On the numerical solution of compartmental models

Introduction

In comomodels, we focus on deterministic models which can be used to represent infection dynamics. Specifically, we investigate compartmental models, which are described by systems of coupled *ordinary differential equations* (ODEs). In these models, the rate of evolution of the state variables through time is modelled as a function of time, parameters and the current state values. If these differential equations are coupled with a set of initial conditions, they form a system whose solution represents the evolution of the state variables over time. Like many differential equation models of interest in scientific applications, typical compartmental models are nonlinear, and they generally cannot be solved exactly. Instead, an approximation of the solution must be obtained using numerical methods, which, in effect, numerically integrate the system by taking small discrete time steps.

Systems of ODEs can be integrated using a fixed time step solver, such as the *forward Euler* method. In preparing this manuscript, we came across a number of examples in the COVID-19 literature where such solvers were employed: Dehning et al. (2020) uses a fixed time step of 1 day with forward Euler to solve an SIR model, which, in turn, is employed as part of a Bayesian approach to infer change points in transmission of COVID-19 in Germany: these change points are then shown to correspond to the dates of publicly announced interventions; Birrell et al. (2021) describes a more complex compartmental model which is solved using a forward Euler scheme with a time step of 0.5 days. This model has been one of the key models used to inform UK governmental policy on COVID-19 (Anderson et al. 2020).

In this notebook, we show that when such methods are used without sufficiently small time steps, the numerical solution can represent a poor approximation. We recommend using more sophisticated adaptive step size solvers to integrate the types of (deterministic) compartmental models encountered in infectious disease epidemiology, as are implemented as defaults in the comomodels package.

An additional question encountered when numerically solving ODEs in applied epidemiology is how best to handle interventions? For example, a government may announce a nationwide lockdown which is immediately thought to change transmission. These types of interventions can cause discontinuities in the right-hand-side of ODEs, which, in turn can cause substantial error in the solutions if these abrupt changes are not accounted for in the numerical integration method. In the last part of the tutorial, we suggest ways to solve compartmental models in epidemiology that allow an accurate numerical solution to be obtained.

This notebook begins with a review of the use of forward Euler on a toy problem with a known solution. Then, it demonstrates the importance of accurate ODE solvers for comomodel's `SEIRD` model. We then show how numerical error can arise when vaccinations are included in the modelling framework – here, we use the `SEIRDV` model to illustrate this. Using this same model, we illustrate approaches to accurately solve this system.

