Numerical solution of SEIRD

Introduction

Compartmental epidemiological models (such as SEIRD and its variants) are *differential equation* models—that is, given particular values of the model parameters, the models do not directly specify a functional form over time, but rather they specify the instantaneous rate of change of the system, forming a system of differential equations whose solution is the model output over time. Like many differential equation models of interest in scientific applications, SEIRD differential equations do not admit analytical solutions. Instead, an approximation of the solution must be obtained using numerical methods.

Epidemiological differential equations can be solved straightforwardly using a simple fixed time step solver such as the Euler method. In this notebook, we show that when such methods are employed without sufficient care in the selection of the time step, high error in the numerical solution may occur. We recommend more sophisticated adaptive step size solvers when solving SEIRD-type models, as implemented in the comomodels package.

This notebook begins with a review of a simple differential equation solver on a toy problem with a known solution. Then, it demonstrates the importance of accurate ODE solvers for comomodel's SEIRD model.

```
library(comomodels)
library(deSolve)
library(ggplot2)
library(tidyverse)
library(glue)
```

Numerical solution of a differential equation using forward Euler

One of the simplest numerical solvers for differential equations is the forward Euler method. This method assumes a first order differential equation

$$\frac{df}{dt} = g(t, f)$$

(note that f may be a vector-valued function), as well as an initial condition $f(t = t_0)$. Forward Euler forms an approximate solution $\{f_i\}_{i=0}^{N}$ on a set of time points $\{t_i = t_0 + i\Delta t\}_{i=0}^{N}$ which is given by:

$$f_0 = f(t = t_0);$$
$$f_{i+1} = f_i + g(t_i, f_i)\Delta t, \quad i = 1, \ldots, N-1.$$

Importance of the solver step size

Note that Δt—the time spacing between adjacent values in the approximate solution—is a critical parameter of the Euler method. While calculating the next function value f_{i+1}, the Euler method approximates the derivative $g(t, f)$ with the constant value $g(t_i, f_i)$ for the interval $(t_i, t_i + \Delta t)$. However, $g(t, f)$ typically varies over time—thus, as Δt increases, this approximation becomes worse, and this approximation error accumulates in the numerical solution and causes it to deviate from the true values.

However, obtaining a satisfactory solution to the differential equation is not as simple as setting Δt to some minuscule value, as decreasing the value of Δt increases the runtime of the solver, and for very small values of Δt, the runtime may be prohibitive. This is because solving the ODE on a time interval of length T requires $T/\Delta t$ iterations of the Forward Euler method, each iteration requiring one evaluation of the function g—smaller values of Δt thus require more calculations of g, which are potentially slow.

Effect of solver step size on solutions

We examine the numerical arising from the solver step size in the SIR model. The proportions of susceptible (S), infected (I), and recovered (R) are modelled according to the differential equations:

$$\frac{dS}{dt} = -\beta SI;$$

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

and

$$\frac{dR}{dt} = \gamma I.$$

We focus on the I compartment and consider the early phase of an epidemic where $S >> I$. Under these conditions, S is approximately constant and we have for dI/dt:

