Numerical Schemes for Solving and Optimizing Multiscale Models with Age of Hepatitis C Virus Dynamics

Vladimir Reinharz[a], Harel Dahari[b], Danny Barash[a,*]

[a] *Department of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel*
[b] *The Program for Experimental & Theoretical Modeling, Division of Hepatology, Department of Medicine, Loyola University Medical Center, Maywood, IL, 60153 USA*

Abstract

Age-structured PDE models have been developed to study viral infection and treatment. However, they are notoriously difficult to solve. Here, we investigate the numerical solutions of an age-based multiscale model of hepatitis C virus (HCV) dynamics during antiviral therapy and compare them with an analytical approximation, namely its long-term approximation. First, starting from a simple yet flexible numerical solution that also considers an integral approximated over previous iterations, we show that the long-term approximation is an underestimate of the PDE model solution as expected since some infection events are being ignored. We then argue for the importance of having a numerical solution that takes into account previous iterations for the associated integral, impeding the use of canned solvers. Second, we demonstrate that the governing differential equations are stiff and the stability of the numerical scheme should be considered. Third, we show that considerable gain in efficiency can be achieved by using adaptive stepsize methods over fixed stepsize methods for simulating realistic scenarios when solving multiscale models numerically. Finally, we compare between several numerical schemes for the solution of the equations and demonstrate the use of a numerical optimization scheme for the parameter estimation performed directly from the equations.

Keywords: Hepatitis C virus, Multiscale model, Age-structured model, Numerical solution, Adaptive stepsize integration, Partial differential equations

1. Introduction

Chronic hepatitis C viral (HCV) infection affects approximately 70 million people worldwide and is the primary cause of liver cirrhosis, liver cancer and liver transplant [1]. There is no vaccine for HCV and for more than a decade the standard-of-care consisting of pegylated interferon-alpha (IFN) and ribavirin was suboptimal. The recent advent of direct-acting antivirals (DAAs) that provide interferon-free, all-oral treatment yielding cure rates exceeding 90% with pangenotypic activity, shorter durations of therapy (8–24 weeks) as compared to IFN-based therapy (24–48 weeks) is considered one of the greatest achievements in medicine [2]. However, despite of these highly effective DAAs, many challenges remain, such as finding an optimal approach to current DAA failures and the elimination of HCV infection and DAAs cost, which is a significant barrier in treating the populations that are most affected by HCV. Thus, there exists a significant need for affordable therapy, with much shorter treatment durations and vaccine development [3].

Mathematical modeling of HCV kinetics has considerably advanced in recent years. It has improved our understanding of intracellular viral genome dynamics [4, 5, 6, 7], T-cell dynamics, and the quantitative events that underlie the immune response to pathogens [8, 9]. The standard model for HCV kinetics during treatment provided many insights into the effectiveness and mechanism of action of interferon-alpha and ribavirin (reviewed in [10, 11]). The models were able to retrospectively predict the duration of treatment needed for HCV eradication (cure) [12, 13, 14] and more recently used in real-time (on treatment) to predict the duration of IFN-free therapy with silibinin+ribavirin needed to achieve

[*] Corresponding author
Email addresses: reinharz@post.bgu.ac.il (Vladimir Reinharz), dbarash@cs.bgu.ac.il (Danny Barash)

cure [15]. In the age of DAAs, new models have been developed to meet the challenges of these new agents such as drug resistance [16]. Notably, the first age-based multiscale mathematical model for HCV kinetics was developed [4, 17, 18] and provided a comprehensive understanding of the nature of viral kinetic patterns observed in patients treated with IFN, HCV protease inhibitors (telaprevir and danoprevir), or HCV NS5A inhibitor daclatasvir and their modes of action. Mathematical models are also valuable in understanding the in vivo dynamics of viruses that trigger both persistent infection (e.g. HIV-1 [8, 19, 20, 9], hepatitis B virus [21, 22, 23], hepatitis D virus [24, 25], Theiler murine encephalomyelitis virus [26], herpes simplex virus [27] and HCV [28, 29, 30]) and acute infection (e.g., influenza A [31, 32, 33] and ebola [34]).

In the context of HCV kinetics, multiscale models are an extension to the classical biphasic model [30] that was introduced in 1998 and treated the infected cell as a "black box", producing virions but without any consideration of the intercellular viral RNA replication and degradation within the infected cell [6, 5, 35]. The biphasic model is a set of three ordinary differential equations (ODEs) with three variables: uninfected target cells (T), productively infected cells (I), and free virus (V). The multiscale models consider the intercellular viral RNA in an additional equation for the variable (R), with the introduction of age-dependency in addition to time-dependency, making it a partial differential equation (PDE) model. The multiscale models study the dynamics of HCV infection under therapy with DAAs and because they include both the intracellular viral RNA replication/degradation and the extracellular viral infection with age-dependency in addition to time-dependency, they are considerably more difficult to solve compared to the biphasic model. Analytical approximations were derived [18, 36, 4], namely the short-term and long-term approximations. While the short-term has been shown to be precise only in the first half-day of treatment, the long-term is in agreement at the asymptotics with a simple numerical solution that utilized a general ODE solver [18].

The aim of this paper is to significantly improve the numerical solution presented in [18] that used a canned solver (an ODE solver called from by a command within a program that is written in a higher level langauage such as Matlab and Mathematica, or Python). This makes the numerical solution framework as explained herein a flexible and robust entity alongside the analytical approximations. As it turns out, because of the properties of the multiscale model and the fact that the differential equations are stiff, some advanced numerical methods that involve adaptive stepsize are needed. To begin with, the use of a canned solver should be replaced with a fully-written solver in the code for the application because of the additional integral introduced in the multiscale model for the variable V that needs to be computed at each time step. Unlike the construction of numerical schemes in other applications, for example in the nonlinear diffusion of digital images [37, 38, 39] where accuracy can be limited, herein it is adviseable to construct a stable and efficient scheme that belongs to the Runge-Kutta family with at least a fourth order of accuracy. However, due to the nature of the differential equations that are stiff and the additional integral that needs to be evaluated at each time step, implicit solvers with adaptive stepsize are considerably more stable and efficient than the standard Runge-Kutta fourth order. Starting for simplicity in building up from explicit and implicit first order schemes, extending to explicit and implicit fourth order schemes and noticing that the differential equations are stiff, we reach explicit schemes with adaptive stepsize [40] that are by an order of magnitude more efficient than fixed stepsize for realistic simulations of viral infection of several days. We then implement implicit schemes with adaptive stepsize [41] that are considerably more stable. The various numerical schemes are described and compared to each other, concluding with an implicit adaptive stepsize integration scheme that is both efficient and stable for use in multiscale models with age of hepatitis C virus dynamics [42]. Finally, the Levenberg-Marquardt optimization scheme is illustrated for performing parameter estimation directly from the equations of the multiscale models.

2. Methods

2.1. The standard HCV model

The standard model that has been used and modified for studying hepatitis C viral dynamics is the Neumann et al. model [30]. The three variables this model keeps track of are the target cells T, in Eq. (1a), the infected cells I in Eq. (1b) and the free virus V in Eq. (1c). The target cells T are produced at constant rate s, die at per capita rate d and are infected by virus V at constant rate β. The infected cells I increase with the new infections at rate $\beta V(t)T(t)$ and die at constant rate δ. The virus V is produced at rate p by each infected cells and is cleared at constant rate c. The ϵ term denotes the effectiveness of the anti-viral treatment that decreases the production from p to $(1-\epsilon)p$. Formally the ensemble of ODEs for this model is:

$$\frac{\mathrm{d}T(t)}{\mathrm{d}t} = s - \beta V(t)T(t) - dT(t) \tag{1a}$$

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t} = \beta V(t)T(t) - \delta I(t) \tag{1b}$$

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = (1-\epsilon)pI(t) - cV(t). \tag{1c}$$

From the mathematical perspective, the standard model is realtively much simpler than the multiscale model. Although it is nonlinear, it can be solved analytically when assuming that T is constant.