We first load the packages necessary for this tutorial.

```
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 forward Euler method. While calculating the next function value f_{i+1}, this 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 a tiny 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 solution accuracy

We examine the numerical error 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 the equation for infectious individuals is given by:

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

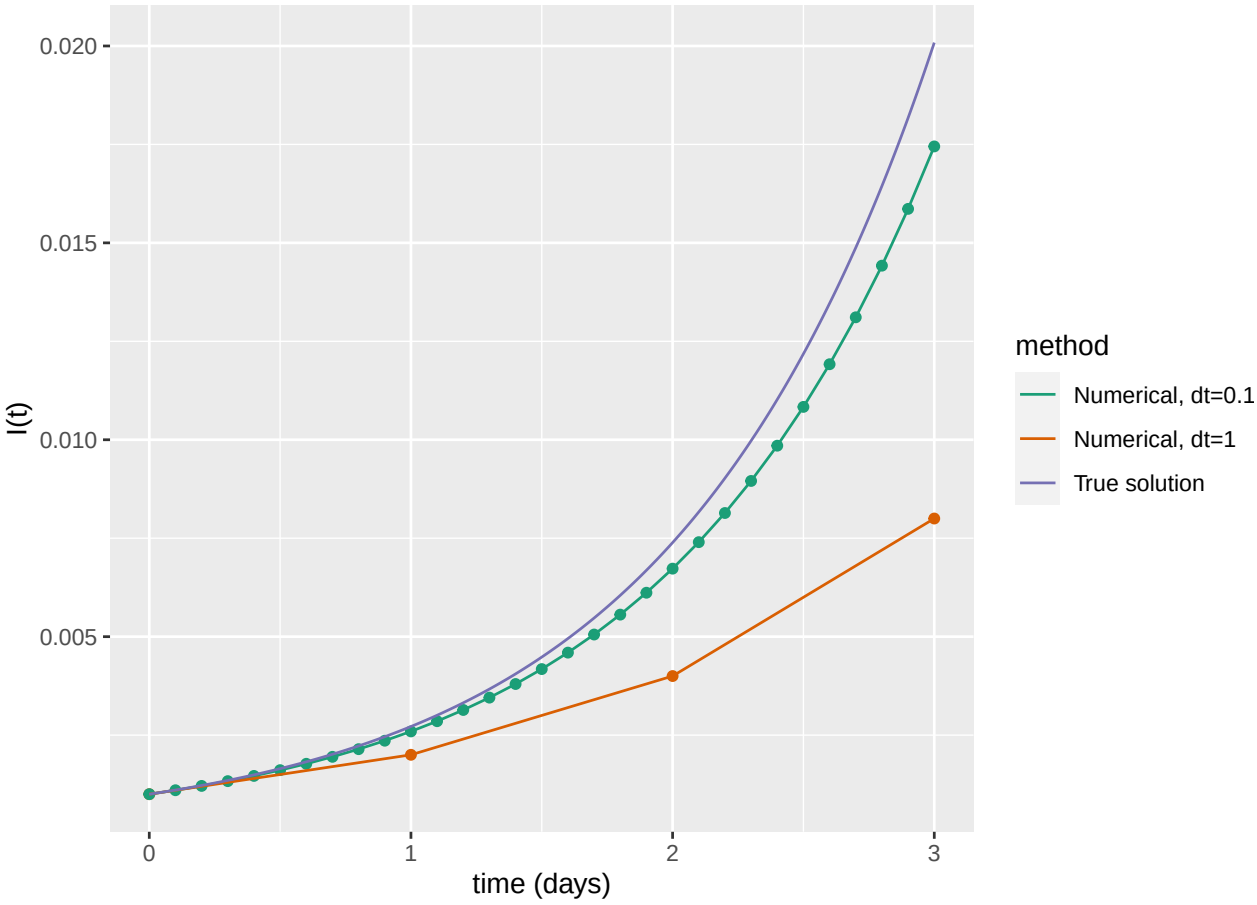
whose analytical solution is:

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

We set $I_0 = 0.001$ and $(\beta - \gamma)S = 1$. We solve this differential equation on the interval from $t = 0$ to $t = 3$ days with two different step sizes – $\Delta t = 1$ day and $\Delta t = 0.1$ day – and in both cases, we compare the obtained approximate solutions with the known true solution.

```r
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"
)

