Simulating Epidemic Spread Using the SIR Model

Submission by:

- ***Maaz Saeed - 2021268***
- ***Khushal Khan - 2021261***

Overview

Project Statement

The project simulates epidemic modeling using the SIR model, solving its Ordinary Differntial Equations (ODEs) numerically to analyze the effects of transmission rate (β), recovery rate (γ) on population.

The SIR Ordinary Differential Equations

The ordinary differential equations (ODEs) for the SIR model are as follows:

$$\frac{dS}{dt} = -\beta SI$$
$$\frac{dI}{dt} = \beta SI - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

In the first ODE $\frac{dS}{dt}$ refers to the change in susceptible with respect to time. β is the transmission rate, S and I are the fraction of population that are susceptible and infected respectively. As susceptible are infected the number of susceptible fraction falls, hence the negative sign.

In the second ODE $\frac{dI}{dt}$ refers to the rate of change of infected population. The first term captures new infections due to interactions between susceptible and infected individuals, while the second term $-\gamma$I, represents the individuals recovering.

In the third ODE $\frac{dR}{dt}$ represents the rate of recovery.

The equations are coupled, that describe the rate of change of S, I, and R. These ODEs depend on the the values of each other, which evolve over time.

The Euler Equations for SIR Model

The Euler method is applied to each of the ODE within a single time step to get new values of *S*, *I*, and *R*.

As time progresses the infection begins to spread making more people susceptible, who may contract the infection and thus go into recovery state The Euler equations are as follows:

$$\begin{aligned} S(t+h) &= S(t) - h \cdot \beta S(t)\, I(t) \\ I(t+h) &= I(t) + h \cdot (\beta S(t)\, I(t) - \gamma I(t)) \\ R(t+h) &= R(t) + h \cdot \gamma I(t) \end{aligned}$$

Code and Simulation (Euler Method)

```
euler_SIR_step (generic function with 1 method)
```

```julia
                        #Transmission Rate, Recovery Rate, Initial Population, Step Size
function euler_SIR_step(β, γ, sir_0::Vector, h::Number)
    s, i, r = sir_0 # Unpack the fraction of populations (S, I, and R)respectively.

    return [
        s - h*β*s*i,
        i + h*(β*s*i - γ*i),
        r + h*γ*i,
    ]
end
```

```
plot_sir! (generic function with 1 method)
```

```julia
function plot_sir!(T, results; label="", caption="", kwargs...)

    p = plot()

    s = getindex.(results, [1])
    i = getindex.(results, [2])
    r = getindex.(results, [3])


    plot!(p, T, s; color=:orange, label=label*" Susceptible", lw=3, kwargs..., xlabel="Time", ylabel="Population Fraction")
    plot!(p, T, i; color=:red, label=label*" Infected", lw=3, kwargs...)
    plot!(p, T, r; color=:green, label=label*" Recovered", lw=3, kwargs...)

    title!(caption)


end
```

```
euler_SIR (generic function with 1 method)
```

```julia
function euler_SIR(β, γ, sir_0::Vector, T::AbstractRange)
    h = step(T)

    num_steps = length(T)

    return [sir_0=euler_SIR_step(β, γ, sir_0, h) for i=1:num_steps]
end
```