2.2. The HCV age-based multiscale model

The multiscale model of hepatitis C virus (HCV) dynamics has been formulated in recent years [18, 36, 4] as a more complex system in order to study HCV dynamics during therapy. It keeps track as in the standard viral dynamics model of uninfected target cells in equation (2a), productively infected cells in equation (2b), and free virus in equation (2c) along with considering the intracellular viral RNA dynamics in an infected cell by adding equation (2d).

The equations were formulated as follows:

$$\frac{\mathrm{d}T(t)}{\mathrm{d}t} = s - \beta V(t)T(t) - dT(t) \tag{2a}$$

$$\frac{\partial I(a,t)}{\partial t} + \frac{\partial I(a,t)}{\partial a} = -\delta(a)I(a,t) \tag{2b}$$

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a - cV(t) \tag{2c}$$

$$\frac{\partial R(a,t)}{\partial t} + \frac{\partial R(a,t)}{\partial a} = (1-\epsilon_\alpha)\alpha(a) - \left[(1-\epsilon_s)\rho(a) + \kappa\mu(a)\right]R(a,t), \tag{2d}$$

subject to the initial and boundary conditions $V(0) = \bar{V}$, $T(0) = \bar{T}$, $I(0,t) = \beta V(t)T(t)$, $I(a,0) = \bar{I}(a)$, $R(0,t) = 1$, and $R(a,0) = \bar{R}(a)$.

Model parameters of T, Eq.(2a), and I, Eq. (2b), are similar to the standard model where a is the age of infection and t is the time duration from therapy initiation. The quantity of intracellular viral RNA R, in Eq. (2d), depends on its production α its degradation μ and expulsion from the cell ρ. The quantity of free viruses V shown in Eq. (2c) depends on the number of assembled and released virions and their clearance rate c. While those parameters should depend on the cell age a, in practice they are considered constant.

An important consideration in this model is that the treatment starts after the infection has reached its steady state. The steady states of the different functions are $\bar{R}(a)$, $\bar{I}(a)$, $\bar{V}$ and $\bar{T}$. Given N, the total number of virions produced by a cell in its lifespan, it can be shown that those values are:

$$\begin{aligned} N &= \frac{\rho(\alpha+\delta)}{\delta(\rho+\mu+\delta)} \\ \bar{R}(a) &= \frac{\alpha}{\rho+\mu} + \left(1 - \frac{\alpha}{\rho+\mu}\right)e^{-(\rho+\mu)a} \\ \bar{I}(a) &= \beta\bar{V}\bar{T}e^{-\delta a} \\ \bar{T} &= c/(\beta N) \\ \bar{V} &= (\beta N s - dc)/(\beta c). \end{aligned} \tag{3}$$

Unlike the standard model, three different antiviral effects of therapy can be distinguished in the multiscale model. The decrease in viral RNA synthesis is represented by ε_α, the reduction in secretion by ε_s and the increase in viral degradation by $\kappa \geq 1$.

Through the method of characteristics, as was derived in [18] and explained in more detail herein in Appendix A, an analytical solution can be found the variable $R(a,t)$. The same method can be applied to derive $I(a,t)$ solution. The ensemble of equations (2) represent the full model.

$$R(a,t) = \begin{cases} & a < t \\ \frac{\alpha}{\rho+\mu} + \left(1 - \frac{\alpha}{\rho+\mu}\right) \mathrm{e}^{-[\rho+\mu]a} & \\ & a \geq t \\ \frac{\alpha}{(\rho+\mu)} + \left(\bar{R}(a-t) - \frac{\alpha}{(\rho+\mu)}\right) \mathrm{e}^{-(\rho+\mu)t} & \end{cases} \tag{4}$$

$$I(a,t) = \begin{cases} \beta V(t-a) T(t-a) \mathrm{e}^{-\delta a} & a < t \\ \bar{I}(a-t) \mathrm{e}^{-\delta t} & a \geq t \end{cases} \tag{5}$$

From equations (2) it can be noticed that computing $V(t)$ necessitates an integral. If $a < t$, in other words the cell age is younger than the time of treatment, i.e., infection occurs after initiation of treatment, the term $I(a,t)$ of the integral in Eq. (5) depends itself on $V(t)$ and $T(t)$. As it will be shown, this makes the analytical solution approximative and impedes the use of canned numerical solvers for this system of equations.

2.3. Analytical solution

To derive an analytical solution for $V(t)$ the dependancies of $I(a,t)$ makes it difficult unless some additional assumptions are taken into account. The solution will be divided into two cases, the short term approximation and long term approximation [18, 36], which mainly differ on how $I(a,t)$ is treated.

2.3.1. Short term approximation

For the short term approximate solution, it is assumed that after therapy is initiated infected cells remain at their steady-state distribution or in other words, new infections occur at the same rate as before therapy initiation [18]. Thus, one can replace $I(a,t)$ by its steady state solution, $I(a,t) = \bar{I}(a) = \beta \bar{V} \bar{T} \mathrm{e}^{-\delta a}$. The integral in V, $\int_0^\infty R(a,t) I(a,t)\,\mathrm{d}a$, is then computed in two parts. The first part is from 0 to t considering the equation of $R(a,t)$ when $a < t$. The second part of the integral, from t to ∞ is computed considering the $R(a,t)$ equations when $a \geq t$. Formally, given $A = (1-\varepsilon_\alpha)\alpha$ and $B = (1-\varepsilon_s)\rho + \kappa\mu$ the short term approximation is:

$$\frac{V(t)}{V_0} = e^{-ct} + (1-\varepsilon_s)\frac{c\rho}{N}\left\{\frac{A+\delta}{(B+\delta)c\delta}\left(1 - e^{-ct}\right) + \frac{1}{B+\delta-c}\left(\frac{N}{\rho} - \frac{A+\delta}{(B+\delta)\delta}\right)\left(e^{-ct} - e^{-(B+\delta)t}\right)\right\}, \tag{6}$$

where N is as in (3).

2.3.2. Long term approximation

Unlike the short approximation, in the long term approximation new infection is ignored since the onset of therapy. In other words, $I(a,t) = R(a,t) = 0$ if $a < t$. This implies that the integral needed to compute $V(t)$ is simplified to $\int_t^\infty R(a,t) I(a,t)\,\mathrm{d}a$, and once again we can consider $I(a,t) = \beta \bar{V} \bar{T} \mathrm{e}^{-\delta a}$.

Another important distinction is that the term $e^{-\gamma t}$ was included in $R(a,t)$ equation assuming that in addition of α being initially inhibited by the factor $1 - \varepsilon_\alpha$, also decreases with time on treatment due to the decay of replication templates (e.g. replication complexes or negative strand HCV RNA), where parameter γ represents slowing of viral RNA synthesis. Without it, R would converge to a non zero steady state and its inclusion has been shown to be consistent with an intracellular model [5] as described in detail in [18]. This simply adds a term to the PDE of R as follows.

$$\frac{\partial R(a,t)}{\partial t} + \frac{\partial R(a,t)}{\partial a} = (1-\epsilon_\alpha)\alpha(a)\mathrm{e}^{-\gamma t} - \left[(1-\epsilon_s)\rho(a) + \kappa\mu(a)\right] R(a,t) \tag{7}$$

The method of characteristics was reapplied as previously to solve the PDE. From these equations, given the simplifications on I and R, it was straight-forward to derive the long-term approximation for V. Given A, B and N as in the previous section, the long-term approximation is:

$$\frac{V(t)}{V_0} = e^{-ct} + (1-\varepsilon_s)\frac{c\rho}{N}\left\{\frac{A}{(B-\gamma)\delta(\delta+\gamma-c)}\times\left(e^{-ct}-e^{-(\delta+\gamma)t}\right)+\frac{1}{B+\delta-c}\times\left(\frac{N}{\rho}-\frac{A}{(B-\gamma)\delta}\right)\left(e^{-ct}-e^{-(B+\delta)t}\right)\right\}. \tag{8}$$

2.4. Numerical methods

We implemented a variety of numerical methods from the simplest to the more advanced ones that are tailored to our needs, starting from first order methods and describing them in detail in order for our derivations to remain self-contained. We then build from bottom up towards higher order methods, adaptive stepsize methods and implicit schemes in order to better characterize the equations of our model using advanced numerical schemes [43, 44].