# Plot
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)') +
  xlab('time (days)')
```



This shows that a step size of $\Delta t = 1$ day results in a very inaccurate solution. Because the derivative is increasing over time, the forward Euler method (which, between solver time steps, assumes that the derivative remains constant) produces an approximate solution that is too low. For $\Delta t = 0.1$ day, the approximate solution is more accurate, although still significantly understates the true solution.

Effect of integrator step size on the SEIRD model

We now examine the impact of step size on the `SEIRD` model, where we set its parameter values to be representative of transmission of the Delta variant of SARS-CoV-2. First, we load the `SEIRD` model from comomodels and specify plausible parameter values and initial conditions.

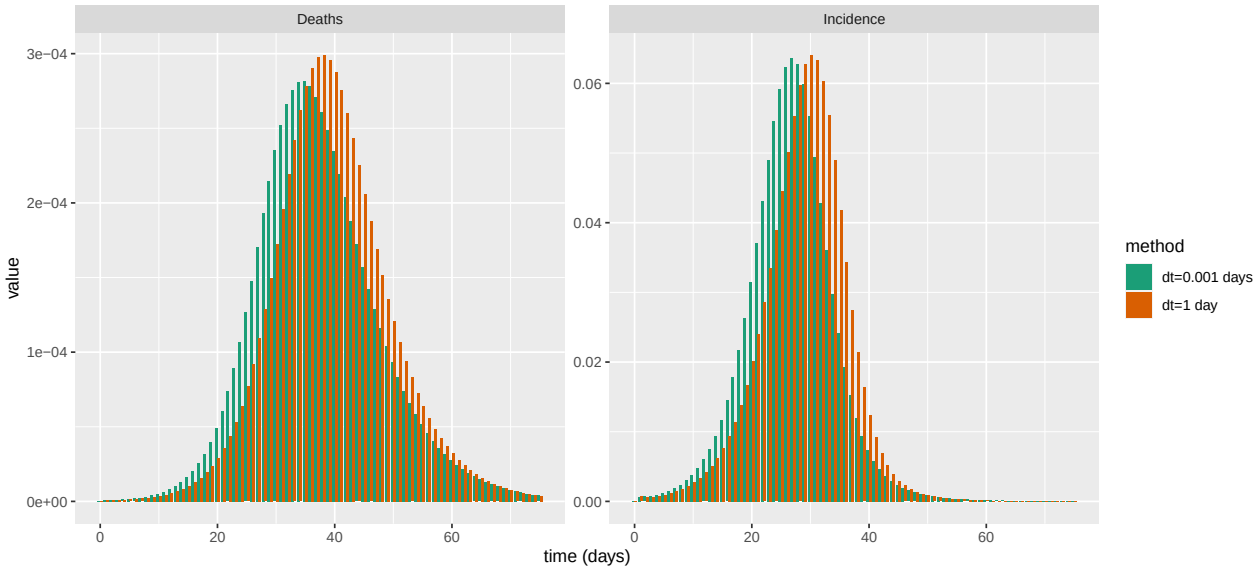
```
# Instantiate model
model <- comomodels::SEIRD()

# Delta parameters
params <- covid_transmission_parameters(variant="delta")
kappa <- params$kappa
```

```
mu <- params$mu
gamma <- params$gamma
R0_target <- params$R0
# Set beta to hit R0 target
beta <- (mu + gamma) * R0_target

transmission_parameters(model) <- list(beta=beta, kappa=kappa, gamma=gamma, mu=mu)

# Assume small proportion initially infected
initial_conditions(model) <- list(S0=0.999, E0=0, I0=0.001, R0=0)
```

Next, we solve 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 reasonable, 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 reasonably accurate (see the section "Adaptive step size methods" below for further details on obtaining numerical solutions with a desired degree of accuracy). We then plot the daily deaths and cases below.

```
# Day time step
times_daily <- seq(0, 75, by=1)
solution_daily <- run(model, times_daily, solve_method="euler")
changes_daily <- solution_daily$changes

# Shorter time step
times_dense <- seq(0, 75, 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$method <- "dt=0.001 days"
changes_daily$method <- "dt=1 day"
all_changes <- rbind(changes_dense, changes_daily)
```

