Lockdown optimisation on an SIR model using JuMP.jl

Initial version here by Simon Frost (@sdwfrost)
Current version Sandra Montes (@slmontes), 2025-03-10

Introduction

This example explores the optimal control of an SIR (Susceptible-Infected-Recovered) model using a time-varying intervention that reduces the infection rate. The population is divided into three categories: susceptible individuals (`S`), infected individuals (`I`), and the total number of cases (`C`). The intervention is modelled as a time-dependent control variable `(t)` that reduces the transmission rate by a factor of `1 - (t)`. The goal is to determine the optimal timing and application of this intervention to minimise the final number of cases (`C`) under the following constraints: (a) cannot exceed a maximum value, and (b) the total cost, measured as the integral of over time, must remain within a specified limit.

The model is described by the following differential equations:

$$\begin{aligned}
\frac{\mathrm{d}S}{\mathrm{dt}} &= -\beta(1-\upsilon(t))SI, \\
\frac{\mathrm{d}I}{\mathrm{dt}} &= \beta(1-\upsilon(t))SI - \gamma I, \\
\frac{\mathrm{d}C}{\mathrm{dt}} &= \beta(1-\upsilon(t))SI,
\end{aligned}$$

Here, is the transmission rate, and is the recovery rate.

In a study by Britton and Leskela (2022), it was demonstrated that the optimal strategy for controlling the epidemic under the above model involves a single lockdown at a set maximum intervention level for , sustained until the cost reaches the specified threshold. To determine whether the optimal policy can be identified numerically, we use a simple Euler discretisation and then use JuMP.jl with IPOPT to optimise.

Libraries

```
using OrdinaryDiffEq
using DiffEqCallbacks
using JuMP
using Ipopt
using Plots
using DataInterpolations
using NonlinearSolve;
```

Functions

ODE system

```
function sir_ode!(du,u,p,t)
    (S, I, C) = u
    ( ,  ,  ) = p
    @inbounds begin
        du[1] = - *(1- )*S*I
        du[2] =  *(1- )*S*I -  *I
        du[3] =  *(1- )*S*I
    end
    nothing
end;
```

SIR simulation

```
function simulate(p, u0, t , dur, ss, alg)
    t  = t  + dur
    lockdown_times = [t , t ]
     ,  ,   = p
    function affect!(integrator)
        if integrator.t < lockdown_times[2]
            integrator.p[3] = 
        else
            integrator.p[3] = 0.0
        end
    end
    cb = PresetTimeCallback(lockdown_times, affect!)
    tspan = (0.0, t +ss)
    # Start with  =0
    prob = ODEProblem(sir_ode!, u0, tspan, [ ,  , 0.0])
    sol = solve(prob, alg, callback = cb)
    return sol
end;
```

Calculate the total number of infected at the end of simulation

```
function final_size(p, u0, t , dur, ss, alg)
    sol = simulate(p, u0, t , dur, ss, alg)
    return sol[end][3]
end;
```

Running the model without intervention

Parameters

```
u0 = [0.99, 0.01, 0.0]; #S, I, C (cumulative incidence)
p = [0.5, 0.25, 0]; #  ,  ,
```

```
t0 = 0.0
tf = 100
dt = 0.1
ts = collect(t0:dt:tf)
alg = Tsit5();
```

Solve using ODEProblem

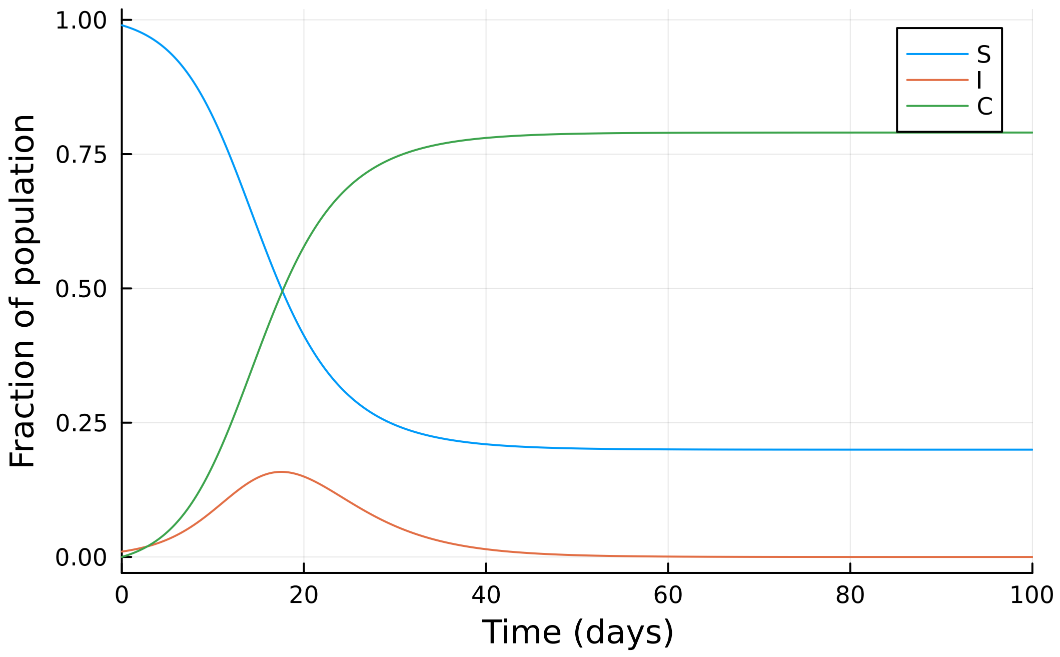
```
prob1 = ODEProblem(sir_ode!, u0, (t0, tf), p)
sol1 = solve(prob1, alg, saveat=ts);
```