$$\frac{dI}{dt} = (\beta - \gamma)SI; \quad I(t = 0) = I_0$$

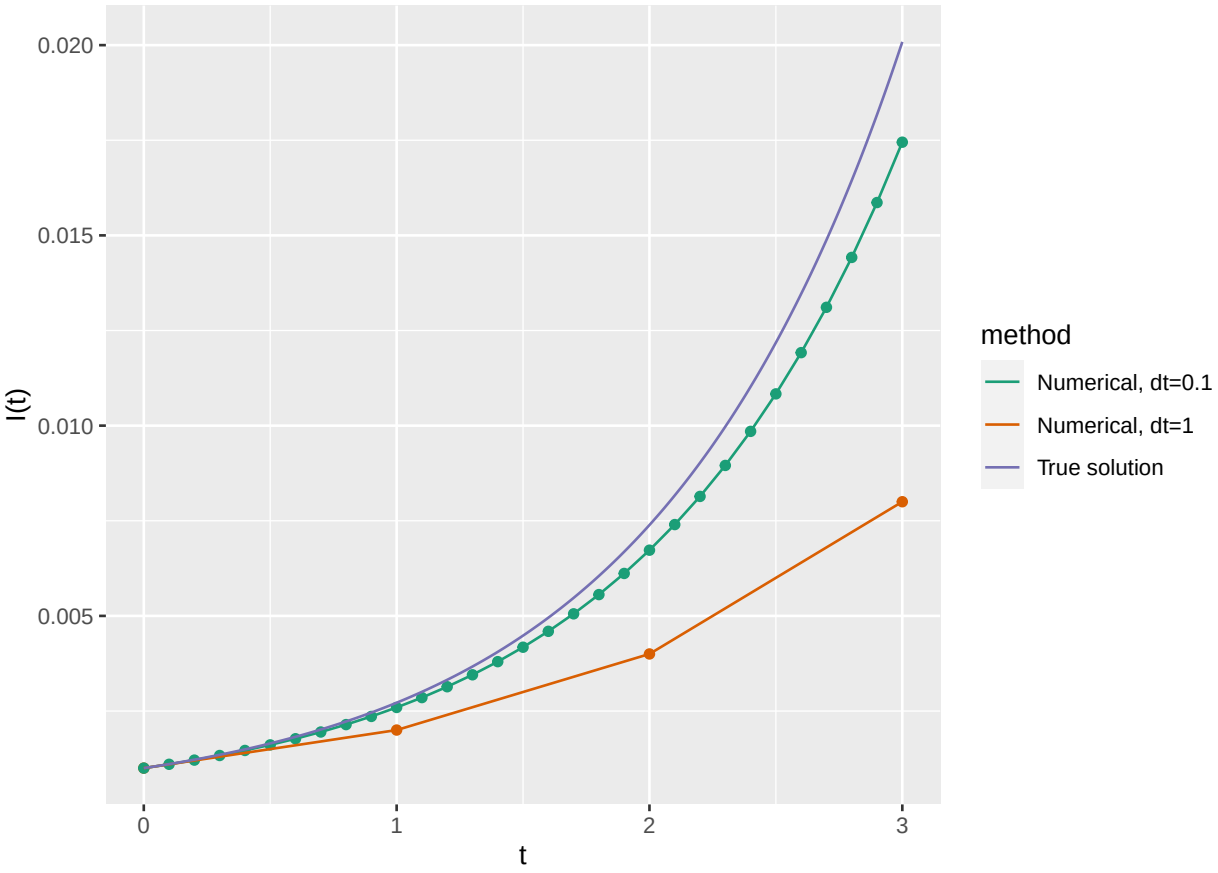
whose analytical solution if S were constant is:

$$I(t) = I_0 \exp((\beta - \gamma)St)$$

We set $I_0 = 0.001$ and $(\beta - \gamma)S = 1$. We solve this differential equation on the interval from $t = 0$ to $t = 3$ with two different choices of the time step ($\Delta t = 1$ and $\Delta t = 0.1$). We compare the obtained approximate solutions with the known true solution.

```
rhs <- function(t, state, params){
  list(state)
}

true_solution <- function(t){
  0.001 * exp(t)
}


sparse_times <- seq(0, 3, by=1)
dense_times <- seq(0, 3, by=0.1)


# Solve using "euler" from deSolve
# This implementation of the method uses the same times on which output is
# requested as the solver time points
numerical_solution_sparse = ode(
  y=0.001,
  times=sparse_times,
```

```
  parms=NULL,
  func=rhs,
  method="euler",
)

numerical_solution_dense = ode(
  y=0.001,
  times=dense_times,
  parms=NULL,
  func=rhs,
  method="euler",
)


# Save the results in data frames for plotting
results_sparse <- data.frame(
  t=numerical_solution_sparse[,1],
  f=numerical_solution_sparse[,2],
  method="Numerical, dt=1"
)

results_dense <- data.frame(
  t=numerical_solution_dense[,1],
  f=numerical_solution_dense[,2],
  method="Numerical, dt=0.1"
)

results_true <- data.frame(
  t=seq(0, 3, by=0.01),
  f=true_solution(seq(0, 3, by=0.01)),
  method="True solution"
)


ggplot() +
  geom_point(data=rbind(results_sparse, results_dense),
             aes(x=t, y=f, color=method),
             show.legend=F) +
  geom_line(data=rbind(results_sparse, results_dense, results_true),
            aes(x=t, y=f, color=method)) +
  scale_color_brewer(palette = "Dark2") +
  scale_fill_brewer(palette="Dark2") +
  ylab('I(t)')
```



We observe that the large step size of $\Delta t = 1$ causes significant inaccuracy in the solution. In particular, because the derivative is increasing over time, the forward Euler method (which, between solver time steps, assumes that the derivative remains constant at an old value) produces an approximate solution that is too low. For the smaller time step $\Delta t = 0.1$, the approximate solution is more accurate, although a negative deviation is still apparent.