```
# Plot
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") +
  xlab('time (days)')
```



These results indicate that for the `SEIRD` model with these parameters, a step size of 1 day results in substantial inaccuracy. This step size resulted in a higher peak case load which occurred later than the accurate solution. It also delayed the peak in deaths.

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 considered 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 that the difference in outputs from each approach can be nontrivial. While difference equations can be reasonable models for certain phenomena, our analysis shows that these models are not straightforward equivalents of differential equations. This is not surprising since differential equations suppose that individuals have a continuous overlap of generations, whereas difference equations suppose that these are non-overlapping (Murray 1989). As such, it is questionable whether difference equations should be used to model transmission dynamics of a disease such as COVID-19, where an infectious individual may "give birth" to a number of infected individuals who coexist with the infector.

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 should be to ensure a reasonable approximation.
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).

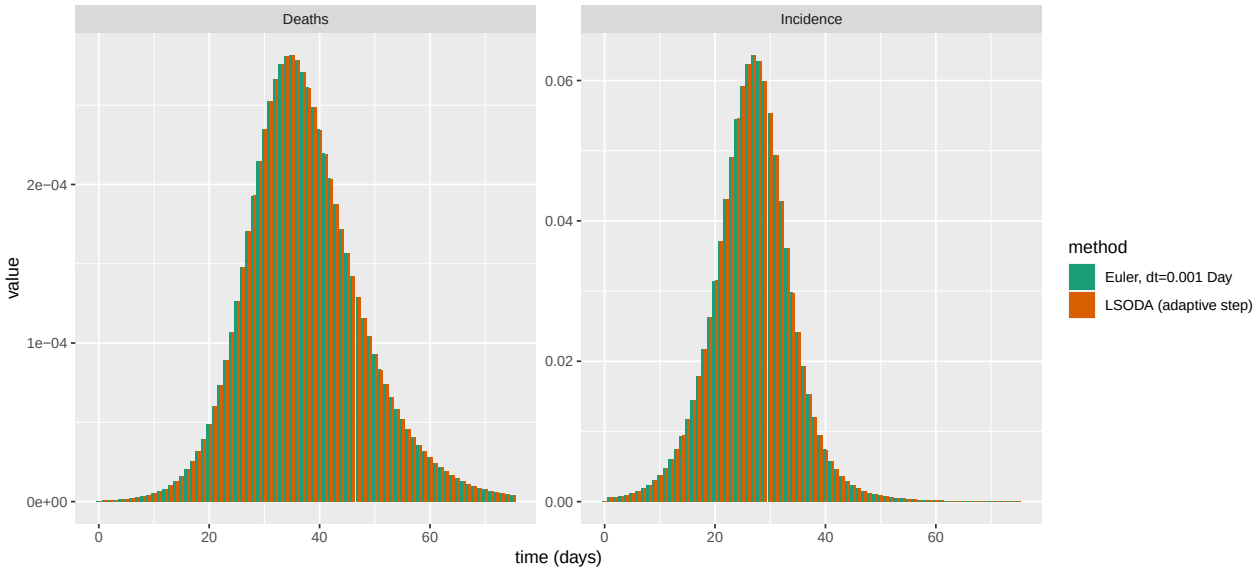
A wide variety of approaches to step size adaptation have been proposed. At each step, these methods typically use an estimator of the error introduced at that time step, and then – at least roughly – they select the largest possible step such that the threshold imposed by the user-supplied tolerance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

Although adaptive step size solvers are challenging to implement, there are many high quality, openly available software packages that implement them. 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. (Although we encourage users to investigate how changes to these error tolerances affect the solution to ensure that the solution is sufficiently accurate.) 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.

```
# Solve via forward Euler with small time step
times_dense <- seq(0, 75, by=0.001)
start <- Sys.time()
solution_dense <- run(model, times_dense, solve_method='euler')
time_euler <- Sys.time() - start

# Solve using LSODA
times_daily <- seq(0, 75, by=1)
start <- Sys.time()
solution_adapt <- run(model, times_daily, solve_method='lsoda')
time_lsoda <- Sys.time() - start

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)