The general scheme of a numerical method in our context requires to solve a function $f(t, x)$ which depends on time and a variable x. In this section we explore the solutions of the functions T and V and—except in subsection 2.4.4 about the Rosenbrock method—$f(t, x)$ will be always defined as follows. As in Eq. (2) in the case of V we have $f(t,x) = (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a - c * x$ and for T, $f(t,x) = s - d * x - \beta V(t) * x$.

2.4.1. Integral implementation challenges

The main obstacle for solving the system of differential equations is that the integral term $(1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a$ in Eq. (2c) makes problematic the direct use of a general ODE solver and limits it to only special cases relating to the time steps. This requires an *ad hoc* implementation.

Another problem caused by the integral is that it does not allow to directly apply canned parameter fitting methods to the complete multiscale model. To overcome this challenge, the long term approximation has been used to determine some parameters [18]. One limitation of such an approach is that it is limited to the multiscale model under treatment, and could only be used to approximate the variables c, δ and ε_α. As a byproduct, we briefly explore how a specialized implementation of the Levenberg–Marquardt numerical method can be developed to perform parameter fitting directly from the multiscale model equations. In Sec. 2.5, we apply it to the parameter s for illustration. We validate our implementation in Sec. 3.6.

No analytical solution can be computed for the aforementioned integral and it must therefore be solved numerically unless analytical approximations are used. This is because of the function $I(a, t)$ shown in Eq. (5). At every time step t, the value of I depends on previously computed values of V and T, which cannot be accessed with general ODE solvers.

In order to compute the integral, a first assumption is that its upper bound is 100. Since the integral values represent the age of a cell we assume that cells die after 100 days.

The integral over $a \in [0, 100]$ can be divided into two cases. For $a > t$, the solutions are pre-computed since they do not depend on any other function. When $a < t$, we need to know some values of V and T. Given D, the number of days the system is to be solved for, the strategy to evaluate this part of the integral depends on the type of framework used for solving the ODE system, as described next.

Non-adaptive step framework. The ODE system is solved with a fixed time step h. This implies that for every integer $\tau \in [0, D/h]$ the values $V(\tau h)$ and $T(\tau h)$ are known. For every value $t = \tau h$ the integral needs to be computed. The Simpson's rule method is used to evaluate the integral, therefore a time step h_a must be selected. Similarly to τ, we define an integer $\tau_a \in [0, 100/h_a]$. Given t, for every $a = \tau_a h_a$ (we recall that $a < t$), the values $V(t-a)$ and $I(t-a)$ are required. Thus, there exist a third integer $\tilde{\tau} \in [0, D/h]$ such that $\tilde{\tau}h = t - a$. Therefore we should have $\tau = \tau_a h_a/h + \tilde{\tau}$. Since τ, τ_a and $\tilde{\tau}$ are integers a solution exists if h_a is a multiple of h. In the following implementation, setting $h_a = 10h$ provides good results.

Adaptive step framework. In that case, the time step h is not fixed and therefore we cannot use Simpson's rule. Given h_0 the initial time step, we limit the value of h to $10h_0$. The integral step size h_a will also be variable and this condition ensures that it will be at most $10h_0$, to maintain the accuracy as in the non-adaptive case.

Given a list of $m+1$ time steps $\mathfrak{T} = [t_0, \cdots t_m]$ such that $t_0 = 0$, $t_i = h_0 + \cdots + h_{i-1}$, and $t = t_m$ is the latest time for which the values of V and I were computed. The strategy consists in computing a quadratic interpolation over carefully chosen values of a. For $i \in [0,|\mathfrak{T}|]$ let a_i be $t_m - t_{m-i}$. By construction the difference between a_i and a_{i+1} is $t_{m-i} - t_{m-i-1} \leq 10h_0$.

To optimize the process, we do not record all values of V and T but instead proceed as follows. Given $\mathfrak{T}$ a list of time steps, $\mathfrak{T}[-2]$ the penultimate element of $\mathfrak{T}$, and $\mathfrak{T}[-1]$ its last element. At the next iteration values are computed for time t. If $t - [T-2] > 10h_0$, we append t to $\mathfrak{T}$. If it is not, we replace the last element $\mathfrak{T}[-1]$ by t. The distance between any two consecutive elements of $\mathfrak{T}$ is at most $10h_0$ by construction, as desired.

2.4.2. First order methods

The Euler methods are the simplest first order methods. They consist of two possible schemes: Forward Euler that is explicit and Backward Euler that is implicit. The **Forward Euler** scheme is as follows. Given V_n and T_n and a step h we simply update the equations:

$$\begin{aligned} V_{n+1} &= V_n + h\times \\ &\left((1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a - c * V_n\right) \\ T_{n+1} &= T_n + h \times (s - dT_n - \beta V_n T_n) \end{aligned} \tag{9}$$

The complementary scheme to Forward Euler is called **Backward Euler**, and is an implicit scheme. It consists of solving the equations $y_{n+1} = y_n + h \times f(t+h, y_{n+1})$ which yields as answers

$$\begin{aligned} V_{n+1} &= V_n + \frac{h(1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a}{1+ch} \\ T_{n+1} &= \frac{shT_n}{1+hd+h\beta V(t+h)}, \end{aligned} \tag{10}$$

where $V(t+h) = V_{n+1}$.

2.4.3. Accurate methods with fixed stepsize

From the first order Euler methods described above, in order to achieve better accuracy we advance to Runge–Kutta 4th order (RK4) and Gauss-Legendre 4th order. Gauss-Legendre is an implicit version of RK4, which is a collocation method based on the points of Gauss-Legendre quadrature [44]. As will be shown, Gauss-Legendre is more stable than RK4 for our problem.

In general, the system of equations to solve **Runge–Kutta** order s is

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{i=1}^{s} b_i k_i \text{ , such that} \\ k_i &= f\left(t + q_i h, y_n + h\sum_{j=1}^{i-1} a_{ij}k_j\right) \end{aligned} \tag{11}$$

and given a_{zw}, b_z, q_z as in Table. 1.

