Review

JOURNAL OF UNIVERSITY OF BABYLON
For Pure and Applied Sciences (JUBPAS)
Special Issue 2: Proceedings of Al-Muthanna University's First Conference/Directing Scientific Research Towards Achieving Sustainable

Vol, 33; S.2 | 2025

Error Analysis in Numerical Techniques using Euler's Method with a Practical Application

Ahmed Salam Razzaq [1a], Hassan Saad Hnait [1b], Safa Hassan Mohammed [1c], Oras B. Jamil [2]
Shakir Razag Alkareem [1d] , Safaa H. Rasool [3]

1 College of Education for Pure Sciences, University of Al Muthanna, Ahmed.salam@mu.edu.iq, Samawa 66001, Iraq..
2 College of Sciences, University of Al Mustansiriyah, orasbj@uomustansiriyah.edu.iq , Baghdad 10052, Iraq.
3 College of Science, University of Baghdad, Saffaa.hasan2000@gmail.com , Baghdad 10052, Iraq.
***Corresponding author email:** Ahmed.salam@mu.edu.iq ; mobile: 07810603612

Published: 31/8/2025

ABSTRACT:

This paper focuses on error analysis and practical applications in modeling dynamic systems using Euler's method in numerical techniques. The dynamics of an infectious disease using the SIR (Susceptible-Infectious-Recovered) model were the primary objective of this paper which is simulated by Euler's method. By using the MATLAB code of Euler's method, the study investigates numerically how different parameters influence the spread of the disease over time, such as transmission and recovery rates, Moreover, the results illustrate Euler's method advantage in approximating solutions to ODEs, Finally, this study insight into the dynamics of disease spread and the potential for using numerical methods to optimize epidemic control strategies.

Key words: Euler's Method ,SIR Model, Disease Dynamics, Numerical Simulation.

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

ISSN: 2312-8135| Print ISSN: 1992-0652 | www.journalofbabylon.com | jub@itnet.uobabylon.edu.iq | info@journalofbabylon.com

Review
JOURNAL OF UNIVERSITY OF BABYLON
For Pure and Applied Sciences (JUBPAS)
Special Issue 2: Proceedings of Al-Muthanna University's First Conference/Directing Scientific Research Towards Achieving Sustainable
Vol, 33; S.2 | 2025

INTRODUCTION

Error Analysis in Numerical Techniques is a critical aspect of numerical analysis that involves reading the mistakes that stand out all through the numerical computations and strategies used to limit or manipulate these errors. This area focuses on knowledge of the assets of errors, how they propagate via calculations, and how they have an effect on the accuracy and reliability of numerical consequences. [7]

In numerical evaluation, errors can arise from diverse sources, consisting of spherical-off errors due to finite precision arithmetic in computer systems, truncation mistakes attributable to approximations made in numerical algorithms, and discretization errors stemming from the use of discrete approximations to non-stop issues.

Error analysis in numerical strategies aims to quantify and analyze these mistakes, which will check the best of numerical solutions and make informed decisions about the reliability of computational consequences. By know-how and controlling errors, researchers and practitioners can enhance the accuracy and efficiency of numerical techniques and make certain the validity of computational simulations and solutions. [7]

Key topics in error analysis in numerical techniques might also encompass techniques for blunder estimation, mistake propagation evaluation, error management strategies, convergence evaluation of numerical algorithms, and techniques for improving the stability and accuracy of numerical computations. Researchers in this discipline often expand error bounds, sensitivity evaluation methods, and adaptive strategies to beautify the robustness and reliability of numerical techniques.

Overall, error analysis in numerical strategies performs a critical function in ensuring the trustworthiness of numerical computations and is crucial for advancing the field of numerical analysis and its packages in numerous scientific and engineering disciplines. The study aims to apply Euler's method to simulate the dynamics of infectious disease using the SIR model, providing numerical approximations of its governing equations. The simulation analyzes how susceptible, infectious, and recovered populations evolve over time in response to parameters like transmission and recovery rates. This approach offers insights into disease spread and supports optimizing strategies for epidemic control.

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

Review
JOURNAL OF UNIVERSITY OF BABYLON
For Pure and Applied Sciences (JUBPAS)
Special Issue 2: Proceedings of Al-Muthanna University's First Conference/Directing Scientific Research Towards Achieving Sustainable
Vol, 33; S.2 | 2025

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

MATERIALS AND METHODS

1. Numerical methods for ordinary differential equations

Numerical methods for ordinary "differential equations" involve techniques used to approximate the solutions of ODEs through numerical computations. These methods are often referred to as "numerical integration," although this term can also indicate the computation of integrals [7].

