Accurate approximation of solutions of infectious disease models with interventions

Wayne H. Enright (†), Christina C. Christara (†) and Paul H. Muir (‡)
(†) Department of Computer Science, University of Toronto, Toronto, ON M5S 2E4, Canada
{enright,ccc}@cs.toronto.edu
(‡) Department of Mathematics and Computing, Saint Mary's University, Halifax, NS B3H 3C3, Canada
paul.muir@smu.ca

Abstract

Since the beginning of the COVID-19 epidemic there has been an intensive global research effort devoted to the study of the virus and in particular to the development of mathematical models of COVID-19 that involve systems of ordinary differential equations (ODEs). In this paper, we consider systems of ODEs rising from an SEIR epidemic model *with interventions*. The impact of these interventions is that the solution to the model is nonsmooth at the points in time where the interventions are introduced or removed. This problem is sufficiently challenging that standard ODE solvers are not able to obtain numerical solutions of these models that have even moderate accuracy. However, we show in this paper that dramatic improvements in the accuracy and reliability of approximate solutions of this model can be obtained by employing carefully chosen, robust, numerical ODE methods. In particular, we consider an algorithm that can *automatically* detect, and efficiently and accurately handle the discontinuities that arise when the model includes interventions that are imposed in an attempt to restrict the spread of the virus and later removed when the spread of the virus is diminished.

Keywords: modelling infectious diseases; SEIR model; continuous Runge-Kutta methods; initial value problems; defect error control; discontinuity detection

Mathematics Subject Classification: 65L05, 65L06, 92C60

1 Introduction

While most current high quality ordinary differential equation (ODE) solvers return a continuous approximate solution, the error of the solution is typically controlled only at a discrete set of points across the domain. That is, the solvers typically do not provide error control for the *continuous* approximate solutions that they return. Over the past two decades, our research group has introduced a class of reliable methods based on control of the *defect* of continuous approximate solutions of systems of initial value ODEs, boundary value ODEs, and delay differential equations (DDEs) (see for example, [11, 12, 18], and references within). The defect of a continuous approximate solution is the amount by which that solution fails to satisfy the ODE. Control of the defect means that the computation is adapted so that an estimate of the maximum defect, over the problem domain, is less than a user-provided tolerance. In this paper, we consider how the expertise we have gained and the methods we have developed can be employed to efficiently compute an accurate and reliable approximate solution to a typical mathematical model of an epidemic *with interventions.*

In the remainder of this section, we provide a general discussion of the requirements for the effective approximate solution of mathematical models arising in scientific computation and then identify the challenges that arise when the model is associated with the simulation of the spread of an epidemic over time where interventions are introduced to control the spread of the disease and later removed when

the spread of the disease has diminished. This is followed, in Section 2, with a review of continuous Runge-Kutta (CRK) methods that can be used to determine a reliable approximate continuous solution to a system of initial value ODEs, and a discussion of how these methods can be used as the basis for a defect control strategy. In Section 3, we describe an approach that can be used with many standard ODE solvers to handle discontinuities, *when the user is able to provide information about the conditions under which each of the discontinuities will arise.* We also discuss in Section 3, the algorithm introduced in this paper that provides *automatic detection and handling of discontinuities*; no extra information is required from the user. In Section 4, we illustrate how this algorithm can be used to reliably, efficiently, and accurately compute approximate solutions to models of epidemics with interventions. We close in Section 5 with our summary and conclusions.

1.1 Approximating solutions of mathematical models

The determination of reliable approximate solutions of models arising in scientific computation typically involves four key stages:

- Formulation of the mathematical model of the system being investigated. (The model may involve parameters, some data-fitting, and systems of ODEs.)
- Approximation of the exact solution of this model relative to a specified accuracy parameter, TOL.
- Visualization of the approximate solution.
- Consideration of the well-posedness and the stability of the approximate solution. (This may involve sensitivity analysis).

Each of these stages should be considered when modelling epidemics and when selecting the ODE method used to approximate the solution of the systems of ODEs that arise. The relevant issues that must be addressed include:

- An approximate solution must be delivered in a way that is easy to understand, display, and manipulate by a casual user.
- A discrete approximate solution (available at only a finite number of points in time) is not sufficient (as it is difficult to visualize and its overall accuracy is difficult to interpret). Therefore, *the numerical method must deliver a continuous approximate solution for any point, t, in the domain of interest.*
- *Some measure of the accuracy of the continuous approximate solution must be controlled by the ODE method and must be easy to quantify and interpret by a user.*
- The ODE method must be adaptive and easy to invoke – one need only specify the error tolerance and the information necessary to define the model. The method will then employ an adaptive strategy in order to obtain a continuous approximate solution for which some measure of its accuracy satisfies the prescribed tolerance.