Different ensembles of equivalent constants a, b and q can be determined. In all the cases presented here they were selected from [43].

q_i	a_{ij}			
0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
$b_i=$	1/6	1/3	1/3	1/3

Table 1: Butcher tableau for Runge–Kutta 4th order

Let us first look at $V(t)$ for V_{n+1} at time $t+h$:

$$\begin{aligned} k_1^V &= (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a - cV_n \\ &= f(t,V_n) \\ k_2^V &= (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t+q_2h)I(a,t+q_2h)\,\mathrm{d}a \\ &\quad - c\left(V_n + ha_{21}k_1^V\right) \\ &= f(t+q_2h, V_n + ha_{21}k_1^V) \\ k_3^V &= (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t+q_3h)I(a,t+q_3h)\,\mathrm{d}a \\ &\quad - c\left(V_n + ha_{32}k_2^V\right) \\ &= f(t+q_3h, V_n + ha_{31}k_2^V) \\ k_4^V &= (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t+q_4h)I(a,t+q_4h)\,\mathrm{d}a \\ &\quad - c\left(V_n + ha_{43}k_3^V\right) \\ &= f(t+q_4h, V_n + ha_{43}k_3^V)\text{, such that} \\ V_{n+1} &= V_n + h\left(b_1k_1^V + b_2k_2^V + b_3k_3^V + b_4k_4^V\right). \end{aligned} \tag{12}$$

Similarly in the case of $T(t)$ for T_{n+1} at time $t+h$ we have (note that $q_1 = 0$):

$$\begin{aligned} k_1^T &= s - dT_n - \beta V(t)T_n \\ &= s - dT_n - \beta V_nT_n \\ k_2^T &= s - d\left(T_n + ha_{21}k_1^T\right) \\ &\quad - \beta V(t+q_2h)\left(T_n + ha_{21}k_1^T\right) \\ k_3^T &= s - d\left(T_n + ha_{32}k_2^T\right) \\ &\quad - \beta V(t+q_3h)\left(T_n + ha_{32}k_2^T\right) \\ k_4^T &= s - d\left(T_n + ha_{43}k_3^T\right) \\ &\quad - \beta V(t+q_4h)\left(T_n + ha_{43}k_3^T\right)\text{, such that} \\ T_{n+1} &= T_n + h\left(b_1k_1^T + b_2k_2^T + b_3k_3^T + b_4k_4^T\right). \end{aligned} \tag{13}$$

Note that $V(t+q_ih) = V_t + q_ihk_i^V$.

An implicit version of Runge–Kutta with a fixed stepsize is **Gauss-Legendre**. Similar to Runge–Kutta order s, for the main equations we have $y_{n+1} = y_n + h\sum_{i=1}^s b_ik_i$. But the value k_i is slightly different, notice the bound of the sum in: $k_i = f\left(t + q_ih, y_n + h\sum_{j=1}^s a_{ij}k_j\right)$.

Given $s = 2$ (for simplicity), we have used the Butcher tableau shown in Table 2.

q_i	$a_{i,j}$	
$1/2 - \sqrt{3}/6$	$1/4$	$1/4 - \sqrt{3}/6$
$1/2 + \sqrt{3}/6$	$1/4 + \sqrt{3}/6$	$1/4$
$b_i =$	$1/2$	$1/2$

Table 2: Butcher tableau for Gauss-Legendre

We need to solve, first for V, the following implicit equations:

$$\begin{aligned} k_1^V &= (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t+q_1h)I(a,t+q_1h)\,\mathrm{d}a \\ &\quad - c\left(V_n + ha_{11}k_1^V + ha_{12}k_2^V\right) \\ &= f(t+q_1h, V_n + ha_{11}k_1^V + ha_{12}k_2^V) \\ k_2^V &= (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t+q_2h)I(a,t+q_2h)\,\mathrm{d}a \\ &\quad - c\left(V_n + ha_{21}k_1^V + ha_{22}k_2^V\right) \\ &= f(t+q_2h, V_n + ha_{21}k_1^V + ha_{22}k_2^V), \end{aligned} \tag{14}$$

which can be re-written as:

$$\begin{aligned} k_1^V &= A + B \times k_2^V \\ k_2^V &= \Bigg((1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a \\ &\quad - cV_n - cha_{21}A\Bigg) \times \frac{1}{1 + ch\left(a_{22} + a_{21}B\right)}, \end{aligned} \tag{15}$$

where $A = \dfrac{(1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a - cV_n}{1 + cha_{11}}$

and $B = \dfrac{-cha_{12}}{1+cha_{11}}$. Similarly for T we have:

$$\begin{aligned} k_1^T &= s - d\left(T_n + ha_{11}k_1^T + ha_{12}k_2^T\right) \\ &\qquad - \beta V(t+q_1h)\left(T_n + ha_{11}k_1^T + ha_{12}k_2^T\right) \\ k_2^T &= s - d\left(T_n + ha_{21}k_1^T + ha_{22}k_2^T\right) \\ &\qquad - \beta V(t+q_2h)\left(T_n + ha_{21}k_1^T + ha_{22}k_2^T\right), \end{aligned} \tag{16}$$

which can be re-written as:

$$\begin{aligned} k_1^T &= \frac{C + Dk_2^T}{E} \\ k_2^T &= \frac{s - (d+\beta V(t+q_2h))(T_n + ha_{21}C)}{1 + h(d+\beta V(t+q_2h))(a_{21}D + a_{22})}, \end{aligned} \tag{17}$$

where $C = s - dT_n - \beta V(t+q_1h)T_n$, $D = -ha_{12}\left(d+\beta V(t+q_1h)\right)$, and $E = 1 + ha_{11}\left(d+\beta V(t+q_1h)\right)$. We recall that $V(t+q_ih) = V_n + q_ihk_i^V$.

In practice, the two functions R and I are only evaluated at t and not $t+q_xh$. Two reasons justify that choice. First, as previously discussed in Sec. 2.4.1, the integral over a is divided into two parts, before and after t (i.e., $\int_0^{100} \mathrm{d}a = \int_0^t \mathrm{d}a + \int_t^{100} \mathrm{d}a$). The latter change would shift the integral to $\int_0^{t+q_xh} \mathrm{d}a + \int_{t+q_xh}^{100} \mathrm{d}a$. We argue that those values are close to each other in realistic setups. The maximal q_x, shown in Table 1, is 1. Given that we chose (see Sec. 2.4.1) the integration step h_a as $10h$, the difference between t and $t+q_xh$ is at most $\frac{h_a}{10}$, a tenth of a step. A simulation over two days requires 200 steps to evaluate the integral, with $h_a = 0.01$. Therefore it represents less than 0.05% of the integral's value. Second, to compute that additional portion in the first part of the integral, it requires to know the value of T and V at a time greater than t. To do so would require a more uncertain approximation than the one being evaluated, increasing the total error. The same reasoning has been used in [45] to show that in the Rosenbrock method, described in the next section, the Jacobian should be computed only once with the initial values and not with the intermediary values.

2.4.4. Accurate methods with adaptive stepsize

As for the previous methods with a fixed stepsize, two adaptive stepsize methods are implemented. Dormand-Prince [40], which is explicit, and the Rosenbrock method [41], which is implicit. Similar integration methods to Dormand-Prince (RKDP) are Fehlberg (RKF) [46] and Cash-Karp (RKCK) [47].

Conceptually, the **Dormand-Prince** method solves the same system as RK4, Eq. (11). The idea behind the adaptive time step is to solve it for two consecutive orders allowing to estimate the total error in the last computed time step. The Butcher tableau [40] in Table 3 is used in our implementation. Traditionally the 5th order estimate—b_i values—is used and the 6th order—b_i^* values— allow to evaluate the error.

The equations are built exactly as for the RK4 method. To evaluate at time $n+1$ the size of the next time step given V_{n+1} and V_{n+1}^*, the method in [43] is used. The parameters are the tolerance $\varepsilon = 10^{-2}$, a *safety* value of 0.9 and the value Δ_0. Δ_0 consists of the ratio of the error given the previous value or $\frac{V_{n+1}-V_{n+1}^*}{V_n}$. If the step just taken is of size h then the next step size is:

$$h_{\text{next}}^V = \begin{cases} 0.9h\left|\frac{\Delta_0}{\varepsilon}\right|^{0.2} & \Delta_0 \geq \varepsilon \\ 0.9h\left|\frac{\Delta_0}{\varepsilon}\right|^{0.25} & \Delta_0 < \varepsilon. \end{cases} \tag{18}$$

In the same way a value h_{next}^T is computed. The smallest from both is considered as h_{next}. If any of those is in the first case, where the step size needs to shrink, we discard the present calculation and restart with the new and smaller step size.