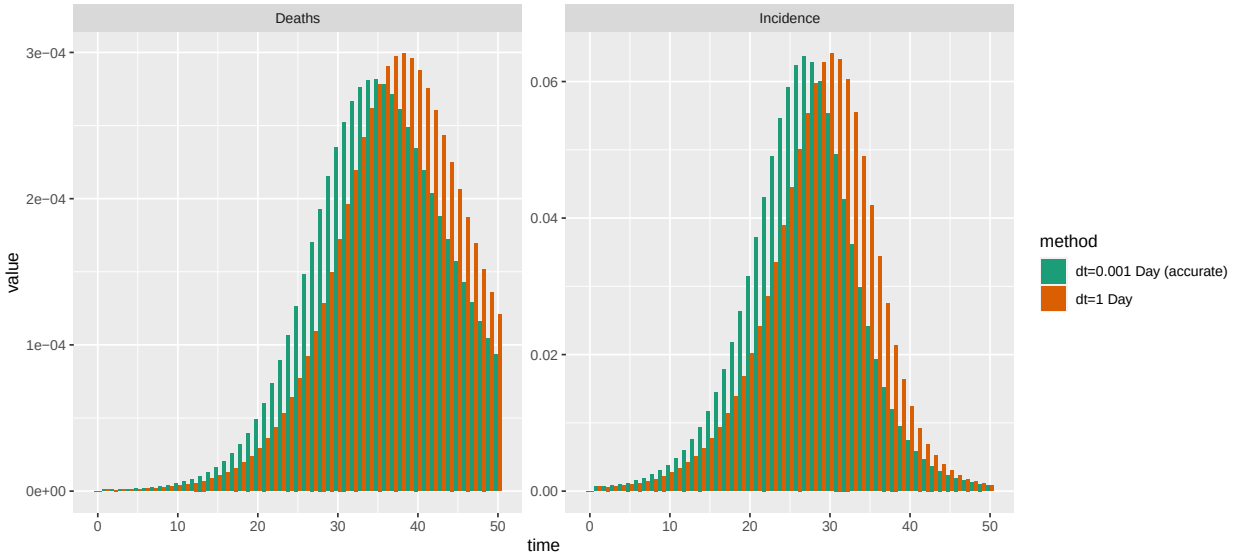
Effect of solver step size on the SEIRD model

First, we load the SEIRD model from comomodels, and specify plausible parameter values and initial conditions.

```
model <- comomodels::SEIRD()
transmission_parameters(model) <- list(beta=1.0, kappa=0.9, gamma=0.5, mu=0.1)

# Covid delta parameters
R0 <- 5
ifr <- 0.0066
kappa <- 1 / 3.7
gamma <- 1/7 * (1-ifr)
mu <- 1/7 * ifr
beta <- (mu + gamma) * R0

transmission_parameters(model) <- list(beta=beta, kappa=kappa, gamma=gamma, mu=mu)
initial_conditions(model) <- list(S0=0.999, E0=0, I0=0.001, R0=0)
```

Next, we run the model using the forward Euler method. We investigate two different choices for the solver step size. The first is a daily time step, which naively might be considered an acceptable default choice, given that we are interested in simulating daily cases and deaths. We also consider a much smaller time step of 0.001 days, for which we know that the solution is sufficiently accurate (see the section "Adaptive step size methods" below for further details on obtaining numerical solutions with a desired degree of accuracy).

```
times_daily <- seq(0, 50, by=1)
solution_daily <- run(model, times_daily, solve_method="euler")
changes_daily <- solution_daily$changes

times_dense <- seq(0, 50, by=0.001)
solution_dense <- run(model, times_dense, solve_method="euler")
df <- solution_dense$changes
df <- as.data.frame(df)

# Sum the deaths and incidences from the dense output to obtain daily values
changes_dense <- data.frame(time=NULL, value=NULL, compartment=NULL)
for (t in times_daily) {
  cases <- subset(df, t-1<df[,1] & df[,1]<=t)
  incidence <- subset(cases, cases[,3]=="Incidence")
  death <- subset(cases, cases[,3]=="Deaths")
  changes_dense <- rbind(changes_dense,
                         data.frame(time=t,
                                    value=sum(incidence$value),
                                    compartment="Incidence",
                                    age_range="0-150"))
  changes_dense <- rbind(changes_dense,
                         data.frame(time=t,
                                    value=sum(death$value),
                                    compartment="Deaths",
                                    age_range="0-150"))
}

# Reorder the deaths and incidences
changes_dense <- changes_dense[order(changes_dense$compartment, changes_dense$time),]

# Save the labels and combine for plotting
changes_dense$method <- "dt=0.001 Day (accurate)"
changes_daily$method <- "dt=1 Day"
all_changes <- rbind(changes_dense, changes_daily)

ggplot(subset(all_changes, all_changes[,3]=="Deaths" | all_changes[,3]=="Incidence"),
       aes(x = time, y = value, fill = method)) +
  geom_bar(stat="identity", position = position_dodge()) +
  facet_wrap(~ all_changes[,3], scales="free") +
  scale_color_brewer(palette = "Dark2") +
  scale_fill_brewer(palette="Dark2")
```



These results indicate that for the SEIRD model with these parameters, a daily step size is much too large to obtain accurate results. The daily step size causes the date of the peak of the epidemic (for both incident cases and deaths) to be overestimated by several days, due to the inaccuracy of the constant derivative assumption introduced by the Euler method.