In many cases, exact solutions for "differential equations" are not feasible. Therefore, in practical applications such as engineering, obtaining a numerical approximation serves as a suitable alternative. The algorithms associated with numerical methods can effectively compute such approximations. Alternatively, calculus techniques can be employed to derive a series expansion of the solution.

ODEs are predominant in various scientific fields, including chemistry, biology, physics and economics. Moreover, some methods within numerical partial "differential equations" entail converting a "partial differential equation" into an "ordinary differential equation" for subsequent resolution.

1.1 The problem:

A first-order differential equation, typically represented as an Initial Value Problem (IVP), is expressed as follows [2]:

$$y'(t) = f\big(t, y(t)\big), \qquad y(t_0) = y_0 \tag{1}$$

Where f is a function $f: [t_0, \infty) \times Rd \to Rd$ and $y_0 \in Rd$ denotes a specified vector initial condition. The term "first-order" indicates that the equation involves only the first derivative of y, with higher-order derivatives not present.

For simplicity and without losing generality in handling higher-order systems, a focus is placed on first-order "differential equations". Higher-order ODEs can be transformed into a larger system of "first-order equations" by presenting other variables[10]. For example, converting the second-order equation $y'' = -y$ into two first-order equations yields $y' = z$ and $z' = -y$.

This segment discusses numerical methods tailored for solving Initial Value Problems (IVPs), noting that Boundary Value Problems (BVPs) necessitate distinct methodologies. In a BVP scenario, values (components) of the solution's y are defined at multiple points, requiring specialized techniques for resolution. For BVPs, approaches like the "shooting method" and its

variations, or "global methods" such as "finite differences", [3] "Galerkin methods", and "collocation methods", are more suitable for effective problem-solving[4].

1.2 Euler's Method

An excellent approximation of an adjoining factor on the given curve can be decided with the aid of moving a small distance lengthwise a tangent line to the curve. To initiate the method, we start with the "differential equation" (1) and replacement the by-product y′ with a "finite distinction approximation" [8]:

$$y'(t) \approx y(t+h) - y(t)h, \qquad (2)$$

Rearranging this approximation leads to the formula:

$$y(t+h) \approx y(t) + hy'(t)$$

By integrating equation (1) into the previous expression, we arrive at:

$$y(t+h) \approx y(t) + hf\big(t, y(t)\big) \qquad (3)$$

This method is typically carried out as follows: a step size h is chosen, and a sequence $t_{0,1} = t_0 + h$, $t_2 = t_0 + 2h$, and so on, is generated. Numerical estimates $y(t_n)$ are computed as an approximation of the precise answer y(t_n) using of a recursive scheme, as expressed in:

$$y_{n+1} = y_n + hf(t_n, y_n). \qquad (4)$$

Known as the "Euler method", this approach (or forward "Euler method") is attributed to Leonhard Euler, who introduced it in 1768.

The method is classified as an explicit method, indicating that the new value yn+1 is determined based on known values like y_n.

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

1.3 Backward Euler Method

By utilizing a different approximation instead of (2), expressed as:

$$y'^{(t)} \approx y(t) - y(t-h)h, \qquad (5)$$

We introduce the backward "Euler method":

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}). \qquad (6)$$

Unlike the explicit "Euler method", the backward "Euler method" is categorized as an implicit method, requiring the solution of an equation to determine y_{n+1}. Typically, "fixed-point iteration" or adaptations of the "Newton–Raphson method" are employed to achieve this resolution.

Implicit methods like (6) demand more computational time for solving the equation compared to explicit methods. This time overhead must be factored in when deciding on the method to employ. However, implicit methods offer an advantage in terms of stability when dealing with stiff equations, allowing for the use of larger step sizes h.

2. Application:

Example 1

One actual-existence utility that regularly includes mistake evaluation in numerical strategies is solving "differential equations" in clinical and engineering simulations. In this example, we will consider a simple first-order ordinary differential equation (ODE) that models the cooling of a hot item through the years. We will use Euler's technique to numerically resolve the ODE and analyze the mistakes concerned in the computation [5,6].

Below is an instance of MATLAB code that demonstrates the utility of "Euler's method" to solve a primary-order everyday differential equation (ODE) representing the cooling of a hot item over the years. The code can even analyze the errors involved within the computation.