1.2 A SEIR epidemic model with interventions

Consider modelling the spread of COVID-19 with a typical SEIR epidemic model [14] that involves the following system of initial value ODEs,

$$\frac{dS}{dt} = \mu\tilde{N} - \mu S - \frac{\beta}{\tilde{N}}IS, \tag{1.1}$$

$$\frac{dE}{dt} = \frac{\beta}{\tilde{N}}IS - aE - \mu E, \tag{1.2}$$

$$\frac{dI}{dt} = aE - \gamma I - \mu I, \tag{1.3}$$

$$\frac{dR}{dt} = \gamma I - \mu R, \tag{1.4}$$

where

- $S(t)$, $E(t)$, $I(t)$ and $R(t)$ denote the populations of Susceptible, Exposed, Infectious and Recovered individuals at time t, where $t \in [t_0, t_{end}]$,
- μ, β and γ are, respectively, the replenishment, transmission, and recovery rates,
- a^{-1} is the incubation period, $\tilde{N}$ is the total population, and,
- the initial time t_0 and the initial state [$S(t_0)$, $E(t_0)$, $I(t_0)$, $R(t_0)$] are assumed to be given, with the model predicting future states corresponding to $t > t_0$.

Christara [6] introduced a simulation of COVID-19 in Canada as an interesting application of software for the numerical solution of ODEs. The simulation is based on the SEIR model, (1.1)-(1.4), with $t_0 = 0$, $t_{end} = 95$, and,

- the initial conditions representing the populations of susceptible, exposed, infectious, and recovered individuals in Canada (on a day in early 2020), chosen to be [37740896, 103, 1, 0],
- the fixed model parameters are $a = 0.125, \gamma = 0.06, \mu = 0.01/365$, and,
- the initial value of the transmission rate is $\beta = 0.9$. (The β parameter is not assumed to be fixed.)

The transmission rate β is piecewise constant, initially equal to 0.9, as mentioned above. This value of β leads to local rapid growth in the exposed and infected populations. When $E(t)$ reaches a value of 25,000, β is set to 0.02 (reflecting an attempt to control the spread of the epidemic by limiting interactions, through public health measures, between infectious and susceptible individuals). This smaller value of β leads to local rapid decay in the exposed and infected populations. At some later time, $E(t)$ reaches a value of 10,000, at which point β is reset to 0.9 (reflecting a loosening of the restrictions that were put in place previously to limit the spread of the virus). The evolution of the disease continues with these alternating interventions invoked or removed according to the value of $E(t)$. Each change in the value of β introduces a discontinuity in the ODE. *It is clear from the above description that $E(t)$ should oscillate between 25,000 and 10,000.*

As mentioned above, *whenever the β value is changed, this introduces a discontinuity in the right-hand side of the system of ODEs. In order for a numerical method to deliver an accurate solution, the*

method must locate and accurately cross each of these discontinuity points. This is the main challenge that must addressed when reliably, efficiently, and accurately approximating the solution of this model.

Disease models with interventions similar to the model described above have been considered since at least 2010 - see, e.g., [2,3,5,15,17], and references within. Some papers provide only theoretical analyses with no numerical results. Most papers that provide numerical results are silent regarding what numerical methods are used to obtain the results. In the papers that do mention a numerical method, only very limited detail is provided. From the above list of papers, in [15] the author indicates that the classical 4-stage, 4th order Runge-Kutta method is used with a fixed time step to obtain the numerical solutions presented in the paper, while in [5], the authors say that they use Euler's method for the numerical computations.

We now return to the model described above and consider a straightforward approach for the computation of a numerical approximation to the solution of the model described above that might be employed by a typical user. In such an approach, the function that is called by the ODE solver to evaluate the right-hand side of the ODE system contains 'if' statements that change the value of β in order to simulate the imposition and relaxation of public health measures based on the value of $E(t)$. We implemented this approach using a standard ODE solver, ode45, from the Matlab problem-solving environment, with its default tolerances (absolute tolerance, 10^{-6}, relative tolerance, 10^{-3}). In Figure 1, we plot the computed approximation to $E(t)$ versus t that we obtained.

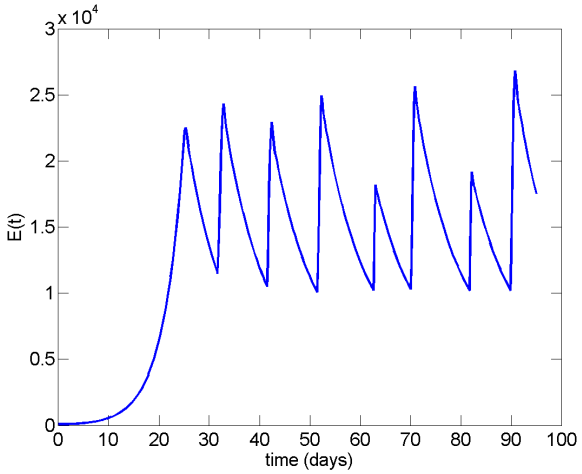

Figure 1: Plot of $E(t)$, number of exposed individuals, vs t, time in days, for the SEIR model with a piecewise constant transmission coefficient, β. Inaccurate solution obtained using ode45, from the Matlab problem-solving environment, with default tolerances.

We next present an accurate numerical solution to the model, obtained using a reliable ODE method (see Section 3), with a relative tolerance of 10^{-6}. In Figure 2, we plot this computed approximation to $E(t)$ versus t.

Examining this plot, we see that $E(t)$, as expected, increases and decreases in a regular fashion between the upper and lower bounds of 25,000 and 10,000, respectively. There are 13 discontinuities in the time interval $[0, 95]$.