Differential equations versus difference equations

The forward Euler recurrence relation defining the approximate solution to a *differential* equation,

$$f_{i+1} = f_i + \Delta t g(t_i, f_i),$$

is identical with a typical definition of a *difference* equation of the sort used in multiple fields of computational biology, including epidemiology (see (Izzo, Muroya, and Vecchio 2009)).

Thus, what we have up till now treated as numerical error arising from the forward Euler method may alternatively be viewed as the discrepancy between an (accurately solved) differential equation model and a corresponding difference equation model. The results above show, for the parameter values considered, that this discrepancy is not negligible. While difference equations can be valid and tenable models for certain phenomena, we see that a daily SEIRD difference equation cannot be considered equivalent to the original SEIRD differential equation model, that is, the same parameter values can cause substantially different outputs from each model. Difference equations tend to be a more appropriate modelling framework when considering non-overlapping generations, which is not generally the case in epidemiological modelling.

Adaptive step size methods

The chief advantage of numerical solvers such as forward Euler described above, which take a fixed grid of time points on which to calculate the approximate solution, is their ease of implementation. However, as seen above, these methods suffer because:

1. To obtain an accurate solution, the solver step size must be set to some small value, but it is often unknown how small this value must be.
2. When the solver step size is set to a small value, the method may be impractically slow.

Both of these defects are addressed by adaptive step size methods. In these, the user specifies not a step size but a *tolerance*—some relative or absolute value which the local error in the approximate solution should not

exceed. Then, the solver algorithm selects the appropriate step size in order for the solution to meet this tolerance, using the theoretical error properties of the solver and various techniques for step size adaptation. Adaptive solvers are able to vary the step size over the course of the solve, selecting very small values only in those regions of time where this is necessary, such as where the solution is changing rapidly over time, and otherwise increasing the step size to larger values. For this reason, they are much more efficient than fixed step solvers (Gautschi 2012).

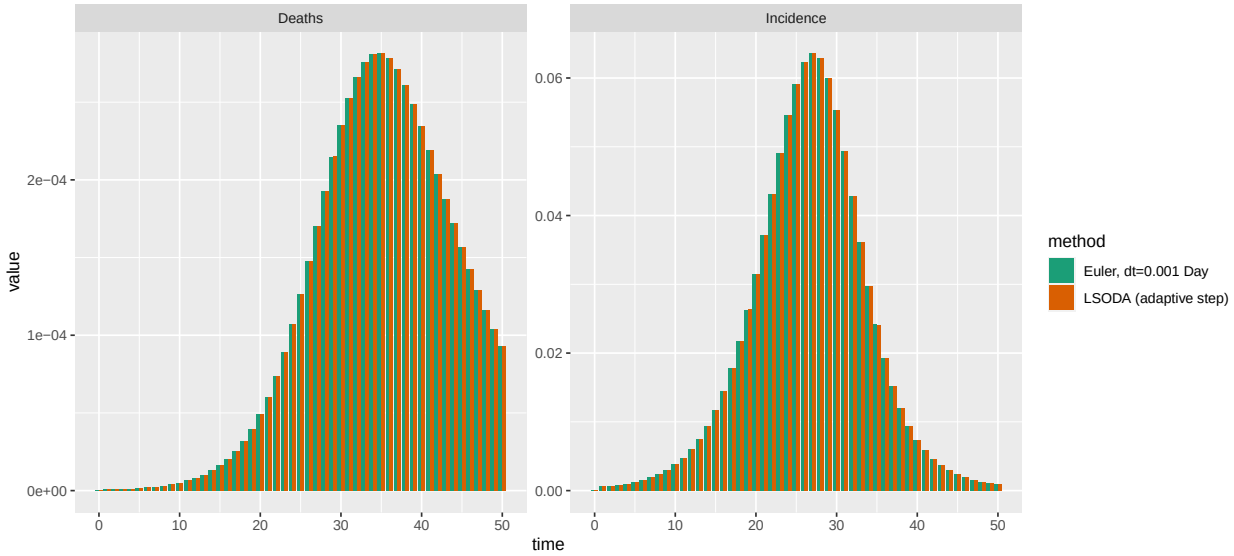
A wide variety of approaches to step size adaptation have been proposed. At each step, these methods typically use some estimator of the error introduced at that time step, and then select roughly the largest possible step such that the threshold imposed by the user-supplied tolerance is not exceeded.

Although adaptive step size solvers are challenging to implement, it is often possible to rely on existing implementations. In R, a wide selection of adaptive step size solvers are provided by the deSolve package (Soetaert, Petzoldt, and Setzer 2010). The comomodels library uses deSolve to handle numerical solution of the systems of ODEs that arise in compartmental epidemiology models, giving users easy access to efficient adaptive step size solvers. In comomodels, default values of the local error tolerances are used, so these need not be specified manually. The default solver used in comomodels is LSODA (Hindmarsh and Petzold 2005). This method automatically switches between the Adams and backward differentiation formula (BDF) solvers (Gautschi 2012) based on the properties of the differential equation being solved.