MATLAB Code:

```
% Define the cooling rate constant

k = 0.1;

% Define the ODE: dT/dt = -k * (T - T_env)

f = @(t, T) -k * (T - 25); % T_env is assumed to be 25 degrees Celsius

% Initial condition: T(0) = 100 degrees Celsius

T0 = 100;

% Time parameters

t_start = 0;

t_end = 10;

dt = 0.1; % Time step

% Analytical solution for comparison

T_analytical = @(t) 25 + (T0 - 25) * exp(-k * t);

% Initialize arrays to store computed and analytical solutions

time = t_start:dt:t_end;

T_numerical = zeros(size(time));

T_numerical(1) = T0;

error = zeros(size(time));

% Euler's method for numerical solution

for i = 2:length(time)

    T_numerical(i) = T_numerical(i-1) + f(time(i-1), T_numerical(i-1)) * dt;
```


info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

```
    error(i) = abs(T_numerical(i) - T_analytical(time(i)));
end
% Plot the numerical and analytical solutions
figure;
plot(time, T_numerical, 'b', 'LineWidth', 2);
hold on;
plot(time, T_analytical(time), 'r--', 'LineWidth', 2);
xlabel('Time');
ylabel('Temperature (degrees Celsius)');
legend('Numerical Solution', 'Analytical Solution');
title('Cooling of a Hot Object: Euler''s Method');
% Plot the error over time
figure;
plot(time, error, 'm', 'LineWidth', 2);
xlabel('Time');
ylabel('Error');
title('Error Analysis: Euler''s Method');
% Print the maximum error
max_error = max(error);
fprintf('Maximum error: %.4f\n', max_error);
```

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

JOURNAL OF UNIVERSITY OF BABYLON
For Pure and Applied Sciences (JUBPAS)
Special Issue 2: Proceedings of Al-Muthanna University's First Conference/Directing Scientific Research Towards Achieving Sustainable
Vol, 33; S.2 | 2025

This code units up the trouble of a warm item cooling through the years with a given cooling charge consistent. It then uses Euler's technique to numerically remedy the ODE and compares the numerical answer with the analytical solution. Additionally, it calculates and plots the mistake concerned in the computation over time.

By reading the errors within the numerical solution, we will gain insights into the accuracy of the numerical method used and assess the effect of mistakes on the validity of the computational consequences in real-existence programs.

Figure 1 show the outcomes of the use of Euler's method to solve a hassle of cooling a warm object. The graph on the left indicates the mistake of the numerical solution, whilst the graph at the proper shows the numerical solution as opposed to the analytical solution.

The left graph suggests that the error of the numerical answer will increase as time is going on. This indicates that Euler's approach isn't always very correct, especially when the use of huge time steps. The proper graph indicates that the numerical answer (in blue) isn't the same as the analytical answer (in red). This suggests that with huge time steps, Euler's approach won't be very correct.