Parameters Sensitivity - Euler

S: 0.72

I: 0.09

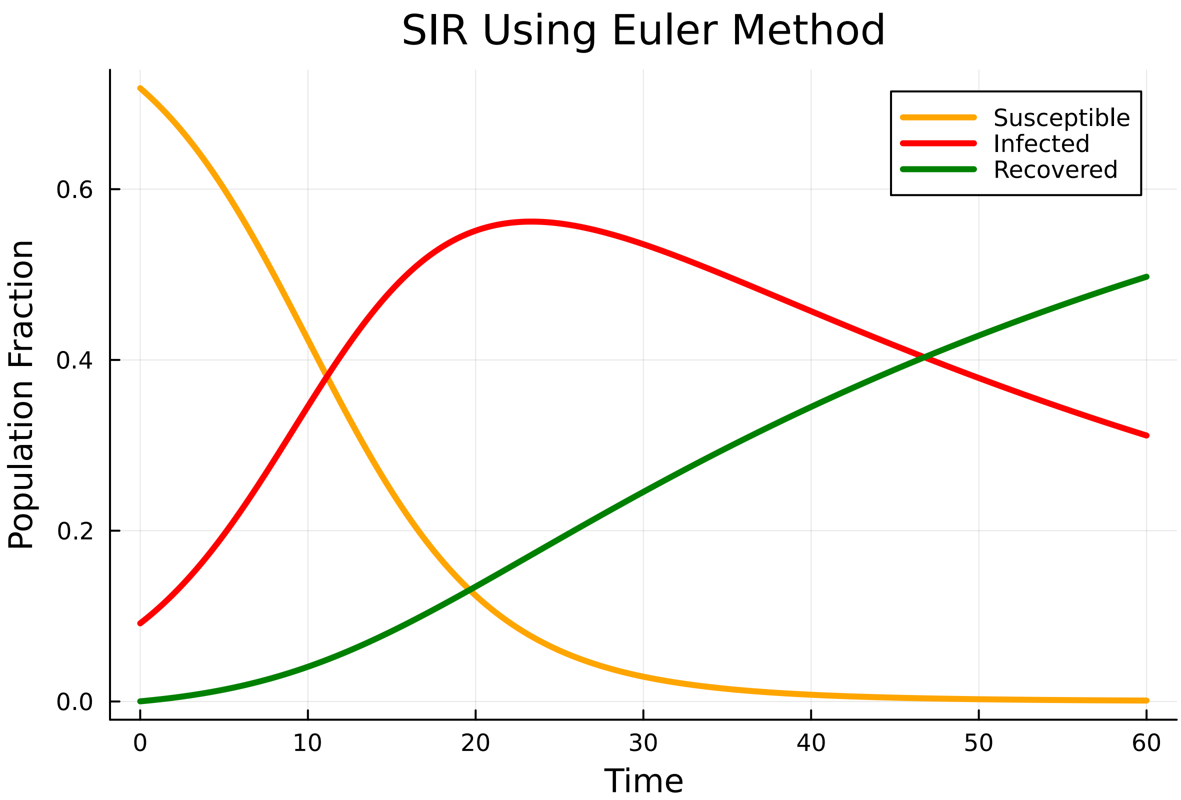
R: 0.0

Transmission Rate

β: 0.26

Recovery Rate

γ: 0.02




```
plot_sir!(0:0.1:60.0, euler_SIR(β, γ, [S, I, R], 0 : 0.1 : 60.0), caption="SIR Using Euler Method")
```

Runge-Kutta Method (RK4) Equations

Fourth-order Runge-Kutta implementation of the SIR (Susceptible-Infected-Recovered) epidemic model that computes the next state of the system using time step `h`.

RK4 slope calculations for each variable:

For Susceptible (S):

$$
\begin{aligned}
Sk_1 &= h \cdot DS(\beta, s, i) \\
Sk_2 &= h \cdot DS(\beta, s + h/2, i + Sk_1/2) \\
Sk_3 &= h \cdot DS(\beta, s + h/2, i + Sk_2/2) \\
Sk_4 &= h \cdot DS(\beta, s + h, i + Sk_3)
\end{aligned}
$$

For Infected (I):

$$
\begin{aligned}
Ik_1 &= h \cdot DI(\beta, \gamma, s, i) \\
Ik_2 &= h \cdot DI(\beta, \gamma, s + h/2, i + Ik_1/2) \\
Ik_3 &= h \cdot DI(\beta, \gamma, s + h/2, i + Ik_2/2) \\
Ik_4 &= h \cdot DI(\beta, \gamma, s + h, i + Ik_3)
\end{aligned}
$$

For Recovered (R):

$$
\begin{aligned}
Rk_1 &= h \cdot DR(\gamma, i) \\
Rk_2 &= h \cdot DR(\gamma, i + Ik_1/2) \\
Rk_3 &= h \cdot DR(\gamma, i + Ik_2/2) \\
Rk_4 &= h \cdot DR(\gamma, i + Ik_3)
\end{aligned}
$$

Next state is computed as:

$$S_{next} = s + \frac{1}{6}(Sk_1 + 2Sk_2 + 2Sk_3 + Sk_4)$$
$$I_{next} = i + \frac{1}{6}(Ik_1 + 2Ik_2 + 2Ik_3 + Ik_4)$$
$$R_{next} = r + \frac{1}{6}(Rk_1 + 2Rk_2 + 2Rk_3 + Rk_4)$$

Code and Simulation (RK4 Method)

RK4_SIR_step (generic function with 1 method)

```
begin
    # The three ODEs
    DS(β, s, i) = -β*s*i
    DI(β, γ, s, i) = β*s*i - γ*i
    DR(γ, i)    = γ*i
                                   # Transmission rate, Recovery rate, initial SIR, conditions, Step size
function RK4_SIR_step(β, γ, initial_conditions::Vector, h)
    s, i, r = initial_conditions # Unpack!

    # Compute Slopes for S
    Sk₁ = h * DS(β, s        , i          )
    Sk₂ = h * DS(β, s + h / 2, i + Sk₁ / 2)
    Sk₃ = h * DS(β, s + h / 2, i + Sk₂ / 2)
    Sk₄ = h * DS(β, s + h    , i + Sk₃    )

    # Compute Slopes for I
    Ik₁ = h * DI(β, γ, s        , i          )
    Ik₂ = h * DI(β, γ, s + h / 2, i + Ik₁ / 2)
    Ik₃ = h * DI(β, γ, s + h / 2, i + Ik₂ / 2)
    Ik₄ = h * DI(β, γ, s + h    , i + Ik₃    )

    # Compute Slopes for R
    Rk₁ = h * DR(γ, i          )
    Rk₂ = h * DR(γ, i + Ik₁ / 2)
    Rk₃ = h * DR(γ, i + Ik₂ / 2)
    Rk₄ = h * DR(γ, i + Ik₃    )

    # Next state
    S_slopes_sum = Sk₁+ 2 * Sk₂ + 2 * Sk₃ + Sk₄
    S_next = s + (1 / 6) * S_slopes_sum

    I_slopes_sum = Ik₁+ 2 * Ik₂ + 2 * Ik₃ + Ik₄
    I_next = i + (1 / 6) * I_slopes_sum

    R_slopes_sum = Rk₁ + 2 * Rk₂ + 2 * Rk₃ + Rk₄
    R_next = r + (1 / 6) * R_slopes_sum


    return [S_next, I_next, R_next]

end

end
```

```
RK4_SIR (generic function with 1 method)
```

```julia
function RK4_SIR(β, γ, sir_0::Vector, T::AbstractRange)

    h = step(T)

    num_steps = length(T)

    return [sir_0=RK4_SIR_step(β, γ, sir_0, h) for i=1:num_steps]
end
```