# Plot
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") +
  xlab('time (days)')
```




```
# Times for each solve
print(paste0("Time taken by forward Euler = ",
             round(time_euler, 3), " seconds"))
#> [1] "Time taken by forward Euler = 1.806 seconds"
print(paste0("Time taken by LSODA = ",
             round(time_lsoda, 3), " seconds"))
#> [1] "Time taken by LSODA = 0.009 seconds"
```

Both methods achieve similarly accurate solutions. 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 (here, daily intervals), and the solver handles the selection of the step sizes. We also note the significant speed

advantage of the LSODA solver, which here is roughly 196 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 generally perform better in terms of solution fidelity and speed.

Discontinuities in the RHS and interventions

Differential equations of the form

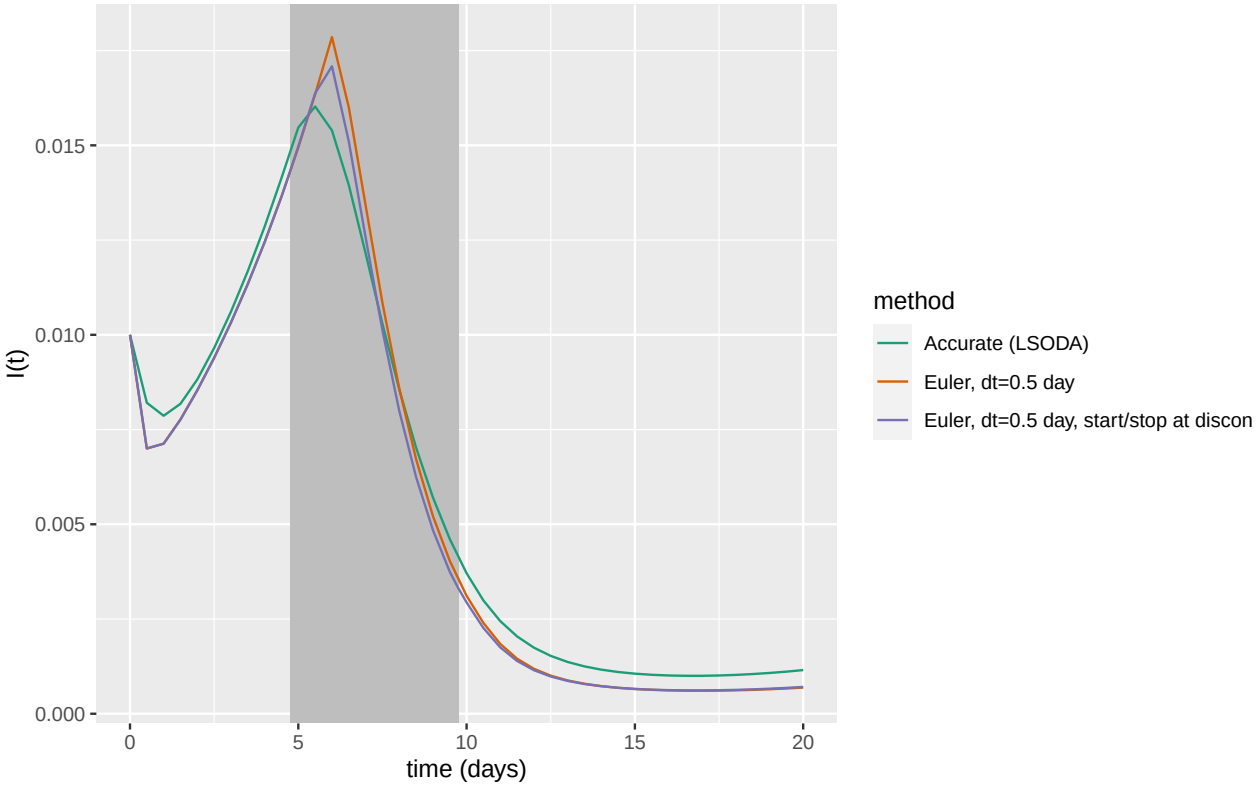
$$\frac{df}{dt} = g(t, f)$$

where g is discontinuous in time 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 models of COVID-19 transmission, discontinuities of g in time are possible, most notably in the `SEIRDV` model where mass vaccination campaigns can be represented as step functions. We now study the effects on solution accuracy of including a discontinuity in the vaccination rate in the `SEIRDV` model. The vaccination rate is set to 2 from $t = 4.75$ to $t = 9.75$ days, and 0 elsewhere. The model is solved using a forward 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=2, delta_V=0.15,
               delta_R=0.05, delta_VR = 0.02)
transmission_parameters(model) <- params
initial_conditions(model) <- list(S0=0.99, E0=0, I0=0.01, R0=0, V0=0, VR0=0)
intervention_parameters(model) <- list(starts=c(4.75),
                                       stops=c(9.75),
                                       coverages=c(1),
                                       tanh_slopes=1000)

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

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

# Solve using forward Euler using stop/start at
# discontinuities
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"
```

```r
# Combine
df <- rbind(df1, df3, df2)

# Plot
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(t)') +
  xlab('time (days)') +
  scale_color_brewer(palette = "Dark2")
```



Both forward Euler solvers experience large error due to the large step size. However, additional error caused by the discontinuous change in vaccination rate is apparent, as, after the vaccination period begins, the Euler method starts to deviate more drastically from the true solution.