Returning to Figure 1, we see that the computed solution shown there does not agree with the more accurate solution shown in Figure 2. In particular, *we see that the maximum and minimum $E(t)$ values do not agree with the correct values of 25,000 and 10,000*. As well, the times at which the interventions

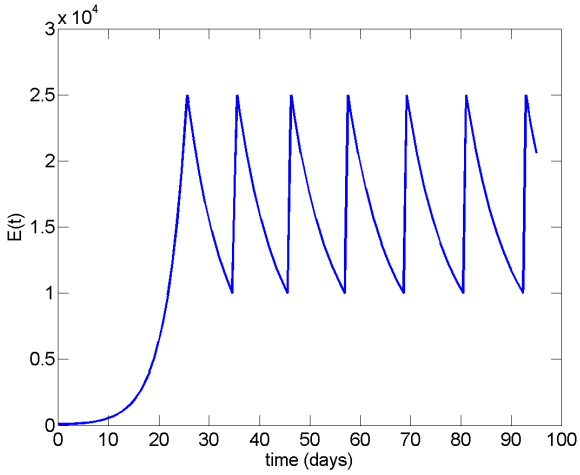

Figure 2: Plot of $E(t)$, number of exposed individuals, vs t, time in days, for the SEIR model with a piecewise constant transmission coefficient, β. Accurate solution obtained using a reliable algorithm (see Section 3).

arise do not agree with the times of the interventions for the accurate solution, and even the number of interventions is different. (This is not an artifact of the plotting of the computed solution; ode45 is not able to accurately locate and step through the discontinuities.) Furthermore, this type of behavior is quite common among ODE solvers when they are applied to this model without the use of a special treatment to locate and accurately handle each of the discontinuities [1].

The point of Figure 1 is to show that a standard ODE solver, used with its default tolerance, when applied to the model we consider in this paper, will not compute a solution that is at all reasonable. The point of Figure 2 is to show what an accurate solution to the model looks like. It is not appropriate to compare the two computations since ode45 returns a solution that is incorrect.

2 CRK methods and defect control

In this section, we briefly review CRK methods and their use within a defect control algorithm.

2.1 CRK methods

Consider a general initial value problem consisting of a system of n ODEs with corresponding initial conditions of the form

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0, \quad \text{for } t \in [t_0, t_{end}], \quad y(t) \in \mathbb{R}^n, \quad y_0 \in \mathbb{R}^n.$$

A standard numerical ODE method will introduce a partitioning of the domain, $[t_0, t_{end}]$, $t_0 < t_1 < \ldots < t_N = t_{end}$, and will compute corresponding discrete approximations, $y_1, \ldots, y_N$, where $y_i \approx y(t_i)$. The y_i's are usually determined from the application of a numerical formula on a sequence of steps of length $h_i = t_i - t_{i-1}$, $i = 1, \ldots, N$, starting from t_0.

On the ith step taken by the ODE method, let $z_i(t)$ be the *exact* solution of the *local* initial value ODE system

$$z_i' = f(t, z_i(t)), \quad z_i(t_{i-1}) = y_{i-1}, \quad \text{for } t \in [t_{i-1}, t_i]. \tag{2.1}$$

A standard s-stage, discrete Runge-Kutta (RK) formula determines y_i in terms of y_{i-1} using

$$y_i = y_{i-1} + h_i \sum_{j=1}^{s} b_j k_j,$$

where the j^{th} stage, k_j, is defined by

$$k_j = f\left(t_{i-1} + h_i c_j,\ y_{i-1} + h_i \sum_{r=1}^{s} a_{jr} k_r\right), j = 1, \ldots, s.$$

Here, b_j, c_j, and a_{jr} are the coefficients that define the discrete RK formula. In order to use the above formula, it is assumed that the user provides to the ODE solver a function, $f(t, y)$, that gives an evaluation of the right-hand side of the ODE system.

A discrete RK formula is of p^{th}-order if it satisfies the condition

$$y_i = z_i(t_i) + O(h_i^{p+1}).$$

A continuous extension of a p^{th}-order, discrete RK formula, called a continuous RK (CRK) method, is determined by introducing $(\tilde{s}-s)$ additional stages to obtain an order p continuous solution approximation for any $t \in (t_{i-1}, t_i)$. This continuous solution approximation, $u_i(t)$, defined on (t_{i-1}, t_i), has the form

$$u_i(t) = u_i(t_{i-1} + \tau h_i) = y_{i-1} + h_i \sum_{j=1}^{\tilde{s}} b_j(\tau)\, k_j, \tag{2.2}$$

where $b_j(\tau)$ is a polynomial of degree at least p and $\tau = \frac{t-t_{i-1}}{h_i}$. Since we have assumed that the CRK method is constructed to be of order p, $u_i(t)$ satisfies the condition that