Without control the final size of total number of cases is $\sim 79\%$

```
fianl_C_sol1 = sol1[end][3]
println("Cumulative incidence fraction without control: ", fianl_C_sol1)
```

```
Cumulative incidence fraction without control: 0.7901973301721557
```

```
plot(sol1,
     xlim=(0, 100),
     labels=["S" "I" "C"],
     xlabel="Time (days)",
     ylabel="Fraction of population")
```



Now we find the peak value of infected individuals and the time at which it occurs:

```
peak_value, peak_index = findmax(sol1[2, :])
peak_time = sol1.t[peak_index]
println("The peak of infections occurs at time: ", peak_time)
```

```
The peak of infections occurs at time: 17.5
```

Running the model with intervention initiated at the peak of infected cases

Demonstrating the impact of an intervention when initiated at the peak of infected cases. Parameters:

```
p2 = copy(p)
p2[3] = 0.5;    #Set   to 0.5
t  = peak_time
dur = 20.0  #Duration of the intervention
ss = 100.0;
```

Simulate with intervention

```
sol2 = simulate(p2, u0, t , dur, ss, alg);
```

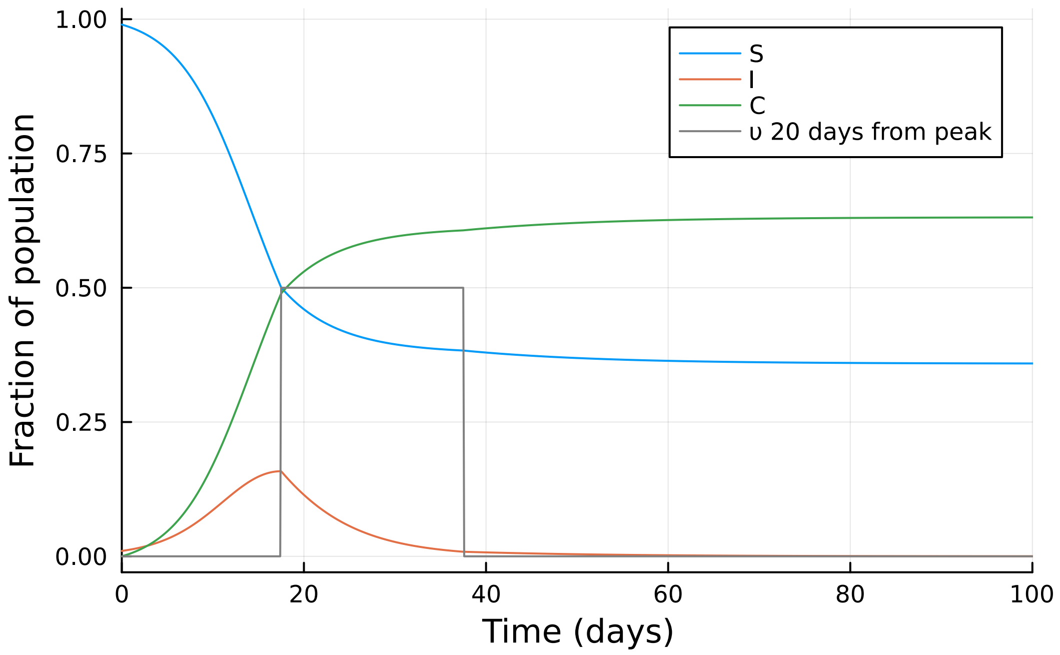
With control starting at the peak of infections and lasting 20 days, the final size of total number of cases is $\sim 63\%$

```
fianl_C_sol2 = sol2[end][3]
println("Cumulative incidence fraction when control starts at the peak: ", fianl_C_sol2)
```

```
Cumulative incidence fraction when control starts at the peak: 0.6312297734622068
```

```
# create a vector that shows the   value for each time step
 _test = zeros(length(ts))
for i in 1:length(ts)
    if ts[i] >= t  && ts[i] <= (t  + dur)
         _test[i] = 0.5  # Set to 0.5 within the time interval
    end
end
```

```
plot(sol2,
     xlim=(0, 100),
     labels=["S" "I" "C"],
     xlabel="Time (days)",
     ylabel="Fraction of population")
plot!(ts,  _test, color=:gray, label="  20 days from peak")
```