The implicit **Rosenbrock** method is implemented as described in [43]. In its most general form the method seeks to solve the following equations given that $\alpha_{i,j}$, γ, $\gamma_{i,j}$, and q_i are well chosen constants. We can notice that a Runge–Kutta scheme is retrieved by setting $\gamma = \gamma_{i,j} = 0$. In this implementation, it is important to note that f is now a vector of two functions, $[\frac{\mathrm{d}T(t)}{\mathrm{d}t}, \frac{\mathrm{d}V(t)}{\mathrm{d}t}]$ and that the notation $f(y_n + x)$ is used to represent the array $[\frac{\mathrm{d}T(t)}{\mathrm{d}t}, \frac{\mathrm{d}V(t)}{\mathrm{d}t}]$ which

q_i	$a_{i,j}$						
0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	-56/15	32/9				
8/9	19372/6561	25360/2187	64448/6561	212/729			
1	9017/3168	355/33	46732/5247	49/176	5103/18656		
1	35/384	0	500/1113	125/192	2187/6784	11/84	
b_i	35/384	0	500/1113	125/192	2187/6784	11/84	0
b_i^*	5179/57600	0	7571/16695	393/640	92097/339200	187/2100	1/40

Table 3: Butcher tableau Dormand-Prince

is exactly:

$$f(y_n+x) = \Big[s - \beta V_n(T_n+x) - d(T_n+x),\\ (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a - c(V_n+x)\Big]. \tag{19}$$

Note that the part $(1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,\mathrm{d}a$ is constant at each iteration. It is computed separately and is not involved in the method.

A formal definition of the general Rosenbrock method of order s is that it consists of solving the implicit equations [43]:

$$y_{n+1} = y_n + \sum_{i=1}^{s} q_i k_i \text{ , such that}$$
$$(1-\gamma h f')k_i = hf\left(y_n + \sum_{j=1}^{i-1}\alpha_{i,j}k_j\right) + hf'\sum_{j=1}^{i-1}\gamma_{i,j}k_j, \qquad i=1,\dots,s, \tag{20}$$

were f' denotes the Jacobian matrix, which is:

$$f' = \begin{pmatrix} -d-\beta V & -\beta T \\ 0 & -c \end{pmatrix}. \tag{21}$$

A final substitution provides a form allowing a few optimizations. Given $g_i = \sum_{j=1}^{i-1}\gamma_{i,j}k_j + \gamma k_i$ the equations become:

$$\begin{aligned}
\left(\frac{1}{\gamma h} - f'\right) g_1 &= f(y_n)\\
\left(\frac{1}{\gamma h} - f'\right) g_2 &= f(y_n + a_{21}g_1)\\
&\quad + q_{21}g_1/h\\
\left(\frac{1}{\gamma h} - f'\right) g_3 &= f(y_n + a_{31}g_1 + a_{32}g_2)\\
&\quad + (q_{31}g_1 + q_{32}g_2)/h\\
\left(\frac{1}{\gamma h} - f'\right) g_4 &= f(y_n + a_{41}g_1 + a_{42}g_2 + a_{43}g_3)\\
&\quad + (q_{41}g_1 + q_{42}g_2 + q_{43}g_3)/h.
\end{aligned} \tag{22}$$

Finally, to better control the error it has been shown that we must add for each g_i the value $hq_{ix}\frac{\partial f}{\partial x}$ [43] on the right hand side where q_{ix} are predetermined constants. Note that since f' is a 2×2 matrix and f is a 2×1 vector, g_is are 2×1 vectors.

Once the four g_is are computed, we can directly evaluate the next values and the error using only 8 parameters, which we will call b_is and e_is. The next value is thus $y_{n+1} = y_n + b_1g_1 + b_2g_2 + b_3g_3 + b_4g_4$ and its associated error $e = e_1g_1 + e_2g_2 + e_3g_3 + e_4g_4$. All the required parameters are now presented and the values depicted in Table 4 are used.

The main advantage of this technique is the automatic step size adjustment, more so when the equations are stiff. As with Dormand-Prince, an error

$\gamma = 0.5$	$a_{21} = 2.0$	$a_{31} = 48/25$
$a_{32} = 6/25$	$q_{21} = -8$	$q_{31} = 372/25$
$q_{41} = -112/125$	$q_{42} = -54/125$	$q_{43} = -2/5$
$b_1 = 19/9$	$b_2 = 1/2$	$b_3 = 25/108$
$b_4 = 125/108$	$e_1 = 17/54$	$e_2 = 7/36$
$e_3 = 0$	$e_4 = 125/108$	$q_{1x} = 1/2$
$q_{2x} = -3/2$	$q_{3x} = 121/50$	$q_{4x} = 29/250$

Table 4: Rosenbrock method parameters

is computed using two sets of coefficients. The current implementation uses terms of order 4 and 5 to evaluate the error. The next step size is found using Eq. (18) with two differences. First, the exponents are 1/3 and 1/4 [43]. Second, the term Δ_0 contains the derivative, formally in that case $\Delta_0 = \frac{e}{y_n + hy'_n + 10^{-30}}$. As previously if a shrinkage of the step is observed we discard the present step and start again with the smaller value. By limiting a step size increase to at most 1% of the previous one the error induced by the integral appears to become negligible (data not shown).

2.5. A numerical scheme for direct parameter estimation from the multiscale model equations

As mentioned in Sec. 2.4.1, the structure of the equations is problematic when trying to use some general fitting methods as canned solvers (e.g., calling the Levenberg-Marquardt [43] as a canned method). In passing, we briefly show here how the multiscale model equations can be adapted for the fitting of parameter s when combined with a full implementation of the Levenberg–Marquardt method, without the use of canned solvers. Such a scheme can be generalized for other variables besides s and is left for future work that is beyond the scope of this contribution.

Briefly stated, the main idea for the development of a new parameter fitting scheme that should be more flexible than what was used in [18], which utilized Levenberg-Marquardt as a canned method, is how the the derivatives of V and T with respect to s can be evaluated. As schemes to solve $\frac{\mathrm{d}T(t)}{\mathrm{d}t}$ and $\frac{\mathrm{d}V(t)}{\mathrm{d}t}$ being already implemented, we can notice that $\frac{\mathrm{d}\frac{\partial V(t)}{\partial s}}{\mathrm{d}t}$ and $\frac{\mathrm{d}\frac{\partial T(t)}{\partial s}}{\mathrm{d}t}$ can be determined and thereafter solved with the already implemented schemes. Those values are exactly:

$$\frac{\mathrm{d}\frac{\partial T(t)}{\partial s}}{\mathrm{d}t} = 1 - \beta\left(\frac{\partial V(t)}{\partial s}T(t) + V(t)\frac{\partial T(t)}{\partial s}\right) - d\frac{\partial T(t)}{\partial s} \tag{23a}$$

$$\frac{\mathrm{d}\frac{\partial V(t)}{\partial s}}{\mathrm{d}t} = (1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)\frac{\partial I(a,t)}{\partial s}\,\mathrm{d}a - c\frac{\partial V(t)}{\partial s} \tag{23b}$$

where:

$$\frac{\partial I(a,t)}{\partial s} = \begin{cases} \beta\left(\frac{\partial V(t-a)}{\partial s}T(t-a) \right. & \\ \left. \qquad +V(t-a)\frac{\partial T(t-a)}{\partial s}\right)\mathrm{e}^{-\delta a} & a < t \cdot \\ \mathrm{e}^{-\delta a} & a \geq t \end{cases} \tag{24}$$

Let us note that the partial derivative of $R(a,t)$ is 0 and thus ignored.