$$u_i(t) = z_i(t) + O(h_i^{p+1}), \quad t \in [t_{i-1}, t_i]. \tag{2.3}$$

The $[u_i(t)]_{i=1}^{N}$ polynomials together (for each of the n components of the ODE system) define a vector of piecewise polynomials, $U(t)$, for $t \in [t_0, t_{end}]$. We require $f(t_{i-1}, y_{i-1})$ and $f(t_i, y_i)$ to be included in the set of stages, $\{k_j\}_{j=1}^{\tilde{s}}$, that appear in (2.2). From the discrete RK method, we have $k_1 = f(t_{i-1}, y_{i-1})$ and we set $k_{s+1} = f(t_i, y_i)$. The interpolant on $[t_{i-1}, t_i]$ is represented in Hermite-Birkhoff form and is required to interpolate y_{i-1} and y_i at t_{i-1} and t_i. The derivative of the interpolant is required to interpolate $f(t_{i-1}, y_{i-1})$ and $f(t_i, y_i)$ at t_{i-1} and t_i. Furthermore, in order to obtain an interpolant of order p, the derivative of the interpolant is required to interpolate several additional stages from the set, $\{k_j\}_{j=1}^{\tilde{s}}$. The Hermite-Birkhoff interpolant, $\tilde{u}_i(t)$, on $[t_{i-1}, t_i]$, is constructed to satisfy the conditions $\tilde{u}_i(t_{i-1}) = y_{i-1}$, $\tilde{u}_i(t_i) = y_i$, $\tilde{u}_i'(t_{i-1}) = f(t_{i-1}, y_{i-1})$, $\tilde{u}_i'(t_i) = f(t_i, y_i)$, and $\tilde{u}_i'(t_{i-1} + c_r h_i) = k_r$, where $r \in \{1, \ldots, \tilde{s}\}$ and c_r is a coefficient associated with k_r. The specific conditions that must be imposed on the Hermite-Birkhoff basis polynomials in order for these conditions to be satisfied are provided in [12]. Finally, $\tilde{u}_i(t)$ is transformed from Hermite-Birkhoff form into the standard CRK form (2.2). See [9] for further details.

Because $u_i(t)$ interpolates y_{i-1} and y_i and its derivative interpolates $f(t_{i-1}, y_{i-1})$ and $f(t_i, y_i)$, it follows that $U(t) \in C^1[t_0, t_{end}]$. Then, $U(t)$ is the continuous approximate solution to the ODE model.

2.2 Defect control for CRKs

As mentioned earlier, an effective method for computing an approximate solution to a system of ODEs must deliver an *error-controlled, continuous* solution approximation. One such type of error control is known as *defect control.* If we use a CRK method satisfying (6) and (7), then the approximate solution, $U(t)$, has an associated defect (or residual) of the form

$$\delta(t) \equiv f(t, U(t)) - U'(t) \equiv f(t, u_i(t)) - u_i'(t), \quad \text{for } \ t \in [t_{i-1}, t_i], \tag{2.4}$$

which is a vector with n components.

For an extension of (2.2), it can be shown that, for sufficiently differentiable $f(t, y(t))$, on the ith step, $[t_{i-1}, t_i]$, we have

$$\delta(t) = G(\tau)h_i^p + O(h_i^{p+1}), \tag{2.5}$$

where $\tau = \frac{t-t_{i-1}}{h_i}$, and $G(\tau)$ depends on the problem and the CRK formula but not on h_i.

To see this, consider an extension of (2.2) that is based on applying a "boot-strapping" algorithm starting with (2.2) as the base interpolant. We first construct several new stages based on evaluations of (2.2) and then we construct a new Hermite-Birkhoff interpolant, $\tilde{u}_i(t)$, that interpolates the solution approximations, y_{i-1} and y_i, and whose derivative interpolates $f(t_{i-1}, y_{i-1})$, $f(t_i, y_i)$ and the new stages. The error of this new interpolant is dominated by the error in the term that depends on y_i, which is $O(h_i^{p+1})$. That is, we have

$$\tilde{u}_i(t) = z_i(t) + O(h_i^{p+1}), \quad t \in [t_{i-1}, t_i]. \tag{2.6}$$

Differentiating this expression, we then also have

$$\tilde{u}_i'(t) = z_i'(t) + O(h_i^p), \quad t \in [t_{i-1}, t_i], \tag{2.7}$$

where this error is also associated with the term in the derivative of the interpolant that depends on y_i.

Letting the defect of $\tilde{u}_i(t)$ on the ith step be $\delta_i(t)$, we have

$$\delta_i(t) = f(t, \tilde{u}_i(t)) - \tilde{u}_i'(t) = [f(t, \tilde{u}_i(t)) - \tilde{u}_i'(t)] - [f(t, z_i(t)) - z_i'(t)], \tag{2.8}$$

where we recall that $z_i(t)$ is the exact local solution satisfying $z_i'(t) = f(t, z_i(t))$. Rearranging terms in the above gives

$$\delta_i(t) = [z_i'(t) - \tilde{u}_i'(t)] + [f(t, z_i(t)) - f(t, \tilde{u}_i(t))]. \tag{2.9}$$

Assuming a Lipschitz condition on $f(t, y(t))$ and recalling that the error of $\tilde{u}_i(t))$ is $O(h_i^{p+1})$ and that the error of $\tilde{u}_i'(t))$ is $O(h_i^p)$, we then have

$$\delta_i(t) = G(\tau)h_i^p + O(h_i^{p+1}), \tag{2.10}$$

where $G(\tau)$ depends on the term of $\tilde{u}_i'(t)$ that depends on y_i. This means that the defect, to leading order, is a multiple of $G(\tau)$, which is the derivative of the Hermite-Birkhoff basis polynomial for the term that depends on y_i. Since this polynomial is known, we can determine the maximum of its derivative on $\tau \in [0, 1]$ and therefore we can determine where, asymptotically, the maximum defect will occur on each step. See [12] for further details. (As mentioned above, $\tilde{u}_i(t)$ is based on $u_i(t)$, which is in turn based on the underlying discrete RK method. For a pth order RK method, $f(t, y(t))$ is required to have continuous

derivatives up to order p. If $f(t, y(t))$ is nonsmooth, this will severely reduce the order of the method. See, e.g., [4] for further details.)