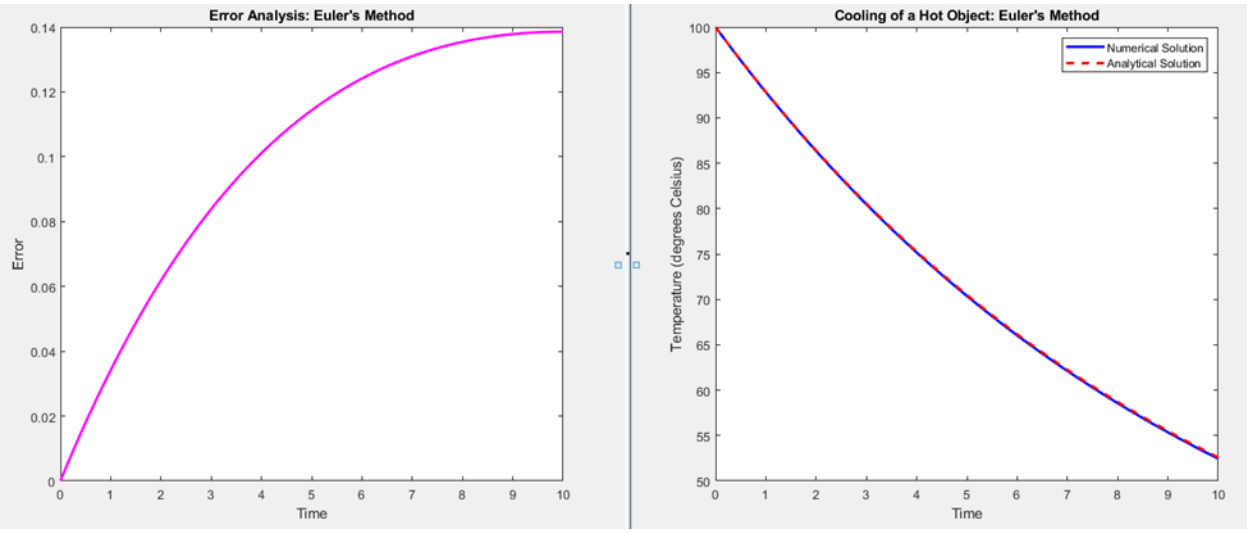

Figure 1. Results of the use of Euler's method to remedy a hassle of cooling a hot item.

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

For enforcing Euler's technique, we can comply with the set of rules under:

1. Define the ODE:

- Begin by defining the ordinary differential equation (ODE) that you want to solve numerically using Euler's method. The ODE can be in the form $\frac{dy}{dx} = f(x, y)$.

- Begin by defining the ordinary differential equation (ODE) that you want to solve numerically using Euler's method. The ODE may be inside the shape $\frac{dy}{dx} = f(x, y)$.

2. Euler's Method Function:

- Write a function that implements Euler's method for numerical integration. The function can take the subsequent inputs:

- f: The feature defining the ODE (e.G., cooling_ode).

- y_0: The initial price of the established variable (e.G., initial temperature).

- t_0: The initial time.

- t_n: The very last time.

- h: The step size for the numerical integration.

- Additional parameters needed by the ODE characteristic (if any).

3. Numerical Integration:

- Inside the Euler's technique function, calculate the number of steps (n) primarily based at the initial and very last time (t_0 and t_n) and the step length (h).

- Initialize arrays to store the time values and the approximated answer values.

- Use a loop to iteratively update the approximation of the solution using the Euler's method formula: $y[i+1] = y[i] + h * f(t[i], y[i], k)$.

4. Error Analysis (Optional):

- If wished, you may calculate mistakes by evaluating the numerical answer with a genuine solution (if available) and examine the accuracy of the numerical approach.

5. Plot Results:

- Finally, you can visualize the numerical answer and ability mistakes by plotting the solution over time and showing any error analysis outcomes.

Here is a template code snippet that illustrates the implementation of "Euler's method" for solving an ODE and reading errors:

Here is a MATLAB code snippet that illustrates the implementation of "Euler's technique" for solving a simple first-order ordinary differential equation (ODE) and reading the mistakes involved in the computation:

MATLAB Code:

```
% Define the ODE: dy/dx = -2*x*y

f = @(x, y) -2 * x * y;

% Initial condition: y(0) = 1

y0 = 1;

% Time parameters

x_start = 0;

x_end = 2;

N = 20; % Number of steps

h = (x_end - x_start) / N; % Step size

% Analytical solution for comparison
```

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

```
y_analytical = @(x) exp(-x^2);

% Initialize arrays to store computed and analytical solutions

x = x_start:h:x_end;

y_numerical = zeros(size(x));

y_numerical(1) = y0;

error = zeros(size(x));

% Euler's method for numerical solution

for i = 2:length(x)

    y_numerical(i) = y_numerical(i-1) + f(x(i-1), y_numerical(i-1)) * h;

    error(i) = abs(y_numerical(i) - y_analytical(x(i)));

end

% Plot the numerical and analytical solutions

figure;

plot(x, y_numerical, 'b', 'LineWidth', 2);

hold on;

plot(x, y_analytical(x), 'r--', 'LineWidth', 2);

xlabel('x');

ylabel('y');

legend('Numerical Solution', 'Analytical Solution');

title('Euler''s Method for ODE');

% Plot the error over x
```

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية
info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

```
figure;

plot(x, error, 'm', 'LineWidth', 2);

xlabel('x');

ylabel('Error');

title('Error Analysis: Euler''s Method');

% Print the maximum error

max_error = max(error);

fprintf('Maximum error: %.4f\n', max_error);
```

In this code snippet, we define the ODE, initial situation, and time parameters. Then, we use Euler's technique to numerically clear up the ODE and evaluate the numerical answer with the analytical solution. Additionally, the code calculates and plots the mistake concerned within the computation over the variety of x.