These equations can be directly integrated in the previously discussed schemes since they are working on a set of ODEs in parallel. Practically, we need to increase the dimensions of the vector f, defined in subsection 2.4.4, to include those new equations. We now need to solve for $f(y_n + x) := \left[\frac{\mathrm{d}V(t)}{\mathrm{d}t}, \frac{\mathrm{d}T(t)}{\mathrm{d}t}, \frac{\mathrm{d}\frac{\partial V(t)}{\partial s}}{\mathrm{d}t}, \frac{\mathrm{d}\frac{\partial T(t)}{\partial s}}{\mathrm{d}t}\right]$. The only additional variable to be inserted is a tolerance on the error of the fitting. In section 3.6, we report on a successful proof of concept for the new parameter fitting scheme that was briefly outlined here and will be fully dealt with in future work.

3. Results

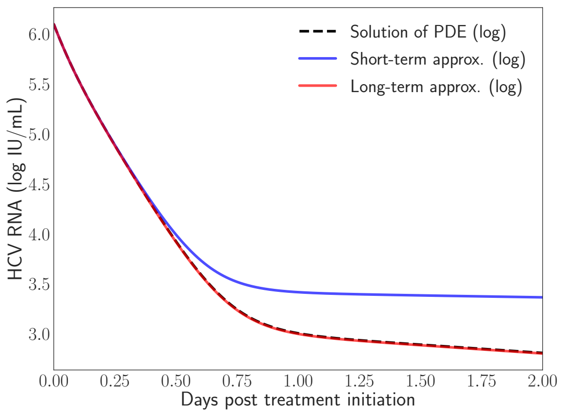
All the methods are implemented in Python3 using the SciPy library. The implementations are freely available at `http://www.cs.bgu.ac.il/~dbarash/HCVnumerics`. The default implementation in SciPy of an ordinary differential equation solver leverages ODEPACK [48] and is referred to as the canned solver *Default*. For this entire section, we used the parameters from [18] and shown in Table. 5. The parameters that are changed through the results are the number of days, the size h of the steps for the ODEs, and the size h_a of the steps for computing the integral. In the case of adaptive methods we bound the stepsize by h as minimum and h_a as maximum.

3.1. Limitation of the short-term approximation

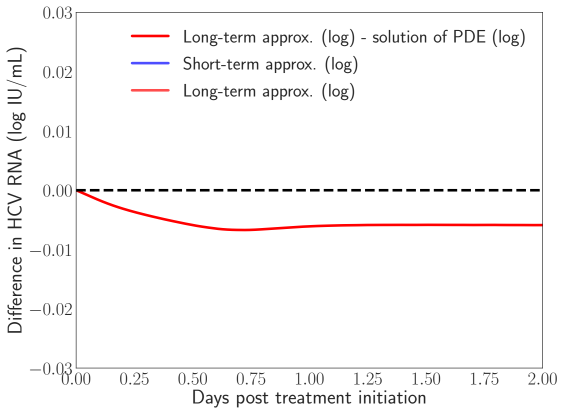
The short-term approximation is shown in Fig. 1a in blue. The results are shown for only two days since the behaviour is smooth and conserved on longer time scales in agreement with previous results (Fig. 2A in [18]). It is clear that after twelve hours the value converges far from the PDE model solution (the PDE model solution is computed by the canned solver *Default* implementation as described at the beginning of the section). This is expected since the effect of the treatment on the infection rate is not taken into account. The analysis will therefore focus on the long-term approximation.

3.2. Limitation of the long-term approximation

In [36] it is hypothesized that "*The long-term approximation is an **underestimate** of the PDE model solution since some infection events are being ignored. However, with realistic parameters characteristic of potent therapy, the difference between them is very small.*" Fig. 1b depicts the result of our scheme, measuring the difference between the long-term approximation and the solution of the multiscale PDE model (Figure 2B in [18]). We show that the long-term approximation is converging to a number with the numerical solution and we are indeed obtaining that the long-term approximation is an underestimate of the PDE model solution (as the difference between the long term approximation and the solution of the PDE model is negative). The difference of less than 0.01 that we are obtaining is very small compared to the y-axis units shown in Figure 1 of [36]. Therefore we are actually viewing in Fig. 1b this small effect that was inferred in [36] regarding the long-term approximation. This limitation of the long-term approximation is worthwhile mentioning although it is less severe than the limitation of the short-term approximation. Therefore in practice, the long-term approximation is favored over the short-term approximation, although the numerical solution offers an attractive alternative to analytical approximations and could be easier to adjust when introducing changes to the model.



(a) The log values of the long-term and short-term approximation are shown compared to the PDE solution using the canned method (*Default*). The convergence close to the long-term approximation can be observed.



(b) We observe the underestimation of the PDE equations by the long-term approximation.

Figure 1: The parameters are $h = 0.001$, $h_a = 0.01$, and over two days.

Parameter	Value	Parameter	Value
s	130 000 cells/mL	β	0.000 000 05 mL day $^{-1}$ virion $^{-1}$
d	0.01 day $^{-1}$	δ	0.14 day $^{-1}$
κ	6.36	c	22.5 day $^{-1}$
α	40 day $^{-1}$	ρ	7.95 day $^{-1}$
μ	1 day $^{-1}$	γ	0.24 day $^{-1}$
ε_s	0.65	ε_α	0.997

Table 5: The parameters of the model used in all simulations, taken from [18].

3.3. The multiscale model equations are stiff

A significant observation that can be inferred already when starting to experiment with the multiscale model equations by numerical solutions is their stiffness, or how they can be numerically unstable even with a small stepsize. The most straightforward numerical method to implement is forward Euler, described in Sec. 2.4.2, followed by RK4 discussed in Sec. 2.4.3. An implicit version of RK4 in the same section is the Gauss-Legendre method. With small time-steps, the quality of the results is similar between all methods. But as we increase the size of the steps, the instability of the equations can be observed as show in Fig. 2 over 20 days. Both forward Euler and RK4 quickly diverge from the long-term approximation, which is close to the numerical solution as shown previously. On the other hand the implicit method of second order Gauss-Legendre that belongs to the Runge-Kutta family tends to the correct solution.

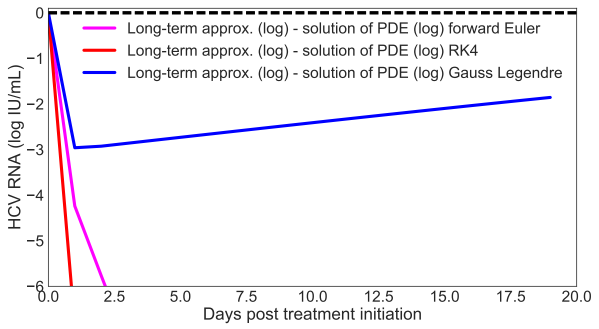


Figure 2: Under the parameter values $ht = 1$, $ha = 1$, and over 20 days, the model equations are shown to be stiff. Both forward Euler and RK4 diverge quickly. Gauss-Legendre (implicit method) is correcting itself and moving towards the desired solution that utilizes small stepsize.