Some of the numerical methods that can be used to increase efficiency and decrease error in this case 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.

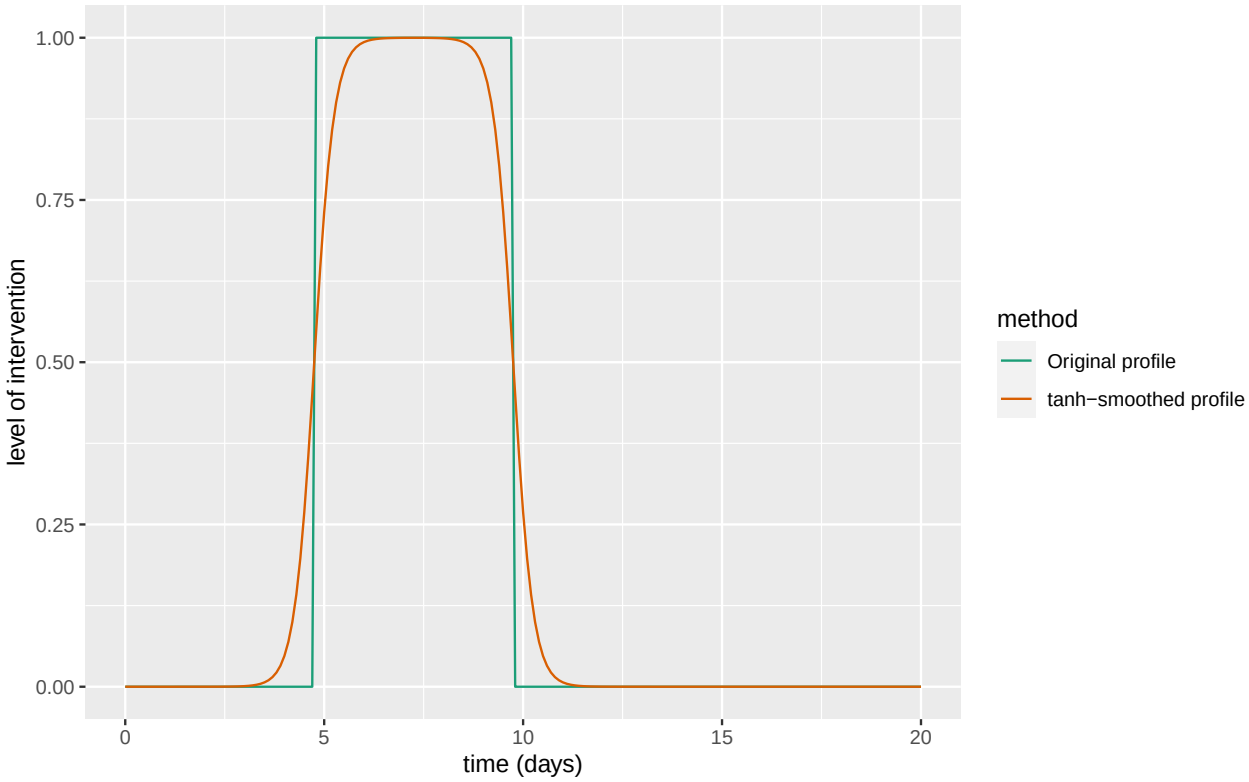
By default, to solve the `SEIRDV` model, we follow the second approach above, and a tanh approximation to the step function is used:

$$H(t) = 0.5\left[\tanh(a(t - t_{\text{start}})) - \tanh(a(t - t_{\text{stop}}))\right], \tag{1}$$

where $a > 0$ represents a "slope" parameter that dictates the curvature of the step approximation: if a is large, $h(t)$ is step-like; if a is small, the function is a more gently sloping curve. Here, t_{start} and t_{stop} represent the times at which an intervention is imposed and when it ends.