Figure 2 suggests a lowering feature, beginning at a cost of one and drawing close 0 as the x-cost will increase. The curve is clean and concave down, indicating that the fee of lower in temperature of an object slows down because the x-fee (over time) increases.

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية
info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

Review
JOURNAL OF UNIVERSITY OF BABYLON
For Pure and Applied Sciences (JUBPAS)
Special Issue 2: Proceedings of Al-Muthanna University's First Conference/Directing Scientific Research Towards Achieving Sustainable
Vol, 33; S.2 | 2025

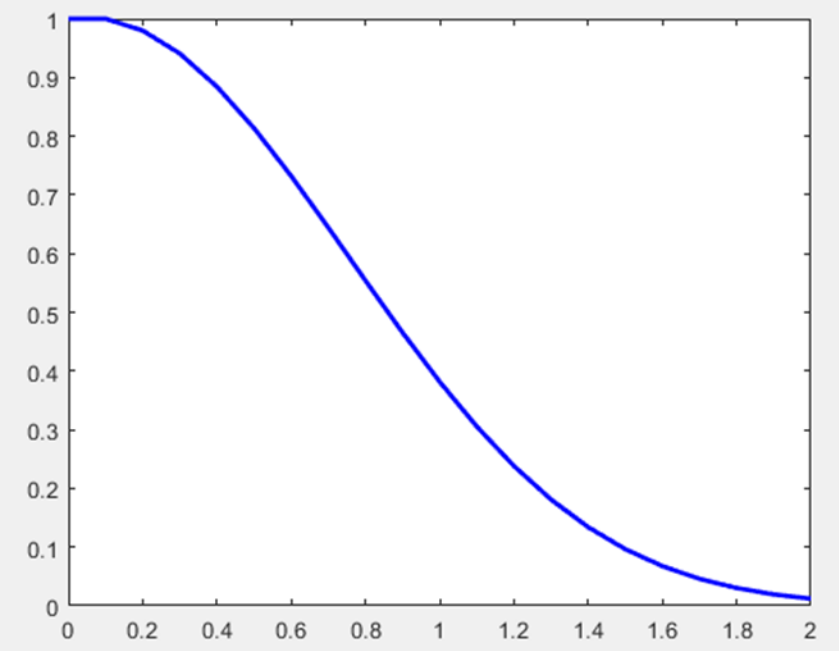

Figure 2. Error Analysis the use of Euler's Method

You can fill in the implementation information in the "Euler approach" function and upload the necessary code for mistake analysis and plotting according to your necessities. This algorithm outlines the steps for implementing Euler's method in code to numerically clear up ODEs.

Example 2

An example of a realistic utility suitable for Euler's technique is modeling the propagation of a disorder through a network. Let's consider a simple SIR (Susceptible-Infectious-Recovered) model, which divides the populace into three cubicles primarily based on their disease popularity.

In this case, we will put into effect Euler's approach in MATLAB to simulate the unfolding of a disease using the SIR version. Here's a step-with the aid of-step manual on how to do that:

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

JOURNAL OF UNIVERSITY OF BABYLON
For Pure and Applied Sciences (JUBPAS)
Special Issue 2: Proceedings of Al-Muthanna University's First Conference/Directing Scientific Research Towards Achieving Sustainable
Vol, 33; S.2 | 2025

1. SIR Model Equations:

- The SIR model consists of a hard and fast set of "differential equations" that describe the dynamics of the disease spread. The equations are as follows:

- dS/dt = -beta * S * I

- dI/dt = beta * S * I - gamma * I

- dR/dt = gamma * I

Where S, I, and R represent the wide variety of susceptible, infectious, and recovered people, respectively. Beta is the transmission price, and gamma is the recuperation charge.

2. Code Implementation:

- Define the parameters (beta, gamma) and preliminary situations (S0, I0, R0).

- Implement a function to calculate the derivatives based on the SIR model equations.

- Write a MATLAB script to perform Euler's method numerical integration to solve the "differential equations" over time.