The above approach gives what is known as an *asymptotically correct* estimate of the maximum defect on each step. A defect control time-stepping algorithm will adaptively determine h_i in an attempt to ensure that maximum for $t \in [t_{i-1}, t_i]$ of $||\delta(t)||$ is bounded by TOL. The accuracy of an approximate solution can then be described in terms of the maximum (over all steps) of $||\delta(t)||$. We have derived and implemented a class of reliable CRK methods which provide an asymptotically correct control of an estimate of the maximum of $||\delta(t)||$ on each step, and are referred to as CRK methods with strict defect control (SDC-CRKs). These methods provide an associated estimate of the maximum defect equal to $||\delta(\hat{t})||$, where $\hat{t}$ is the location of the maximum defect in $[t_{i-1}, t_i]$. See [12] for a description and justification of the particular class of methods of this type for orders four to eight.

A numerical ODE method that employs defect control proceeds by taking a sequence of steps from the initial time, t_0, to the final time, t_{end}. To employ defect control, the method has additional features. On each step, in addition to using a discrete formula to obtain an approximate solution at the end of the step, the method also determines a continuous solution approximation for the whole step. The method then determines an estimate of the maximum defect of the continuous approximate solution on that step. If the estimated maximum defect is less than the user-provided tolerance, the step is accepted and the computation continues to the next step. At this point, the step size may be increased if the estimated maximum defect is substantially smaller than the tolerance. If the estimated maximum defect is greater than the tolerance, the step size is said to have failed. In this case, a step selection strategy is employed to determine a smaller trial step size, and a new step is attempted. As is the case for most reliable ODE methods with adaptive step size control, *we assume that the step selection heuristic will not allow the step size to more than double after an accepted step.* Similarly, if the estimated maximum defect is greater than the tolerance, *we assume that the step selection heuristic will ensure that the new trial step size is not less than half the failed step size.*

When a discontinuity arises, this can substantially complicate the process of attempting to obtain an accurate and reliable numerical solution in an efficient manner. See for example, [13], in particular Figure 1 in [13] where it can be seen that the presence of a discontinuity can lead to substantial inefficiencies for a standard adaptive error control ODE solver.

3 Numerical approximation of the solution of the SEIR model with discontinuities

In this section, we first review a standard approach for discontinuity detection and handling *involving the use of user-provided event functions*, and then we introduce our approach for discontinuity detection and handling, through defect sampling, *which does not require any additional information to be provided by the user*.

3.1 Discontinuity detection and handling based on user-provided event functions

An essential consideration for algorithms for discontinuity detection and handling is whether the user is expected to provide an additional function that provides a characterization of the discontinuity. Such a function is referred to an *event function* or a "root function" since the function that characterizes the discontinuity is constructed so that it has a root at $t = t^*$, the time at which the discontinuity occurs. For an example of related work (where event functions are used), see [16].

Assume that the ODE system is written so that $y_2(t)$ corresponds to $E(t)$ in the SEIR model. Then, event functions that characterize the discontinuities arising from interventions are

$$g_1(t, y(t)) = y_2(t) - 25000, \tag{3.1}$$

when $y_2(t)$ is increasing (i.e., when $y_2'(t) > 0$), and

$$g_2(t, y(t)) = y_2(t) - 10000, \tag{3.2}$$

when $y_2(t)$ is decreasing (i.e., when $y_2'(t) < 0$). *Note that, for each of the above event functions, the time, t at which an event function has a root is the time when a discontinuity in the SEIR model occurs.*

When the user can provide an event function, the solver proceeds as follows. At the end of each successful time step, the solver evaluates the event function and compares the sign of the event function at the end of the step with the sign of the event function at the beginning of the time step. When there is no change in sign, the solver proceeds to the next step. When there is a change in sign, the solver makes use of the built-in continuous solution approximation available across the step to perform a search to locate the first time at which the event function equals zero. The solver then steps from the beginning of the time step, t_{i-1}, to this event time and returns to the user with a solution approximation at the event time. The user can then call the solver with a new right-hand side function, $f(t, y(t))$, that characterizes the ODE after the discontinuity. Since the right-hand side function has changed, the solver must be restarted with a *cold start*. (See the next subsection for a description of a cold start).

This approach makes use of a standard initial value ODE solver that has an event detection capability. Such solvers are available in many common computing platforms, e.g., Matlab. As mentioned earlier, while these solvers return a continuous approximate solution, they impose control of an estimate of the error only at a discrete set of points, i.e., at the end of each time step. No control of the error of the continuous approximate solution is provided.

3.2 Automatic discontinuity detection and handling

In some applications, the user may not be able to characterize the time at which a discontinuity arises in terms of an event function. In such cases, *it is necessary for the solver to be able to automatically detect and accurately step past the discontinuity.* We now discuss a strategy for the *automatic* detection and handling of ODEs with discontinuities *that does not require the user to provide event functions that characterize the times at which the discontinuities arise.* The approach is based on earlier work which began in [10] and has evolved over a sequence of several investigations, which have focused on improving the reliability and effectiveness of the numerical methods that are implemented (see, for example, [8, 18], and references therein).

We now describe the algorithm that implements automatic discontinuity detection and handling.