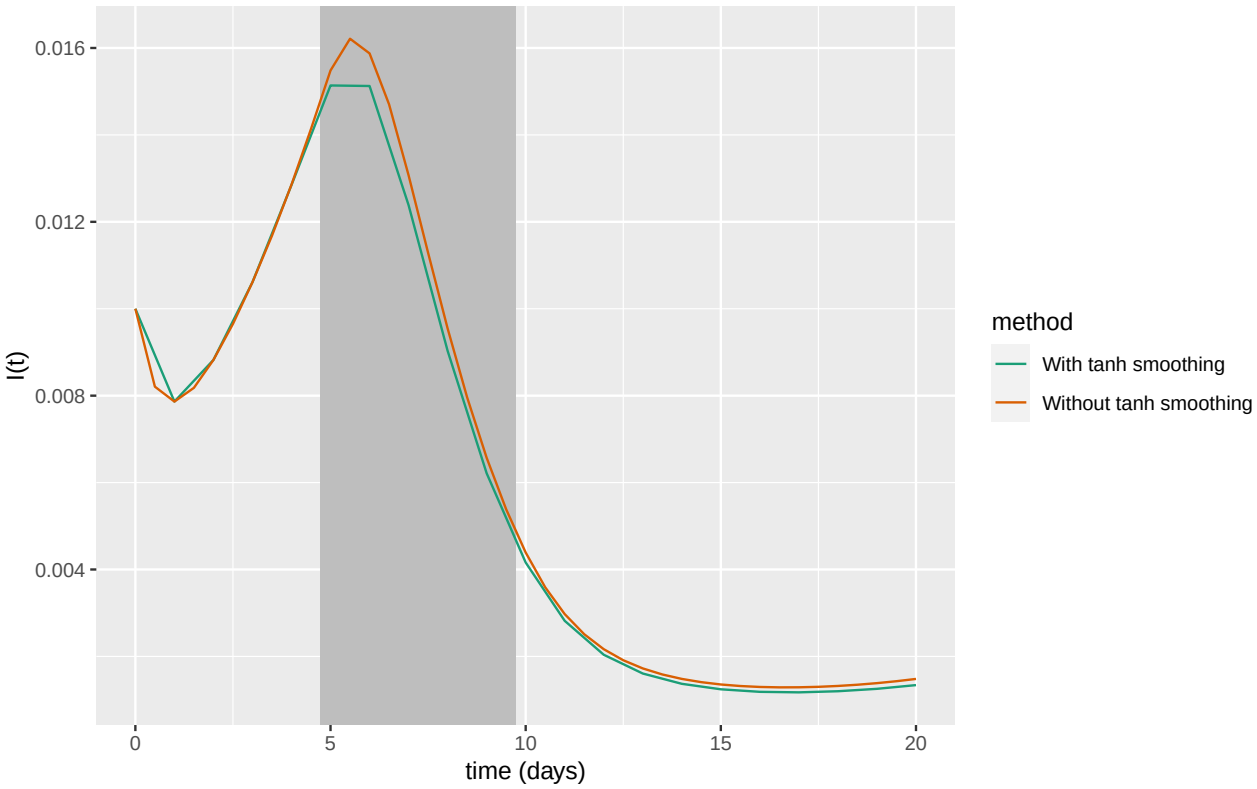
In the `SEIRDV` model, tanh-smoothing can be performed by supplying an appropriate slope (we use $a = 2$ below) to the tanh function that is used to smooth the vaccination profile. We compare a step-like vaccination protocol with a smoothed version below.

```
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"

# Plot
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.