In the following example, we compare the performance of the forward Euler solver with a fixed, small step size to that of the LSODA adaptive step size solver.

```
times_dense <- seq(0, 50, by=0.001)
print("Timing for Euler (fixed step):")
#> [1] "Timing for Euler (fixed step):"
ptm <- proc.time()
solution_dense <- run(model, times_dense, solve_method='euler')
proc.time() - ptm
#>    user  system elapsed
#>   1.027   0.004   1.032

times_daily <- seq(0, 50, by=1)
print("Timing for LSODA (adaptive step):")
#> [1] "Timing for LSODA (adaptive step):"
ptm <- proc.time()
solution_adapt <- run(model, times_daily, solve_method='lsoda')
proc.time() - ptm
#>    user  system elapsed
#>   0.005   0.000   0.005

df <- solution_dense$changes
df <- as.data.frame(df)

# Sum the deaths and incidences from the dense output to obtain daily values
changes_dense <- data.frame(time=NULL, value=NULL, compartment=NULL)
for (t in times_daily) {
  cases = subset(df, t-1<df[,1] & df[,1]<=t)
  incidence = subset(cases, cases[,3]=="Incidence")
  death = subset(cases, cases[,3]=="Deaths")
  changes_dense = rbind(changes_dense,
                        data.frame(time=t,
                                   value=sum(incidence$value),
                                   compartment="Incidence",
                                   age_range="0-150"))
  changes_dense = rbind(changes_dense,
```

```
                        data.frame(time=t,
                                   value=sum(death$value),
                                   compartment="Deaths",
                                   age_range="0-150"))
}

# Save the labels and combine for plotting
changes_adapt <- solution_adapt$changes
changes_dense$method <- "Euler, dt=0.001 Day"
changes_adapt$method <- "LSODA (adaptive step)"
all_changes <- rbind(changes_dense, changes_adapt)

ggplot(subset(all_changes, all_changes[,3]=="Deaths" | all_changes[,3]=="Incidence"),
       aes(x = time, y = value, fill = method)) +
  geom_bar(stat="identity", position = position_dodge()) +
  facet_wrap(~ all_changes[,3], scales="free") +
  scale_color_brewer(palette = "Dark2") +
  scale_fill_brewer(palette="Dark2")
```



Both methods achieve the same accurate solution. However, for the LSODA solver, it is not necessary to manually select a time step: we merely provide the time points at which we want to access the solution (daily), and the solver handles the selection of the step sizes. We also note the significant speed advantage of the LSODA solver, which may be more than 50 times faster (the actual runtimes will be system dependent).

Adaptive solvers are not infallible. On highly challenging differential equations, and when supplied with insufficient tolerances for the simulation or inference task at hand, they may cause problems for simulation and inference. However, for the simulation of compartmental epidemiological models such as those covered by the comomodels package, adaptive solvers such as LSODA (the default choice of solver in comomodels) generally perform better in terms of solution fidelity and speed.

Discontinuities in the RHS (interventions)

Differential equations of the form

$$\frac{df}{dt} = g(t, f)$$

where g is discontinuous present further challenges to both adaptive and fixed step size methods, whose error properties (and, for adaptive step size methods, the algorithms used to determine the time steps) typically make assumption such as differentiability and Lipschitz continuity of g.

In the epidemiological models of the sort described by comomodels, discontinuities of g in time are possible, most notably in the SEIRDV model where step function interventions may be used to specify the rate of vaccination.

Some of the numerical methods that can be used to increase efficiency and decrease error in the case of discontinuous g are:

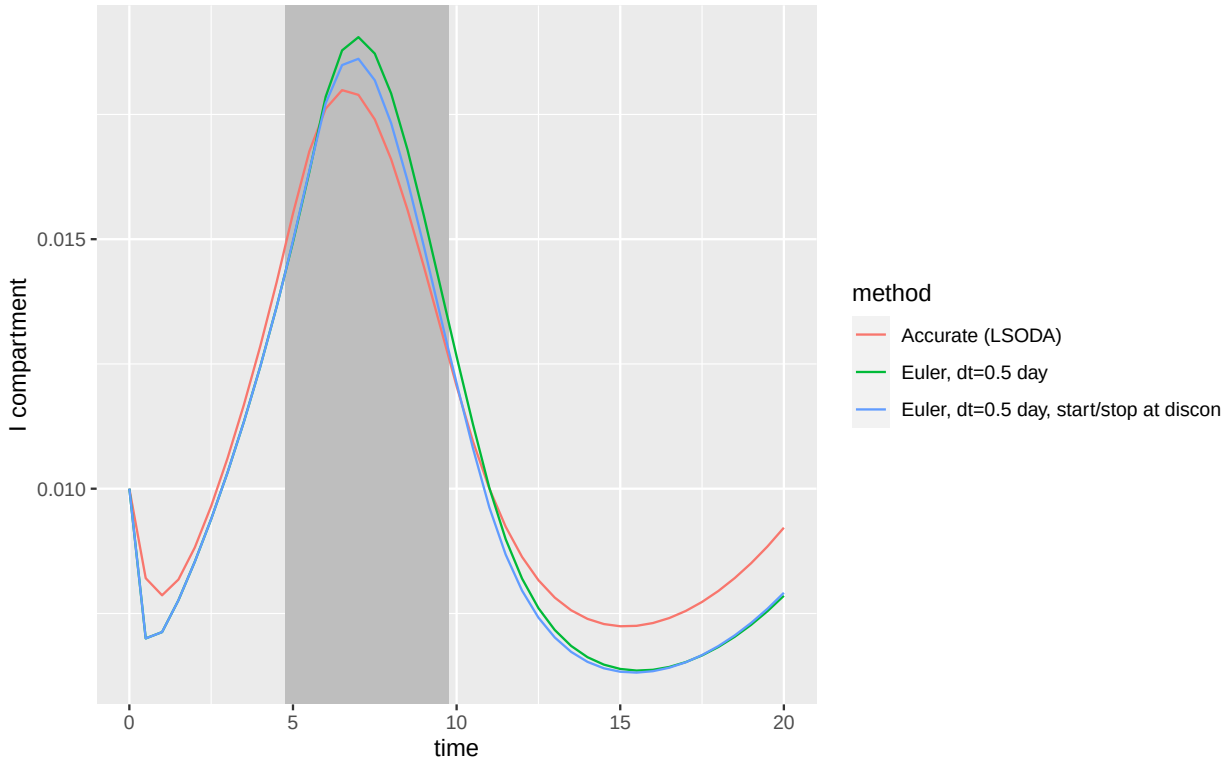
1. Divide the time interval into sets where g is continuous, and solve the ODE on each set separately.
2. Replace the discontinuous g with a smooth approximation, and use a standard solver.

The second approach is adopted by the SEIRDV model, as a tanh approximation to the step function is used.

First, we study the effects of a discontinuity in the vaccination rate in the SEIRDV model without a tanh approximation. The vaccination rate is set to 0.15 from $t = 4.75$ to $t = 9.75$, and 0 elsewhere. The model is solved using an Euler solver with a large time step of 0.5 days, once without any consideration of the discontinuities (green curve) and once with solver time steps inserted at the discontinuities (blue curve, method 1 above). These results are compared to an accurate (but potentially inefficient) solution obtained using the LSODA solver. The value of the I compartment is shown over time, and the time period of vaccination is shaded in gray.

```
model <- SEIRDV()
# Select transmission parameters where the effect of the discontinuity can be
# observed on a short time scale of 20 days
params <- list(beta=1, kappa=0.9, gamma=0.5, mu=0.1, nu=0.4, delta_V=0.15, delta_R=0.05)
transmission_parameters(model) <- params
initial_conditions(model) <- list(S0=0.99, E0=0, I0=0.01, R0=0, V0=0)
intervention_parameters(model) <- list(starts=c(4.75),
                                       stops=c(9.75),
                                       coverages=c(1),
                                       tanh_slopes=1000)

times <- seq(0, 20, by = 0.5)
df1 <- run(model, times, solve_method="lsoda")$states
df1$method = "Accurate (LSODA)"

solver_times <- seq(0, 20, by=0.5)
df2 <- run(model, solver_times, solve_method="euler")$states
df2$method = "Euler, dt=0.5 day"


solver_times <- sort(c(solver_times, 4.75, 9.75))
df3 <- run(model, solver_times, solve_method="euler")$states
df3$method = "Euler, dt=0.5 day, start/stop at discon"

df <- rbind(df1, df3, df2)

ggplot(subset(df, df[,3]=="I"),
       aes(x = time, y = value)) +
  geom_rect(aes(xmin=4.75,
                xmax=9.75,
                ymin = -Inf,
                ymax = Inf), fill = 'gray', alpha = 0.025) +
  geom_line(aes(color=method)) +
```

```
  ylab('I compartment')
```