3.4. The problems encountered when using a canned solver for the numerical equations

Without taking a detailed examiniation of the equations, a straight-forward strategy that is simple to implement and pursue is the utilization of a canned solver (an ODE solver called from by a command within a program that is written in a higher level language such as Matlab and Mathematica, or Python). Those canned solvers solve the equations automatically, determining the time steps given a tolerance provided by the user (only initial and extremal values for the time step can be given). However, a restriction of such a strategy is that the equations need to be self contained in a set of ODEs without additional integrals that may complicate the matter, as discussed in Sec. 2.4.1. The integral term $(1-\epsilon_s)\int_0^\infty \rho(a)R(a,t)I(a,t)\,da$ cannot be solved analytically and thus a numerical approach must be used. Doing so requires to know for which times t the values of V and T are known, and what they are. Such a scheme is incompatible with the canned solver strategy. A workaround to not fully implement a solver within a program is therefore to use a canned solver for small increments of times h and after each iteration, compute the integral again. We illustrate this approach in Fig. 3 and it is the strategy used to evaluate the system with what we call a *Default* method. This is not sustainable for long time intervals as the number of iterations grows quickly and defeats the purpose of the implementation of advanced ODE solvers that utilize multistep methods or adaptive timestep methods. In this strategy, there is no advantage in using in the canned solver a sophisticated multistep method such as the Backward Differentiation Formula (BDF) or the Rosenbrock method that are especially suited for stiff equations because of the small time increments that do not provide a chance to exploit the advanced method used in the canned solver.

In contrast, a full implementation of the Rosenbrock method gives full access to the error term. With that information, a more subtle integration of the problematic integral is performed as shown in Fig. 4. There are two main differences. First, we need to keep track of the times at which the ODE system is solved in the array $\mathcal{I}$, which is required to compute the integral since those are not fixed anymore. Second, after each iteration we use the known error to modify if possible the time step h to potentially decrease the number of required iterations. This is shown to be a successful strategy as presented in Table 6.

3.5. Advantages of implicit adaptive stepsize methods

As previously explained, the integral term $(1 - \epsilon_s) \int_0^\infty \rho(a) R(a,t) I(a,t)\, \mathrm{d}a$ impedes the use of adaptive stepsize in the canned solvers. Such methods are essential to allow fast computations of complex models like ours. We present in Table. 6 the number of iterations computed using the Rosenbrock and Dormand-Prince methods, shown in Sec. 2.4.4, in comparison with the default canned method for three different values of h and h_a. With the smallest stepsize, the Rosenbrock method is more than nine times as efficient as the default canned method, giving results of similar accuracy as shown in Fig. 5 and with simulations extended to 14 days. This is crucial since the methods with fixed stepsize can take up to tens of seconds per day of simulation with $h = 0.001$ and $h_a = 0.01$. Under those conditions a nine-fold increase in speed, through the decrease in the number of iterations, provides a much needed advantage when testing large number of parameters over long time periods. This is the main inference from Table. 6, showing the advantage of adaptive stepsize methods. Interestingly, with the highest value of h, the Rosenbrock method takes proportionately additional steps. This is due to the greediness of the stepsize adjustment, which when increased too quickly (thereby inducing a large error) backtracks and starts again with a smaller value. The Dormand-Prince adaptive stepsize method performs somewhat better than the Rosenbrock method in terms of number of iterations but is vulnerable to the stiffness of the multiscale model equations, as elaborated below.

Simpler methods than Rosenbrock exist with adaptive stepsize. A well used scheme is the Dormand-Prince method that was described in Sec. 2.4.4. The latter is an explicit method, while Rosenbrock is implicit, and therefore Dormand-Prince is vulnerable to the stiffness of the equations. We show in Fig. 5 that with a modest increase in the stepsize h the Dormand-Prince method already converges to the

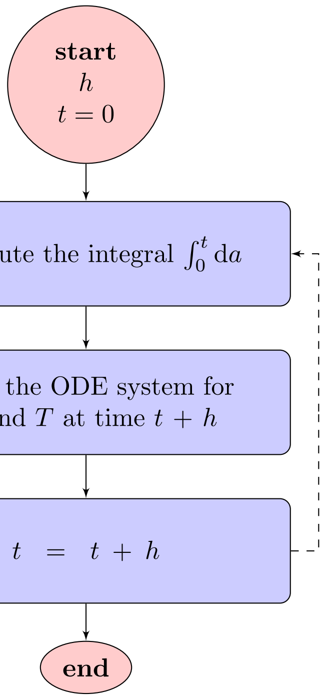

start
h
$t = 0$
Compute the integral $\int_0^t \mathrm{d}a$
Solve the ODE system for V and T at time $t + h$
$t = t + h$
end

Figure 3: **Canned method flow** When using the canned solver, the production of free virus depends on all preceding time steps. The canned solver to solve the ODE must therefore be called with a small time step h and use that value to reevaluate the integral before the next iteration. This strategy does not allow to exploit the benefits of the specialized method that is used within the canned solver.

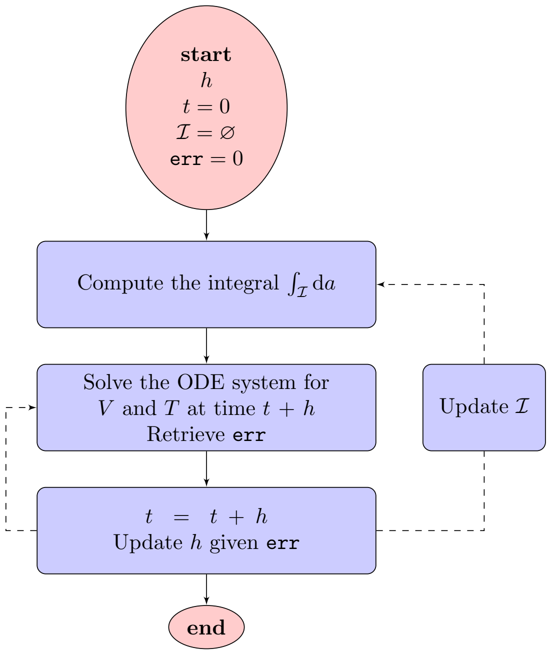

start
h
$t = 0$
$\mathcal{I} = \varnothing$
$\texttt{err} = 0$
Compute the integral $\int_{\mathcal{I}} \mathrm{d}a$
Solve the ODE system for V and T at time $t + h$
Retrieve err
Update $\mathcal{I}$
$t = t + h$
Update h given err
end

Figure 4: **Rosenbrock implementation flow** The full implementation of the method gives us the error, `err`, at each time step. Doing so allows to vary the size of the step h. We keep track of the time steps necessary to compute the integral in an auxiliary array $\mathcal{I}$ (initialized at the beginning to an empty set) to take full advantage of the adaptive step sizes.

h	h_a	Ros.	Do.–Pr.	*Default*
0.001	0.01	1543	1403	14 000
0.005	0.05	582	295	2800
0.01	0.1	563	290	1400

Table 6: Number of steps taken by the Dormand–Prince (Do.–Pr.) and Rosenbrock (Ros.) methods given h and h_a compared to the number of steps taken with the canned method (*Default*). The results are for a simulation run of 14 days.

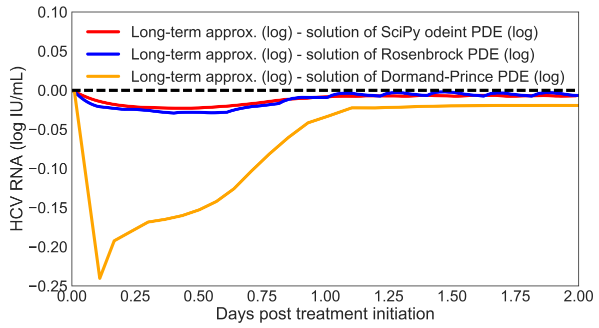


Figure 5: Adaptive time step stability with parameters $ht = 0.01$, $ha = 0.1$, and over two days. Rosenbrock and the canned solver (*Default*) converge to the same solution while Dormand-Prince does not converge to the right solution.

wrong value compared to the default method and the Rosenbrock method. We note that Rosenbrock needs almost a third of the number of iterations as the *Default* method in that case with similar results.