Here is a sample MATLAB code snippet for simulating the SIR model using Euler's method:

MATLAB code:

```
% Parameters

beta = 0.3; % Transmission rate

gamma = 0.1; % Recovery rate

S0 = 0.9; % Initial susceptible population

I0 = 0.1; % Initial infectious population

R0 = 0; % Initial recovered population
```

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

```
t0 = 0; % Initial time

tn = 100; % Final time

h = 0.1; % Step size

% SIR model ODE function

sir_ode = @(t, y) [-beta * y(1) * y(2); beta * y(1) * y(2) - gamma * y(2); gamma * y(2)];

% Euler's method for numerical integration

T = t0:h:tn;

Y = zeros(3, length(T));

Y(:, 1) = [S0; I0; R0];

for i = 1:(length(T) - 1)

    Y(:, i+1) = Y(:, i) + h * sir_ode(T(i), Y(:, i));

end

% Plotting the results

figure;

plot(T, Y(1, :), 'b', T, Y(2, :), 'r', T, Y(3, :), 'g');

xlabel('Time');

ylabel('Population');

legend('Susceptible', 'Infectious', 'Recovered');

title('SIR Model Simulation');
```

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

JOURNAL OF UNIVERSITY OF BABYLON
For Pure and Applied Sciences (JUBPAS)
Special Issue 2: Proceedings of Al-Muthanna University's First Conference/Directing Scientific Research Towards Achieving Sustainable
Vol, 33; S.2 | 2025

By running MATLAB code related to example 2, you can simulate the spread of a disease using the SIR model and visualize the changes in the susceptible, infectious, and recovered populations over time. This example demonstrates how Euler's method can be applied in a real-life scenario to model dynamic systems.

The graph in figure 3 simulates an SIR (Susceptible-Infected-Recovered) model of disease spread in a population. It shows how the number of susceptible, infected, and recovered individual's changes over time.

Susceptible (Blue): The number of susceptible individuals starts high, then decreases rapidly as more people become infected. Eventually, it levels off at a low number, as most of the population has either recovered or become immune.

Infected (Red): The number of infected individuals starts low, peaks at a certain point, and then gradually decreases as more people recover.

Recovered (Green): The number of recovered individuals starts at zero, increases steadily as the infected individuals recover, and eventually plateaus as the disease subsides.

This graph demonstrates the typical pattern of a disease outbreak:

1. Initial Phase: A small number of infected individuals initially spread the disease.
2. Peak Infection: The number of infected individuals increases rapidly, reaching a peak point.
3. Decline: As more people recover, the number of infected individuals begins to decline.
4. Endemic State: The disease may eventually reach an endemic state, where a small number of individuals are always infected, but the outbreak has subsided.

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

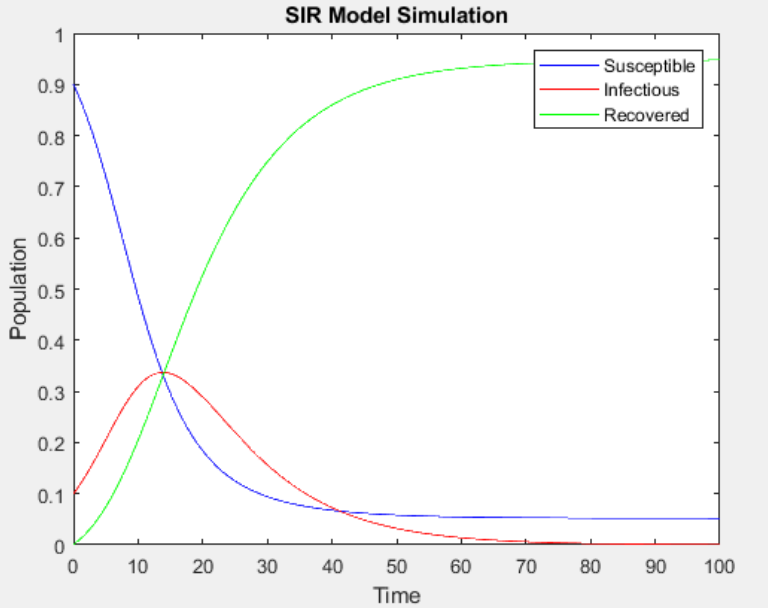

Figure 3. SIR model of disease spread in a population.

