odeR3, R1_ode_3, 'r-', time_odeR3, R2_ode_3, 'b');
xlabel('Time');
ylabel('Gene Expression');
xlim([0 10]);
ylim([0 20]);
legend('R1', 'R2');
title('Mutual Repression of R1 and R2');

figure(6);
hold on
for i = 1:10;
    y0_3 = [3, i];
    [time_odeR3, R_task3] = ode45(@(t, y) task3model(t, y, p), timeR3, y0_3);
    R1_ode_3 = R_task3(:, 1);
    R2_ode_3 = R_task3(:, 2);

     if i > 3
        color = [0, 1, 0]; % Green for R2(0) > 3
    elseif i < 3
        color = [0.5, 0, 0.5]; % Purple for R2(0) < 3
    else
        color = [1, 0.647, 0]; % Orange for R2(0) = 3
     end
     plot(R1_ode_3,R2_ode_3, 'color', color);
end
xlabel('Gene Expression R1');
ylabel('Gene Expression R2');

legendStrings = cell(1, 10);
for i = 1:10
    legendStrings{i} = sprintf('Initial R2(0) = %d', i); % Only show R2(0)
in the legend
end
legend(legendStrings, 'Location', 'best', 'FontSize', 10);


title('Toggle Switch');
```

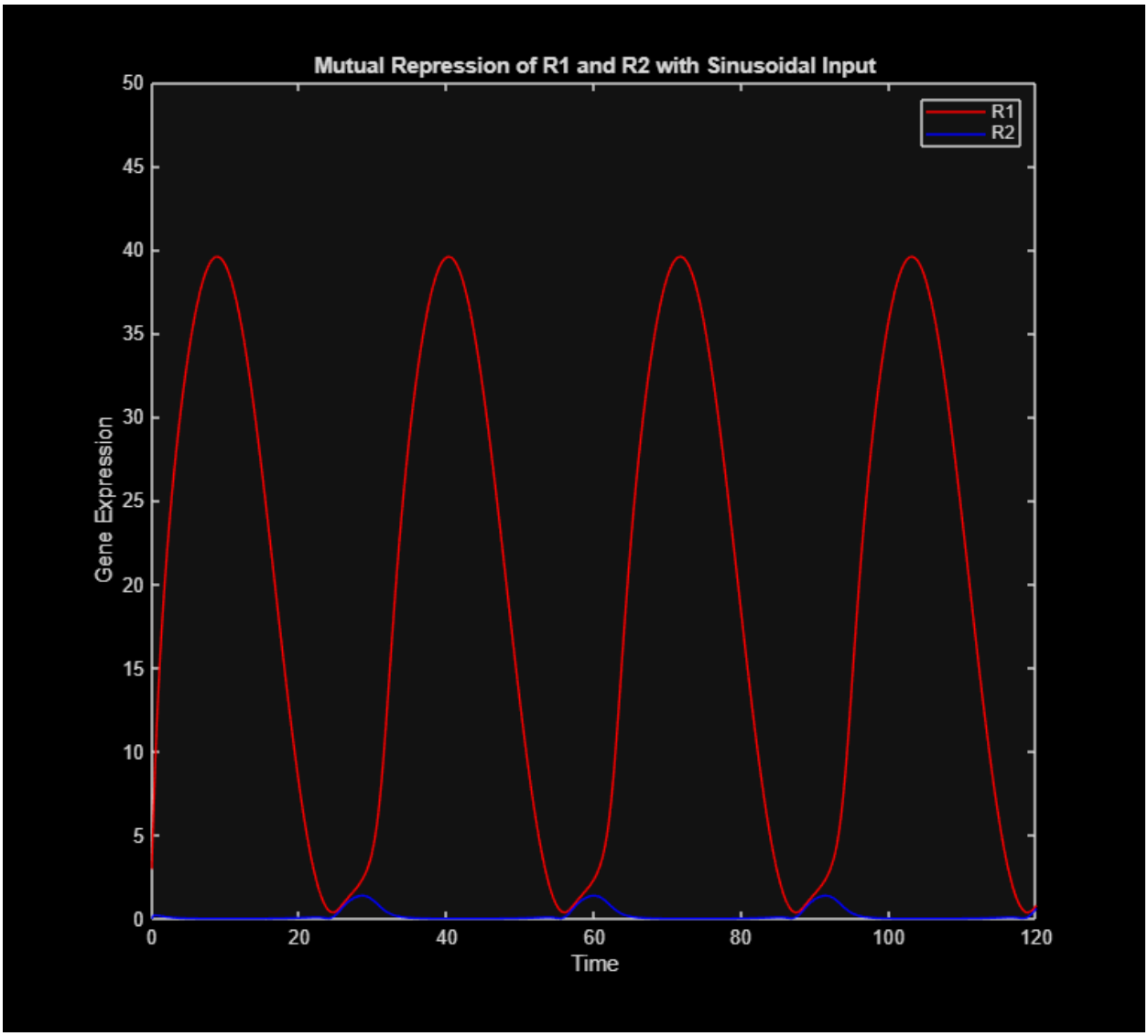

Mutual Repression of R1 and R2
R1
R2
Gene Expression
Time
0
2
4
6
8
10
12
14
16
18
20
0
1
2
3
4
5
6
7
8
9
10



Task 3.2.

```
% Time range
timeR3 = [0 120];

% Initial conditions for R1 and R2
y0_3 = [3, 0];

% Solve the ODE using ode45
[time_odeR3, R_task3_2] = ode45(@(t, y) task3_2model(t, y, p), timeR3, y0_3);

% Extract R1 and R2 from the ODE solution
R1_ode_3_2 = R_task3_2(:, 1);
R2_ode_3_2 = R_task3_2(:, 2);

% Plot the results
figure(7);
```

```
plot(time_odeR3, R1_ode_3_2, 'r-', time_odeR3, R2_ode_3_2, 'b');
xlabel('Time');
ylabel('Gene Expression');
xlim([0 120]);
ylim([0 50]);
legend('R1', 'R2');
title('Mutual Repression of R1 and R2 with Sinusoidal Input');
```



in Aufgabe 3 reicht es nur ODE zu nutzen

```
% code muss nicht in den Bericht

% Ergebnisse müssen interpreteiert werden

% Präsentation so strukturieren, dass es am meisten Sinn macht, nicht so,
% wie die Aufgaben sortiert sind

% In der Prüfung geht es mehr um die mathematischen Zusammenhänge nicht um
```

```
% Code

% Bei Task 3 Euler weglassen

% Bei Aufgabe 3.2 kann durch Ausprobieren gezeigt werden
```

Published with MATLAB® R2024b