Searching for the optimal intervention time

Parameters

```
  = p2[1]
  = p2[2]
 _max = p2[3]
 _total = 10.0; # maximum cost

S0 = u0[1]
I0 = u0[2]
C0 = u0[3]

T = Int(tf/dt)

silent = true;
```

Model setup

```
model = Model(Ipopt.Optimizer)
set_optimizer_attribute(model, "max_iter", 1000)
if !silent
    set_optimizer_attribute(model, "output_file", "JuMP_lockdown.txt")
    set_optimizer_attribute(model, "print_timing_statistics", "yes")
end;
```

Variables:

```
@variable(model, 0 <= S[1:(T+1)] <= 1)
@variable(model, 0 <= I[1:(T+1)] <= 1)
@variable(model, 0 <= C[1:(T+1)] <= 1)
@variable(model, 0 <=  [1:(T+1)] <=  _max);
```

We can discretise the SIR model in different ways, but here we use a simple Euler discretisation:

```
@expressions(model, begin
        infection[t in 1:T], (1 -  [t]) *   * I[t] * dt * S[t]  # Linear approximation of i
        recovery[t in 1:T],   * dt * I[t] # Recoveries at each time step
    end);
```

```
@constraints(model, begin
    S[1]==S0
    I[1]==I0
    C[1]==C0
    [t=1:T], S[t+1] == S[t] - infection[t]
    [t=1:T], I[t+1] == I[t] + infection[t] - recovery[t]
    [t=1:T], C[t+1] == C[t] + infection[t]
    dt * sum( [t] for t in 1:T+1) <=  _total
end);
```

```
@objective(model, Min, C[T+1]);
```

```
if silent
    set_silent(model)
end
optimize!(model)
```

```
******************************************************************************
This program contains Ipopt, a library for large-scale nonlinear optimization.
 Ipopt is released as open source code under the Eclipse Public License (EPL).
         For more information visit https://github.com/coin-or/Ipopt
******************************************************************************
```

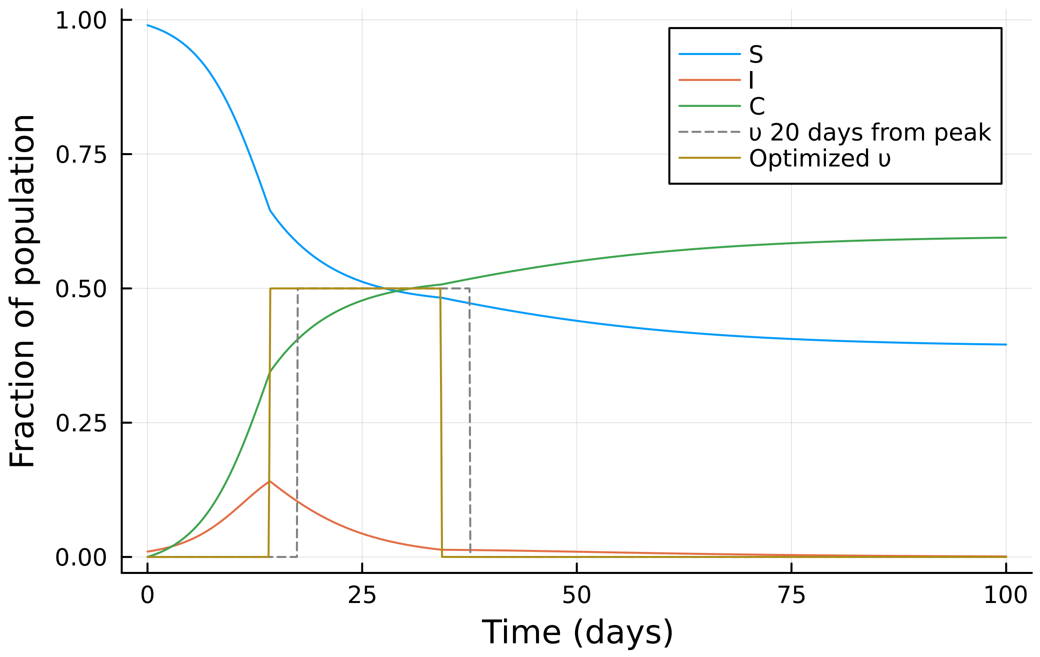
```
termination_status(model)
```

```
LOCALLY_SOLVED::TerminationStatusCode = 4
```

```
S_opt = value.(S)
I_opt = value.(I)
C_opt = value.(C)
 _opt = value.( );
```