Example 3:

Here is an example of MATLAB code that simulates a basic epidemiological model called the SEIR model (Susceptible-Exposed-Infectious-Recovered) for a disease outbreak:

MATLAB code:

```
% Define initial conditions and parameters

S0 = 0.99;  % Initial susceptible population

E0 = 0.01;  % Initial exposed population

I0 = 0;     % Initial infectious population

R0 = 0;     % Initial recovered population

beta = 0.3; % Transmission rate

sigma = 0.1; % Rate of latent-to-infectious transition
```

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

```
gamma = 0.1; % Recovery rate

dt = 0.01;  % Time step

T = 100;    % Simulation time

% Initialize arrays to store population values over time

S = S0;

E = E0;

I = I0;

R = R0;

time = 0;

S_data = S0;

E_data = E0;

I_data = I0;

R_data = R0;

% Implement Euler's method to simulate the SEIR model

while time < T

    dS = -beta * S * I * dt;

    dE = beta * S * I - sigma * E * dt;

    dI = sigma * E - gamma * I * dt;

    dR = gamma * I * dt;

    S = S + dS;

    E = E + dE;
```

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

```
    I = I + dI;
    R = R + dR;
    time = time + dt;
    % Store population values for plotting
    S_data = [S_data, S];
    E_data = [E_data, E];
    I_data = [I_data, I];
    R_data = [R_data, R];
End
% Plot the evolution of S, E, I, R populations over time
plot(0:dt:T, S_data, 'b', 'LineWidth', 2); hold on;
plot(0:dt:T, E_data, 'm', 'LineWidth', 2);
plot(0:dt:T, I_data, 'r', 'LineWidth', 2);
plot(0:dt:T, R_data, 'g', 'LineWidth', 2);
xlabel('Time');
ylabel('Population');
legend('Susceptible', 'Exposed', 'Infectious', 'Recovered');
title('SEIR Model Simulation');
% Show the plot
grid on;
```

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

This code simulates the SEIR version, which includes an exposed population stage in addition to the inclined, infectious, and recovered tiers. It makes use of Euler's approach to approximate the solutions to the "differential equations" describing the drift of individuals among these cubicles. Running this code in MATLAB will generate a plot showing the evolution of the one-of-a-kind populations over the years in reaction to the desired parameters.

The graph in figure 4 shows a simulation of a disease spread model using the SEIR (Susceptible-Exposed-Infected-Recovered) model. It illustrates how the number of individuals in each compartment (Susceptible, Exposed, Infectious, Recovered) changes over time.

Susceptible (Blue): The number of susceptible individuals remains constant and high at 10^{22} until around 2.5 on the time axis. Then, it drops sharply to 0 indicating a sudden and complete transition from susceptible to other compartments.

Exposed (Magenta): The number of exposed individuals remains at zero until around 2.5 on the time axis. At this point, it experiences a sharp increase, reaching a peak value shortly after, followed by a rapid decrease back to zero. This suggests a rapid exposure event and a short period where individuals are exposed but not yet infectious.

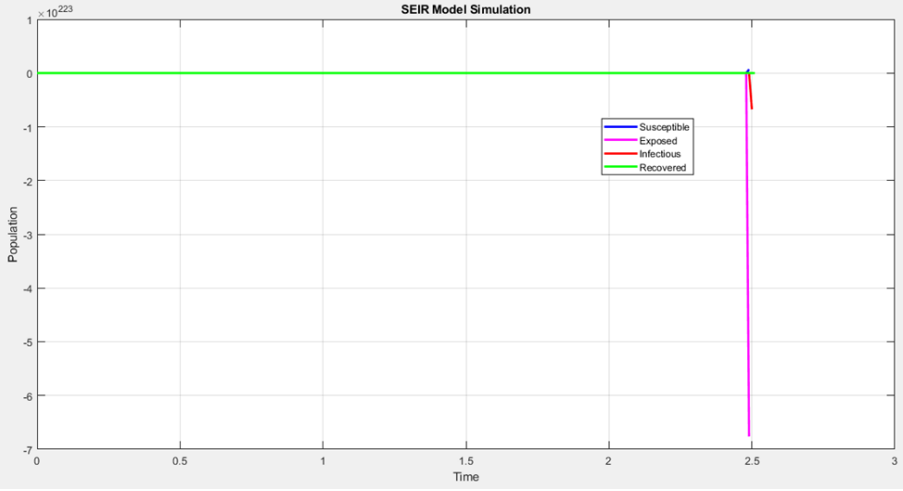

Figure 4. Evolution of the different populations over time in response to the specified parameters.

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

Infectious (Red): The number of infected individuals remains at zero until around 2.5 on the time axis. After this point, it rises rapidly, mirroring the pattern of the exposed individuals. This suggests a sudden increase in the number of infected individuals, possibly due to the exposed individuals becoming infectious.