- As a standard part of the defect control time-stepping algorithm described in Section 2.2, the software checks that the estimated maximum defect of the continuous numerical solution is less than the user tolerance. The time-stepping algorithm selects the time-step to attempt to provide an approximate solution whose defect is slightly less than the tolerance. When the step is successful, the discontinuity detection algorithm also checks to see (i) if the defect is *much* less than the tolerance, and (ii) if the current successful step was proceeded by a failed step *for which the defect of the approximate solution was large*, i.e., greater than 1. Since the step size is chosen to attempt

to deliver an approximate solution whose defect is slightly less than the user tolerance, a defect of size greater than 1 represents a very large defect. This tells us that there is a large change in the solution behaviour in the region beyond the right endpoint of the successful step. For the problem we consider in this paper, that large change in the behaviour of the solution corresponds to a discontinuity in the derivative of the solution, i.e., in $f(t, y(t))$. For the problem we consider in this paper, there is no other way that the discontinuity detection can be triggered.

- When the above conditions are satisfied, a potential discontinuity is suspected in the region between the end of the current successful step and the end of the previous failed step. To be more specific, assume that the successful step is from t_{i-1} to $t_i = t_{i-1} + h_i$ and that the previous failed step was from t_{i-1} to $t_{i-1} + H_i$, where $h_i < H_i$, due to our previous assumptions regarding the step size selection heuristic. In fact, since there was a failed step with a large defect, obtained using a step size of H_i, the new step h_i will have been chosen to be as small as possible, i.e., $h_i = \frac{H_i}{2}$.) This means that $t_i < t_{i-1} + H_i$, and the discontinuity is suspected in the region $[t_i, t_{i-1} + H_i] = [t_i, t_{i-1} + 2h_i]$.

- To locate the discontinuity, the defect of the current continuous solution approximation is sampled, using a bisection search, on the interval $[t_i, t_{i-1} + H_i]$, to find the time, $t^* \in [t_i, t_{i-1} + H_i]$, where the defect first exceeds 1. This point, t^*, is the location of the discontinuity. The tolerance employed in the bisection search is TOL.

- When t^* is located, we evaluate the current continuous solution approximation at t^*. The resultant approximate solution value is used as the initial value to restart the computation.

- The computation for the next step is restarted, with a *cold start*. This means that the solver is restarted with a small initial step size and no "memory" of the computation that precedes the cold start. This allows the solver to efficiently step past the discontinuity.

Below we provide the algorithm that corresponds to the process described above. We assume that $[t_{i-1}, t_{i-1} + H_i]$ is rejected with defect > 1, and the next step $[t_{i-1}, t_i]$ is accepted with defect $<$ TOL.

As mentioned earlier, the approach described above is based on the analysis presented in [10]. An essential point here is that the above algorithm automatically detects and locates a discontinuity; the algorithm does not require any information from the user in order to characterize the conditions under which the discontinuities arise.

Furthermore, it is important to note that we do not use the numerical method to step across a discontinuity point. The numerical method is only asked to give solution approximations on the parts of the domain (within tolerance TOL) where the solution is smooth.

4 Numerical results

In this section, we present numerical results that show that, using the algorithm described in Section 3.2, we can compute an accurate approximation to the solution of the SEIR model with discontinuities described earlier in this paper.

Assuming we have an ODE solver (with adaptive stepsize and defect control capability) that computes discrete solution values, y_i, and continuous extensions, $u_i(t)$, the algorithm described in Section 3.2 can be successfully applied to this discontinuous SEIR model.

Algorithm 1 Automatic Discontinuity Detection and Handling in $[t_i, t_{i-1} + 2h_i]$

Input: Scalars t_{i-1}, t_i and TOL; functions $u_i(t)$, the continuous solution approximation associated with the step $[t_{i-1}, t_i]$, and defect(t), returning the defect at a given point t, using $u_i(t)$.

Output: Scalar t^*, the estimated location of discontinuity.

$h_i = t_i - t_{i-1}$.
t^* = bisectCRKd($t_i, t_{i-1} + 2h_i$, TOL) % use bisection to determine t^*
Evaluate $u_i(t)$ at t^* to obtain the solution approximation at t^*.
Do a cold restart from t^*.

function bisectCRKd(a, b, TOL)
while $((b - a) \geq$ TOL) **do**
  $tm = (a + b)/2$
  DEFm = defect(tm)
  if (DEFm > 1) **then**
    $b = tm$
  else
    $a = tm$
  end if
end while
Set $t^* = b$; Return(t^*)

To investigate the reliability and effectiveness of the algorithm described in Section 3.2, we ran it with tolerances of 10^{-4}, 10^{-6}, 10^{-8} and 10^{-10}. As the exact solution to the problem is unknown, we estimate the error of a computed quantity by taking the difference (in absolute value) between the quantity and the same quantity computed with the most severe tolerance (10^{-10}).

In Table 1, we report approximations for the times (i.e., locations) of discontinuities and the corresponding values for the number of exposed individuals at these times. We also present the relative error for each of these quantities, estimated as described above. Although, as mentioned earlier, the problem has 13 discontinuities (see Figure 2), for brevity, we include information for only the last two discontinuity times, denoted by $TD(12)$ and $TD(13)$, and the corresponding exposed values, $E(12)$ and $E(13)$. Note also that, regarding these discontinuity times, one involves the lower limit of $E(t) = 10,000$ and the other involves the higher limit of $E(t) = 25,000$.