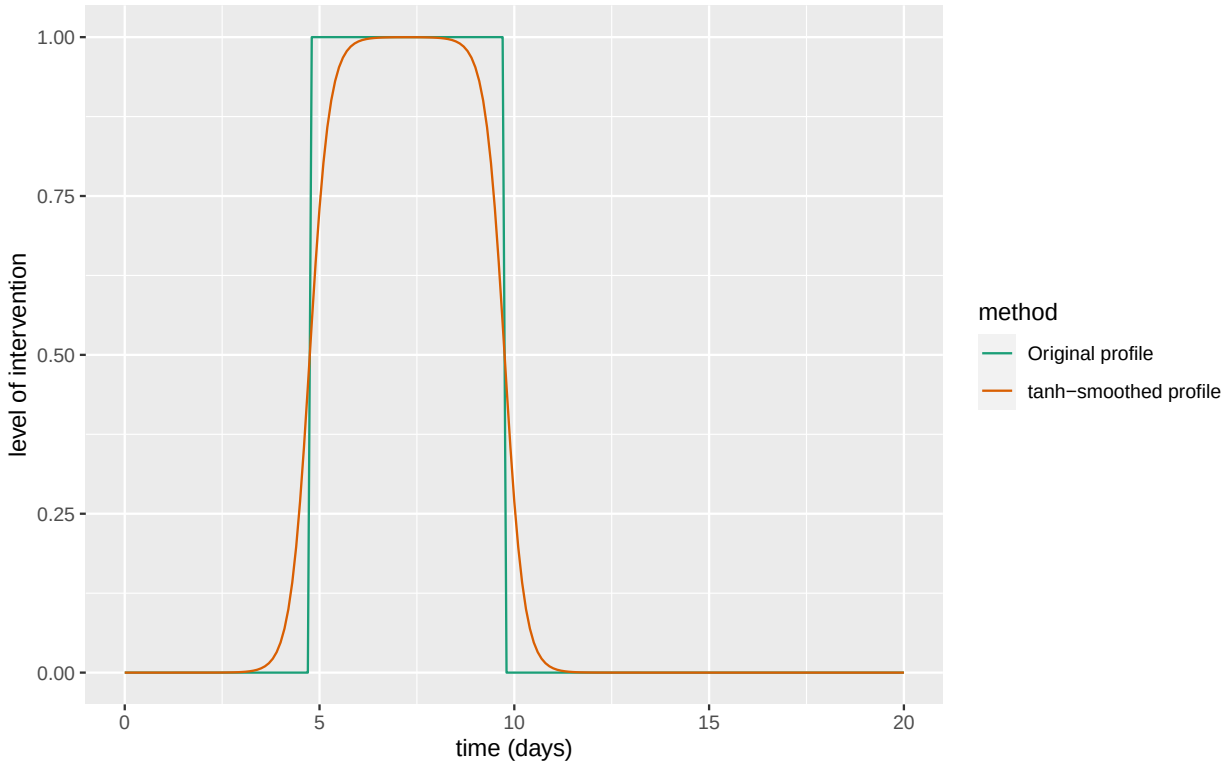


Both Euler solvers experience large error due to the large step size. However, additional error caused by the discontinuous δ_V is apparent, as, after the vaccination period begins, the Euler method without consideration of the discontinuity starts to deviate more drastically from the true solution.

An alternative approach is to smooth the discontinuity. In comomodels SEIRDV, this can be performed by supplying an appropriate slope to the tanh function that is used to smooth the δ_V profile:

```
int_parms <- InterventionParameters(start=c(4.75),
                                    stop=c(9.75),
                                    coverage=c(1),
                                    tanh_slope=1000)

sim_parms <- SimulationParameters(start=0, stop=20, tstep = 0.1)
i1 <- intervention_protocol(int_parms, sim_parms)
i1$method <- "Original profile"

int_parms2 <- InterventionParameters(start=c(4.75),
                                    stop=c(9.75),
                                    coverage=c(1),
                                    tanh_slope=2)
sim_parms2 <- SimulationParameters(start=0, stop=20, tstep = 0.1)
i2 <- intervention_protocol(int_parms2, sim_parms2)
i2$method <- "tanh-smoothed profile"

ggplot(rbind(i1, i2), aes(x=time, y=coverage, color=method)) +
  geom_line() +
  scale_color_brewer(palette = "Dark2") +
```