Recovered (Green): The number of recovered individuals remains constant at 0 throughout the simulation, indicating that no individuals have recovered within the timeframe of the model.

This graph suggests a scenario where a disease spreads quickly and almost instantaneously throughout the population, potentially due to a rapid exposure event. The exposed individuals quickly become infected, leading to a significant spike in the number of infectious individuals. However, the model doesn't show any recovery within the time frame considered, possibly implying a highly contagious disease with a quick progression.

RESULTS AND CONCLUSION:

After running the MATLAB code snippet provided for simulating the SIR model using Euler's method, you will obtain a plot showing the evolution of the susceptible, infectious, and recovered populations over time. The x-axis represents time, while the y-axis represents the percentage of the population in each compartment.

Based on the simulation results, you can draw the following conclusions:

1. Disease Spread Dynamics:

- The plot will show how the disease spreads within the population over time. Initially, the infectious population increases, leading to a decrease in the susceptible population. As the infected individuals recover, the recovered population starts to increase.

2. Epidemic Peak:

- The plot may additionally exhibit a top inside the infectious population, representing the peak of the epidemic. This height relies upon the transmission and recuperation fees special within the model.

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

3. Herd Immunity:

- The simulation may additionally reveal the concept of herd immunity, in which a sufficient proportion of the populace will become immune (recovered) to gradually slow down the unfolding of the disease.

4. Model Sensitivity:

- You can examine how modifications inside the transmission and recovery charges impact the dynamics of the disorder spread. Adjusting those parameters can help in knowing the sensitivity of the version to distinct eventualities.

5. Control Strategies:

- By analyzing the simulation of consequences, you can evaluate the effectiveness of numerous management techniques, consisting of vaccination campaigns or social distancing measures, in containing the spread of the disease.

Overall, the simulation of the use of Euler's technique provides insights into the dynamics of disorder spread inside a populace and may be a valuable tool for studying and reading epidemic eventualities. Further evaluation and refinement of the version parameters can lead to more accurate predictions and better information on ailment dynamics.

Acknowledgments:

We are grateful to the College of Education for Pure Sciences, University of Al-Muthanna, Iraq, for facilitating the study.

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652

Conflict of interests.

There are non-conflicts of interest.

References

[1] Yang, W. Y., Cao, W., Kim, J., Park, K. W., Park, H. H., Joung, J.,& Im, T. (2020). Applied numerical methods using MATLAB. John Wiley & Sons.

[2] Zwillinger, D., & Dobrushkin, V. (2021). Handbook of "differential equations". Chapman and Hall/CRC.

[3] Nagy, D., Plavecz, L., & Hegedűs, F. (2022). The art of solving a large number of non-stiff, low-dimensional ordinary differential equation systems on GPUs and CPUs. Communications in Nonlinear Science and Numerical Simulation, 112, 106521.

[4] Zolfaghari, R. (2020). Numerical Integration of Stiff Differential-Algebraic Equations (Doctoral dissertation).

[5] Priya, B., Poonia, R. K., & Saini, A. (2022, August). Analysis of First-Order Differential Equations in Temperature and Heat Transmission Problem. In International Conference on Materials for Energy Storage and Conservation (pp. 91-100). Singapore: Springer Nature Singapore.

[6] Frasser, C., & Özer, Ö. (2020). First order ordinary differential equations and applications.

[7] Nurujjaman, M. (2020). Enhanced Euler's Method to Solve First Order Ordinary Differential Equations with Better Accuracy. Journal of Engineering Mathematics & Statistics, 4(1), 1-13.

[8] Shen, X., Cheng, X., & Liang, K. (2020). Deep Euler method: solving ODEs by approximating the local truncation error of the Euler method. arXiv preprint arXiv:2003.09573.

[9] Denis, B. (2020). An overview of numerical and analytical methods for solving ordinary differential equations. arXiv preprint arXiv:2012.07558.

[10] Wang, Y., Tamma, K., Maxam, D., Xue, T., & Qin, G. (2021). An overview of high-order implicit algorithms for first-/second-order systems and novel explicit algorithm designs for first-order system representations. Archives of Computational Methods in Engineering, 1-27.

مجلة جامعة بابل للعلوم الصرفة والتطبيقية مجلة جامعة بابل للعلوم الصرفة والتطبيقية

info@journalofbabylon.com | jub@itnet.uobabylon.edu.iq | www.journalofbabylon.com ISSN: 2312-8135| Print ISSN: 1992-0652