From the results of Table 1, we note that the algorithm satisfies the requested tolerance as well as can be expected. This is true for both the times at which the discontinuities occur, as well as the values of the component of the solution corresponding to exposed individuals. Overall, the algorithm is reliable and faithful to the tolerance.

Table 1: Approximate values of last and second last discontinuity times, TD(13) and TD(12), respectively, the corresponding approximate values for the number of exposed individuals, E(13) and E(12), respectively, and the associated relative differences from the most severe tolerance results, obtained using the algorithm described in Section 3.2, for the indicated tolerances. The relative difference (*"Rel Diff"*) can be viewed as an approximation to the relative error of the computed solution.

TOL	$TD(13)$	*Rel Diff*	$TD(12)$	*Rel Diff*
10^{-4}	93.003718841734269	3.5×10^{-5}	92.419612490702121	3.5×10^{-5}
10^{-6}	93.000477491294248	2.3×10^{-7}	92.416386344852526	2.3×10^{-7}
10^{-8}	93.000498170684267	6.2×10^{-9}	92.416406906408923	6.2×10^{-9}
10^{-10}	93.000498747632832		92.416407476858438	
	$E(13)$	*Rel Diff*	$E(12)$	*Rel Diff*
10^{-4}	25000.670390978328	2.7×10^{-5}	9999.9995946587187	4.1×10^{-8}
10^{-6}	25000.013092744615	5.2×10^{-7}	9999.9994753177125	5.2×10^{-8}
10^{-8}	25000.000115099829	4.6×10^{-9}	9999.9999986860366	1.3×10^{-10}
10^{-10}	25000.000000883927		9999.9999999641277	

In Table 2, we report the number of derivative evaluations, i.e., evaluations of $f(t, y)$, performed by the algorithm for each of the tolerances.

Table 2: Number of derivative evaluations (NDEVS) when using the algorithm in Section 3.2 (automatic detection and handling of discontinuities), for the indicated tolerances, TOL.

TOL	*NDEVS*
10^{-4}	1359
10^{-6}	1999
10^{-8}	2248
10^{-10}	3762

We observe from Table 2 that the number of function evaluations increases in a reasonably smooth way as the requested tolerance is decreased.

To provide some context for the results given in Table 2, we also provide results obtained using the Matlab solver, ode45. (This solver implements the DOPRI5 method which is based on a 4(5) Runge-Kutta formula pair [7]). *We employ ode45 using event detection*, as described in Section 3.1. When ode45 is given event functions and allowed to operate in event detection mode to find the times at which the discontinuities arise, it is able to compute accurate, reliable solutions to the SEIR model with interventions. Regarding the approximation of the times of the discontinuities and the corresponding values of $E(t)$, we found that ode45 delivers results of comparable accuracy to those shown in Table 1.

In Table 3, we report the number of derivative evaluations required by ode45 in event detection mode for tolerances of $10^{-4}, 10^{-6}, 10^{-8}$, and 10^{-10}. We note that the results from Table 3 cannot be directly compared with those in Table 2 because (i) the DOPRI5 method used in ode45 is of lower order than the Runge-Kutta method used in the implementation of the algorithm from Section 3.2, (ii) the two methods employ different types of error control (ode45 controls an estimate of the error of the approximate solution

Table 3: Number of derivative evaluations (NDEVS) for ode45 (using its event detection capability), for the indicated tolerances, TOL.

TOL	*NDEVS*
10^{-4}	518
10^{-6}	770
10^{-8}	1514
10^{-10}	3416

only at a discrete set of points in the problem domain, while the algorithm from Section 3.2 controls the defect of the continuous approximate solution across the entire domain), and, (iii) *ode45 is given extra information about the locations of the discontinuities through the event functions provided by the user.* Nonetheless, comparing the results from Table 2 with the results from Table 3, we observe that the implementation of the algorithm from Section 3.2 requires more derivative evaluations than ode45, although the difference in the number of derivative evaluations becomes less significant as the tolerance gets sharper. Furthermore, we note that, as the tolerance gets sharper, the number of derivative evaluations for ode45 increases faster than for the algorithm of Section 3.2. Finally, we emphasize that, for ode45 with event detection, the user is required to interact with the solver in a more sophisticated way, in that, not only is the definition of the model needed, but also the definition of (one or more) event functions is needed, and the handling of stopping and restarting the computation at the discontinuity points must be done by the user.

The point of introducing Tables 2 and 3 is not to compare the efficiencies of the two methods and respective software, but to show that the algorithm we describe in Section 3.2 is able to deliver accurate results at a cost that is somewhat comparable to the approach described in Section 3.1, even though the latter approach requires the user to provide extra information and to interact with the solver in a more complicated way.

5 Summary and conclusions

In this paper, we have considered an SEIR COVID-19 ODE model representing the spread of the virus, subject to interventions imposed, under certain conditions, to reduce the spread of the virus; these interventions are removed when the spread of the virus is sufficiently reduced. The derivative of the solution of this system of ODEs has discontinuities corresponding to the imposition and removal of these interventions. We note that the model we have chosen should not be considered to be a definitive COVID-19 model; it is simply an example of a typical model of this type; similar models that involve solution components that exhibit rapid growth and decay over certain regions of the domain and that involve interventions leading to discontinuities in the derivative of the solution to the model will present similar challenges for standard ODE solvers.