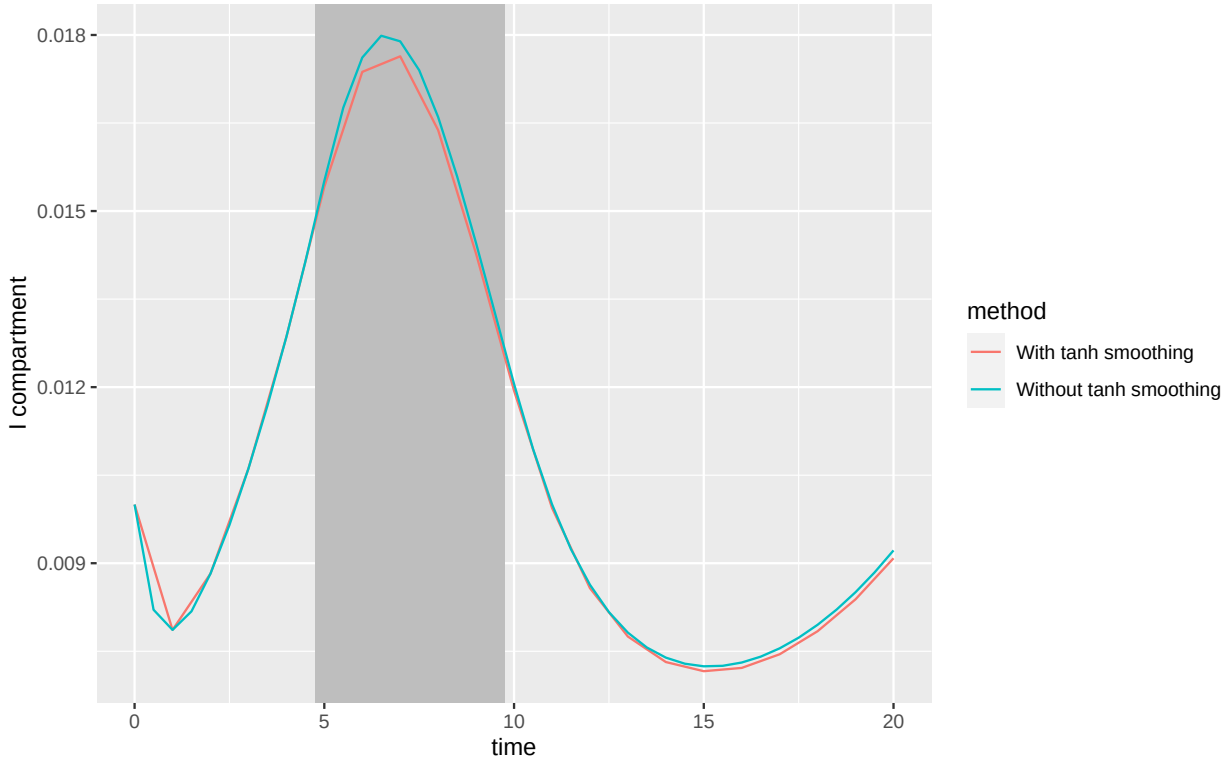
```
  labs(x = "time (days)", y = "level of intervention")
```



Next, we compare the effects of the tanh smoothing on the simulated incidences. The value of the I compartment is shown over time, and the time period of vaccination is shaded in gray.

```
model <- SEIRDV()
params <- list(beta=1, kappa=0.9, gamma=0.5, mu=0.1, nu=0.4, delta_V=0.15, delta_R=0.05)
transmission_parameters(model) <- params
initial_conditions(model) <- list(S0=0.99, E0=0, I0=0.01, R0=0, V0=0)
intervention_parameters(model) <- list(starts=c(4.75),
                                       stops=c(9.75),
                                       coverages=c(1),
                                       tanh_slopes=1000)

times <- seq(0, 20, by = 0.5)
df1 <- run(model, times, solve_method="lsoda")$states
df1$method = "Without tanh smoothing"


intervention_parameters(model) <- list(starts=c(4.75),
                                       stops=c(9.75),
                                       coverages=c(1),
                                       tanh_slopes=2)

times <- seq(0, 20, by = 1)
df2 <- run(model, times, solve_method="lsoda")$states
df2$method = "With tanh smoothing"

df <- rbind(df1, df2)
```

```
ggplot(subset(df, df[,3]=="I"),
       aes(x = time, y = value)) +
  geom_rect(aes(xmin=4.75,
                xmax=9.75,
                ymin = -Inf,
                ymax = Inf), fill = 'gray', alpha = 0.025) +
  geom_line(aes(color=method)) +
  ylab('I compartment')
```



Some slight differences are observed as a result of the tanh smoothing. Although the tanh smoothing does change the model behavior, in most applications of epidemiological modelling the smoothed version of the model is not only more tractable for numerical solvers but also more closely resembles realistic situations, where policies and interventions that change the parameters of the model take some time to be implemented.

Further details and a general treatment may be found in (Stewart 2011), Chapter 8.

References

Gautschi, W. 2012. *Numerical Analysis*. Springer Science & Business Media.

Hindmarsh, AC, and LR Petzold. 2005. "LSODA, Ordinary Differential Equation Solver for Stiff or Non-Stiff System." International Nuclear Information System, Nuclear Energy Agency of the OECD (NEA).

Izzo, G, Y Muroya, and A Vecchio. 2009. "A General Discrete Time Model of Population Dynamics in the Presence of an Infection." *Discrete Dynamics in Nature and Society* 2009.

Soetaert, K, T Petzoldt, and RW Setzer. 2010. "Solving Differential Equations in r: Package deSolve." *Journal of Statistical Software* 33: 1–25.

Stewart, DE. 2011. *Dynamics with Inequalities: Impacts and Hard Constraints*. SIAM.