```
# Set parameters of model
model <- SEIRDV()
params <- list(beta=1, kappa=0.9, gamma=0.5, mu=0.1,
               nu=1.5, delta_V=0.15, delta_R=0.05, delta_VR=0.01)
transmission_parameters(model) <- params
initial_conditions(model) <- list(S0=0.99, E0=0, I0=0.01, R0=0, V0=0, VR0=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"

# Combine
df <- rbind(df1, df2)

# Plot
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(t)') +
  xlab('time (days)') +
  scale_color_brewer(palette = "Dark2")
```



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 of discontinuous right-hand-sides of ODEs may be found in Chapter 8 of Stewart (2011).

Conclusions

In almost all cases, transmission dynamics models cannot be solved exactly. Instead, they are solved approximately, using numerical integration methods. The simplest of these methods is the forward Euler scheme, which integrates the system through a series of discrete intermediary steps through time, where the step size is predefined and constant. Using step sizes that have been used in high-profile published works on COVID-19 transmission, we showed that, for parameter ranges relevant for COVID-19 transmission, an `SEIRD` model solved with forward Euler deviates substantially from an accurate solution. These differences are large enough as to potentially influence policy decisions. To avoid such issues, we suggest using adaptive stepping methods, such as LSODA.

We also described how discontinuities in the right-hand-side of ODEs can lead to further numerical errors if naive methods are used. Here, we showed that representing a mass vaccination campaign as a step function produced inaccuracies. We then suggested remedies, either via changes to the numerical integration scheme or to the representation of the vaccination campaign itself.

Given these results, we now reflect on their relevance for the types of models applied to COVID-19 transmission.

These models often consist of large systems of coupled ODEs, and the cocktail of interventions considered may include many which may be represented as step functions. In such large systems, the extent of numerical error is likely to be greater than that which we found here using the `SEIRD` model. There is then a distinct chance that these errors may lead to differences in optimal choice of policy. As such, we strongly advocate that epidemiological modellers use the types of integration methods, such as adaptive step size methods,

which can more appropriately control for errors. Even if using these methods, it is still important to manually investigate whether changes to the algorithm settings manifests in changes to the outputs.

Another consequence of numerical errors is through their effects on parameter inference. Typical approaches to this include Bayesian inference, which, for systems of ODEs, is almost invariably performed through Markov chain Monte Carlo (MCMC) methods (Lambert 2018). These methods explore parameter space through a series of discrete steps, where, at each step – corresponding to a distinct set of parameter values – the system must be numerically solved. These solutions are then used to guide where to sample next and, in doing so, they affect the set of parameter draws used to represent parameter uncertainty. Since errors in the solution generally depend on the chosen parameter values, the numerical error will vary as a Markov chain steps through parameter space. These errors may then lead to arbitrary changes to the path of the chains, and, in doing so, lead to biased inferences. More work is required to quantify these effects and to determine their relevance in applied epidemiological modelling.

References

Anderson, R, C Donnelly, D Hollingsworth, M Keeling, C Vegvari, R Baggaley, and R Maddren. 2020. "Reproduction Number (R) and Growth Rate (r) of the COVID-19 Epidemic in the UK: Methods of Estimation, Data Sources, Causes of Heterogeneity, and Use as a Guide in Policy Formulation." *The Royal Society* 2020.

Birrell, P, J Blake, E Van Leeuwen, N Gent, and D De Angelis. 2021. "Real-Time Nowcasting and Forecasting of COVID-19 Dynamics in England: The First Wave." *Philosophical Transactions of the Royal Society B* 376 (1829): 20200279.

Dehning, J, J Zierenberg, FP Spitzner, M Wibral, JP Neto, M Wilczek, and V Priesemann. 2020. "Inferring Change Points in the Spread of COVID-19 Reveals the Effectiveness of Interventions." *Science* 369 (6500).

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.

Lambert, B. 2018. *A Student's Guide to Bayesian Statistics*. Sage.

Murray, James D. 1989. *Mathematical Biology, Vol. 19 of Biomathematics*. Springer, Berlin, Germany.

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.