Plotting the results shows that the optimiser has identified a policy similar to the one suggested by Britton and Leskela (2022), which is a single lockdown of intensity `_max` and a duration `_total/ _max`. But the optimiser has also identified a time to start the lockdown that is not the peak of the infection curve.

```
plot(ts, S_opt, label="S", xlabel="Time (days)", ylabel="Fraction of population")
plot!(ts, I_opt, label="I")
plot!(ts, C_opt, label="C")
plot!(ts, _test, color=:gray, linestyle=:dash, label="  20 days from peak")
plot!(ts, _opt, label="Optimized  ")
```



```
tolerance = 1e-3
max_indices = findall(x -> abs(x -  _max) < tolerance,  _opt)
max_times = ts[max_indices]
intervention_length = length(max_indices)*dt

println("Duration of intervention in days: ", intervention_length)
println("The start of intervention is at time: ",  max_times[1])
```

```
Duration of intervention in days: 19.900000000000002
The start of intervention is at time: 14.3
```

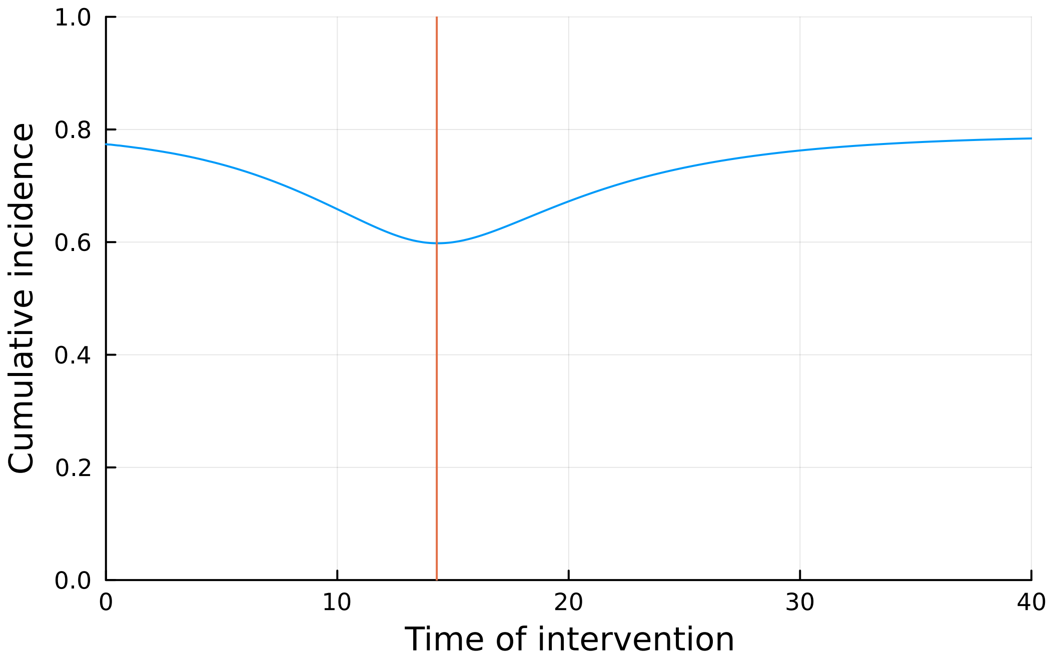
With control starting at the optimased time and lasting ~ 20 days, the final size of total number of cases is $\sim 59\%$

```
fianl_C_sol3 = C_opt[end]
println("Cumulative incidence fraction when control is optimised: ", fianl_C_sol3)
```

```
Cumulative incidence fraction when control is optimised: 0.5945130623911311
```

