Lockdown optimisation on an SIR model using JuMP.jl

Simon Frost (@sdwfrost) and Sandra Montes (@slmontes)

2025-03-10

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 `υ(t)` that reduces the transmission rate by a factor of `1 - υ(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) `υ` cannot exceed a maximum value, and (b) the total cost, measured as the integral of `υ` 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, β is the transmission rate, and γ 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 `υ`, 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
    (β, γ, υ) = p
    @inbounds begin
        du[1] = -β*(1-υ)*S*I
        du[2] = β*(1-υ)*S*I - γ*I
        du[3] = β*(1-υ)*S*I
    end
    nothing
end;
```

SIR simulation

```
function simulate(p, u0, t□ , dur, ss, alg)
    t□  = t□  + dur
    lockdown_times = [t□ , t□ ]
    β, γ, υ = p
    function affect!(integrator)
        if integrator.t < lockdown_times[2]
            integrator.p[3] = υ
        else
            integrator.p[3] = 0.0
```

```
        end
    end
    cb = PresetTimeCallback(lockdown_times, affect!)
    tspan = (0.0, t□ +ss)
    # Start with υ=0
    prob = ODEProblem(sir_ode!, u0, tspan, [β, γ, 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]; # β, γ, υ
```

```
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)
```