This paper includes numerical results that show that an incorrect solution is obtained when a standard ODE solver is employed to attempt to solve the model, using a typical approach for the introduction of the interventions. This paper also shows that it is possible to develop a numerical method that can compute a high accuracy, defect-controlled continuous solution approximation that can be used together with a defect sampling algorithm to accurately locate and then step past each discontinuity, allowing an accurate

solution approximation to be obtained.

The approach described in this paper delivers a continuous approximate solution that can be evaluated at any point in $[t_0, t_{end}]$. The accuracy of this continuous approximate solution is imposed through control of an estimate of its defect. The continuous approximate solution that is returned will have an estimated defect that is less than the user tolerance across the entire domain. The software requires from the user only the information that is required to define the ODE system and the user tolerance. *The significant advantage of this approach is that it can automatically locate the discontinuities; it does not require the user to provide additional event functions that characterize the discontinuities.*

Author contributions

W. H. Enright: conceptualization, formal analysis, funding acquisition, methodology, software, writing original draft; C. C. Christara: conceptualization, formal analysis, funding acquisition, methodology, software, writing review & editing, visualization; P. H. Muir: formal analysis, funding acquisition, methodology, software, writing review & editing.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors would like to thank Jonathan Calver and Humaid Agowun for helpful discussions. Author W. H. Enright's work was supported in part by the NSERC Discovery Grant RGPIN-2016-05595. Author C. C. Christara's work was supported in part by the NSERC Discovery Grant RGPIN-2021-03502. Author P. H. Muir's work was supported in part by the NSERC Discovery Grant RGPIN-2017-05811.

Conflict of interest

The authors declare no conflict of interest.

References

[1] H. Agowun and P. Muir, *Performance analysis of a collection of ODE solvers on COVID-19 models with discontinuities*, Saint Mary's University, Department of Mathematics and Computing Science, Technical Report 2022_001, https://cs.smu.ca/tech_reports/, (2022).

[2] J. Arino and C. McCluskey, *Effect of a sharp change of the incidence function on the dynamics of a simple disease*, Journal of Biological Dynamics, 4 (2010), pp. 490–505.

[3] M. Bisiacco and G. Pillonetto, *Covid-19 epidemic control using short-term lockdowns for collective gain*, Annual Reviews in Control, 52 (2021), pp. 573–586.

[4] J. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge–Kutta and General Linear Methods*, John Wiley and Sons, New Jersey, 1987.

[5] X. CHEN, C. XIANG, AND Y. YANG, *Sliding mode control of the susceptible-exposed-asymptomatic-infected-recovered model with vaccination*, Mathematical Methods in the Applied Sciences, 48 (2024), pp. 3936–3955.

[6] C. CHRISTARA, *The importance of being computationally accurate: the case of COVID-19*. CSC336 course sample paper, University of Toronto, Toronto, Ontario, Canada, `www.cs.toronto.edu/~ccc/Courses/cssu.pdf`, April 2020.

[7] J. DORMAND AND P. PRINCE, *A family of embedded Runge-Kutta formulae*, Journal of Computational and Applied Mathematics, 6 (1980), pp. 19–26.

[8] W. ENRIGHT AND H. HAYASHI, *A delay differential equation solver based on a continuous Runge-Kutta method with defect control*, Numerical Algorithms, 16 (1997), pp. 349–364.

[9] W. ENRIGHT, K. JACKSON, S. NØRSETT, AND P. THOMSEN, *Interpolants for Runge-Kutta formulas*, ACM Transactions on Mathematical Software (TOMS), 12 (1986), pp. 193–218.

[10] W. ENRIGHT, K. JACKSON, S. NORSETT, AND P. THOMSEN, *Effective solution of discontinuous IVPs using a Runge-Kutta formula pair with interpolants*, Applied Mathematics and Computation, 27 (1988), pp. 313–335.

[11] W. ENRIGHT AND P. MUIR, *New interpolants for asymptotically correct defect control of BVODEs*, Numerical Algorithms, 53 (2010), pp. 219–238.

[12] W. ENRIGHT AND L. YAN, *The reliability/cost trade-off for a class of ODE solvers*, Numerical Algorithms, 53 (2010), pp. 239–260.

[13] C. GEAR AND O. OSTERBY, *Solving ordinary differential equations with discontinuities*, ACM Transactions on Mathematical Software, 10 (1984), pp. 23–44.

[14] H. HETHCOTE, *The mathematics of infectious diseases*, SIAM Review, 42 (2000), pp. 599–653.

[15] V. PICCIRILLO, *Nonlinear control of infection spread based on a deterministic SEIR model*, Chaos, Solitons & Fractals, 149 (2021), p. 111051.

[16] L. SHAMPINE AND S. THOMPSON, *Event location for ordinary differential equations*, Computers and Mathematics with Applications, 39 (2000), pp. 43–54.

[17] Y. XIAO, X. XU, AND S. TANG, *Sliding mode control of outbreaks of emerging infectious diseases*, Bulletin of mathematical biology, 74 (2012), pp. 2403–2422.

[18] H. ZIVARIPIRAN AND W. ENRIGHT, *An efficient unified approach for the numerical solution of delay differential equations*, Numerical Algorithms, 53 (2010), pp. 397–417.