Now we use the `final sizes` function to produce simulations at different start times. In the following plot we can observe how the a 20-day constant-level lockdown of 0.5 the lowest Cumulative incidence is obtained at the optimal time found prevoiusly.

```
t_opt = max_times[1]
fs(u, p_) = final_size(p2, u0, u[1], dur, ss, alg);
final_sizes = [fs([x], []) for x in ts]
plot(ts,
     final_sizes,
     xlabel="Time of intervention",
     ylabel="Cumulative incidence",
     ylim=(0,1),
     xlim=(0,40),
     legend=false)
vline!([t_opt])
```



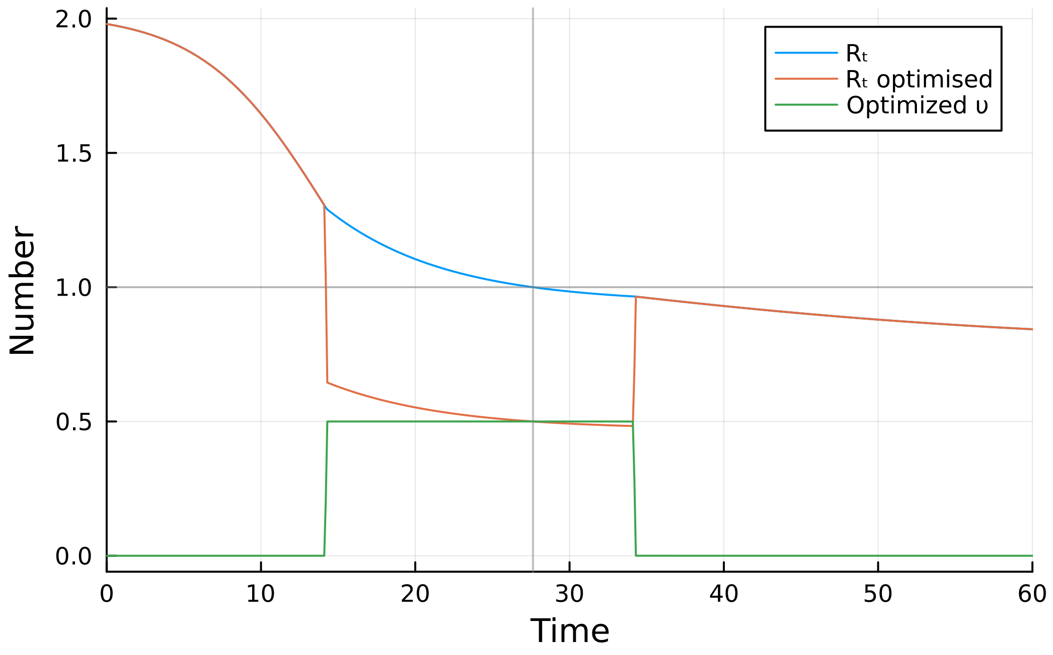
We can also calculate the effective reproductive number, `R` in the presence of the intervention

```
R _opt =  .* S_opt ./    #Not taking into account the intervention
R  _opt = R _opt .* (1 .-  _opt);  #Taking into account the intervention
```

And calculate the time at which `R ==1` using a root-finding approach:

```
R _interp = CubicSpline(R _opt,ts)
f(u, p) = [R _interp(u[1]) - 1.0]
u0 = [(tf-t0)/3]
Rtprob = NonlinearProblem(f, u0)
Rtsol = solve(Rtprob, NewtonRaphson(), abstol = 1e-9).u[1];
```

```
plot(ts, R _opt, label="R ", xlabel="Time", ylabel="Number", legend=:topright, xlim=(0,60))
plot!(ts, R _opt, label="R  optimised")
plot!(ts,  _opt, label="Optimized  ")
vline!([Rtsol], color=:gray, alpha=0.5, label=false)
hline!([1.0], color=:gray, alpha=0.5, label=false)
```



Discussion

We mentioned that different discretisation methods can be used. We also tried discretising the system using exponential approximations to model the transition probabilities. If we model the time between transitions (such as infections or recoveries) as an exponentially distributed random variable with rate , the probability of transition between states during the interval `dt` can be approximated by:

$$1 - exp(-\lambda \cdot dt),$$

For larger `dt`, this method may be more accurate than a simple Euler approximation. However, smaller timesteps, although giving results closer to the continuous time system, resulted in the solver struggling to converge. Therefore, at the chosen `dt==0.1`, the simple Euler discretisation provided better results.

We could also have used other optimisation methods by fixing the intervention length to 20 days and optimising the start of the intervention, which can be observed when we compared the optimal time obtained with the lowest cumulative incidence found simulating different intervention start times using the function `final size`. However, JuMP optimisation's advantage is that we could numerically confirm that according to the constraints set to the intervention cost and total cases, the optimal intervention was a single lockdown at the maximum level set and for a period of ~ 20 simulated days.