Parameters Sensitivity - RK4

S: 0.97

I: 0.05

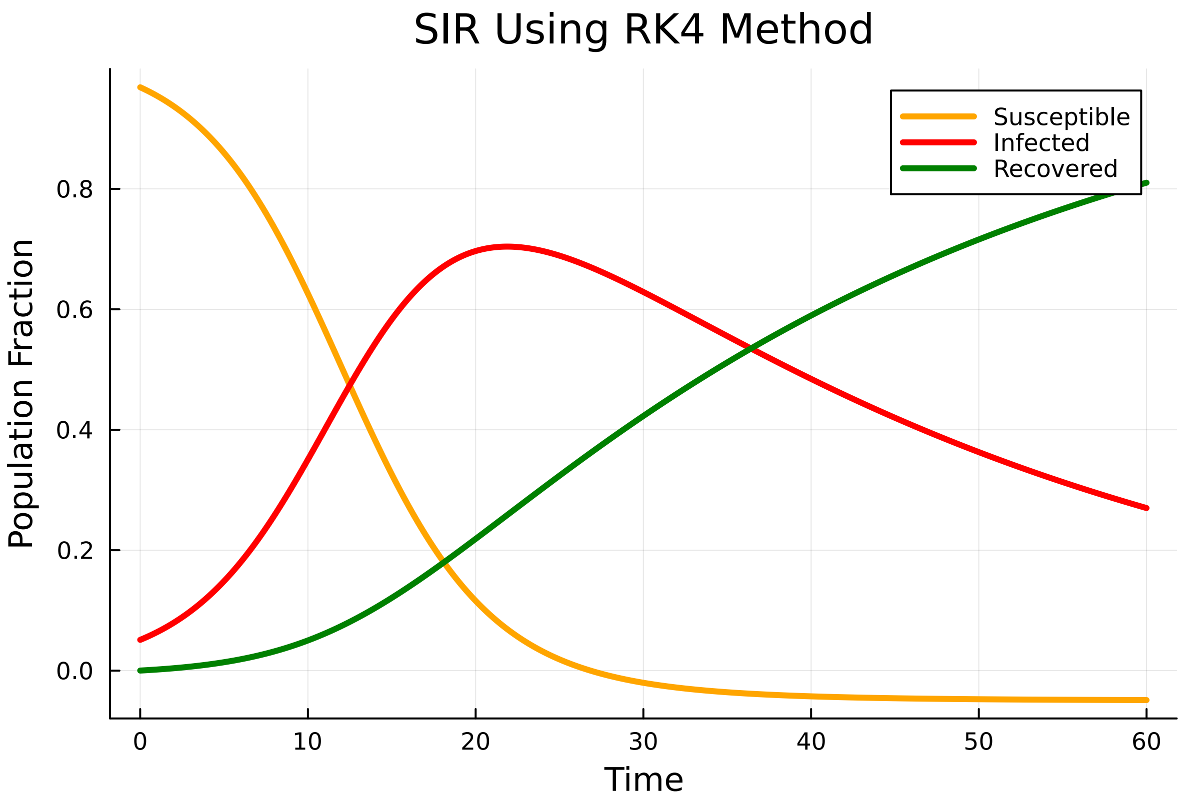
R: 0.0

Transmission Rate

β: 0.25

Recovery Rate

γ: 0.03




```julia
plot_sir!(0:0.1:60.0, RK4_SIR(β2, γ2, [S2, I2, R2], 0 : 0.1 : 60.0), caption="SIR Using RK4 Method")
```

Results and Discussion

In the initial stage, the susceptible population is exposed to the infection, causing the susceptible graph (orange) to decline as individuals transition from S to I. This leads to a rise in the infection graph (red).

The infection graph peaks when the majority of the population contracts the virus, marking the peak infection point.

As infected individuals transition to recovery, the recovery graph (green) rises, while the infection graph decays, eventually stabilizing as the epidemic concludes.

Interventions like physical distancing, lockdowns, and vaccination can help flatten the infection curve and lead to premature conclusion of the epidemic.

The model does not take into account the possibility of reinfection.

Summary

In summary, the population transitions from susceptible (S) to infected (I) and then to recovery (R), with the graphs rising and falling accordingly. The trends align with expected epidemic behavior without interventions.

References

1. MIT Computational Thinking - 18.S191