3.6. Parameter estimation example performed directly from the multiscale model equations

In sub-section 2.5, we briefly sketched an idea for a new strategy that performs flexible parameter fitting and is in line with our overall strategy to tackle the problematic issues with the integral mentioned in sub-section 2.4.1. As a proof of concept, to evaluate our method, we generated an ensemble of data points over one day with our implementation of the Rosenbrock method and the parameters in Table 5.

We then applied our implementation of the Levenberg–Marquardt method with different starting points for the parameter s, which can not be recovered when using the long-term approximation. Using 20 values of s equally distributed in $[0, 2s]$ we observed that in all cases the method converged to the correct value of s used to generate the data points, inside the provided tolerance on the error of 0.001.

4. Conclusions

A viral dynamic model that considers intracellular viral RNA replication, namely an age-structured PDE multiscale model, has been recently put forth to study viral hepatitis dynamics during antiviral therapy [18, 36, 4]. This type of model is more complicated to solve than previous models. The seminal works that introduce the age-based multiscale model are predominant by analytical approximations, with some numerical solutions that are either based on simple first-order methods or canned solvers (ODE solvers used in higher level languages such as Matlab and Mathematica, or Python). Neither of these numerical solutions are satisfactory for realistic simulations of several days of infection and therapy in terms of accuracy, efficiency and stability.

Analytical approximations to the multiscale model have limitations. The short-term approximation holds for only half a day and the long-term approximation is an underestimate of the PDE model solution, which was anticipated in [36] because some infection events are being ignored with this analytical approximation. By experimenting with numerical solutions, one can observe that the governing differential equations are stiff and therefore advanced numerical methods are needed. First-order methods with fixed stepsize or alternatively, the use of canned solvers, do not offer a comprehensive solution treatment to the model and are limited in scope. For example, in the multiscale model, there is an integral term that needs to be computed at each time step depending on previous iterations, which is inadequately done by canned solvers. Higher-order methods with adaptive stepsize are offering a considerable improvement in terms of both accuracy and efficiency. An implicit higher-order method with adaptive stepsize was found to be the most stable, in addition to being the most accurate and efficient among the various methods being compared that are adequate for this model. As a byproduct, the overall strategy of using fully implemented numerical schemes applied directly on the multiscale model equations allows parameter estimation of the full multiscale model.

Although it is expected that the Rosenbrock mehod remains competitive for this application, other methods that might be advantageous in some ways could still be tried. Among them is the Backward Differentiation Formula (BDF), which is an implicit method belonging to the class of linear multistep methods that have been used for the solution of stiff differential equations, or other variants of Diagonally Implicit Runge-Kutta (DIRK) that are described in [49]. In these other DIRK variants, one nonlinear system has to be solved per step, whereas Rosenbrock schemes are derived by linearizing a DIRK method, resulting in schemes where only a linear system has to be solved per step. Therefore the cost of time should be lower, but larger time integration errors in the Rosenbrock schemes might arise, leaving the possibility that some of these other DIRK variants might also be competitive in their accuracy. Nevertheless, the Rosenbrock method is expected to remain attractive for solving the multiscale model numerically because of its efficiency.

5. Acknowledgments

The research was supported in part by the U.S. National Institute of Health (NIH) grant R01-AI078881. VR is funded by Azrieli and FQRNT post-doctoral fellowships. The authors would like to thank Libin Rong and Daniel Coffield for providing the initial code of the multiscale model.

6. References

[1] World Health Organization, Guidelines for the screening, care and treatment of persons with hepatitis C infection., World Health Organization, 2014.

[2] AASLD/IDSA HCV Guidance Panel, Hepatitis C guidance: AASLD-IDSA recommendations for testing, managing, and treating adults infected with hepatitis C virus, Hepatology 62 (3) (2015) 932–954.

[3] H. R. Rosen, “hep C, where art thou”: What are the remaining (fundable) questions in hepatitis C virus research?, Hepatology 65 (1) (2017) 341–349.

[4] J. Guedj, H. Dahari, L. Rong, N. D. Sansone, R. E. Nettles, S. J. Cotler, T. J. Layden, S. L. Uprichard, A. S. Perelson, Modeling shows that the NS5A inhibitor daclatasvir has two modes of action and yields a shorter estimate of the hepatitis C virus half-life, Proc Natl Acad Sci USA 110 (2013) 3991–3996.

[5] H. Dahari, B. Sainz, A. S. Perelson, S. L. Uprichard, Modeling subgenomic hepatitis C virus RNA kinetics during treatment with alpha interferon, Journal of virology 83 (13) (2009) 6383–6390.

[6] H. Dahari, R. M. Ribeiro, C. M. Rice, A. S. Perelson, Mathematical modeling of subgenomic hepatitis C virus replication in Huh-7 cells, Journal of virology 81 (2) (2007) 750–760.

[7] A. U. Neumann, S. Phillips, I. Levine, S. Ijaz, H. Dahari, R. Eren, S. Dagan, N. V. Naoumov, Novel mechanism of antibodies to hepatitis B virus in blocking viral particle release from cells, Hepatology 52 (3) (2010) 875–885.

[8] A. S. Perelson, Modelling viral and immune system dynamics, Nature Reviews Immunology 2 (1) (2002) 28–36.

[9] D. Burg, L. Rong, A. U. Neumann, H. Dahari, Mathematical modeling of viral kinetics under immune control during primary HIV-1 infection, Journal of Theoretical Biology 259 (4) (2009) 751–759.

[10] H. Dahari, E. Shudo, R. M. Ribeiro, A. S. Perelson, Mathematical modeling of HCV infection and treatment, Hepatitis C: Methods and Protocols (2009) 439–453.

[11] H. Dahari, J. Guedj, A. S. Perelson, T. J. Layden, Hepatitis C viral kinetics in the era of direct acting antiviral agents and interleukin-28B, Current hepatitis reports 10 (3) (2011) 214–227.

[12] E. Snoeck, P. Chanu, M. Lavielle, P. Jacqmin, E. N. Jonsson, K. Jorga, T. Goggin, J. Grippo, N. L. Jumbe, N. Frey, A comprehensive hepatitis C viral kinetic model explaining cure, Clinical Pharmacology & Therapeutics 87 (6) (2010) 706–713.

[13] N. M. Dixit, J. E. Layden-Almer, T. J. Layden, A. S. Perelson, Modelling how ribavirin improves interferon response rates in hepatitis C virus infection, Nature 432 (7019) (2004) 922–924.

[14] J. Guedj, A. S. Perelson, Second-phase hepatitis C virus RNA decline during telaprevir-based therapy increases with drug effectiveness: Implications for treatment duration, Hepatology 53 (6) (2011) 1801–1808.

[15] H. Dahari, S. Shteingart, I. Gafanovich, S. J. Cotler, M. D'Amato, R. T. Pohl, G. Weiss, Y. J. Ashkenazi, T. Tichler, E. Goldin, et al., Sustained virological response with intravenous silibinin: individualized IFN-free therapy via real-time modelling of HCV kinetics, Liver International 35 (2) (2015) 289–294.

[16] L. Rong, H. Dahari, R. M. Ribeiro, A. S. Perelson, Rapid emergence of protease inhibitor resistance in hepatitis C virus, Science translational medicine 2 (30) (2010) 30ra32.

[17] J. Guedj, H. Dahari, S. L. Uprichard, A. S. Perelson, The hepatitis C virus NS5A inhibitor daclatasvir has a dual mode of action and leads to a new virus half-life estimate, Expert review of gastroenterology & hepatology 7 (5) (2013) 397–399.

[18] L. Rong, J. Guedj, H. Dahari, D. J. J. Coffield, M. Levi, P. Smith, A. S. Perelson, Analysis of hepatitis C virus decline during treatment with the protease inhibitor danoprevir using a multiscale model, PLOS Comput. Biol. 9 (2013) e1002959.

[19] D. D. Ho, A. U. Neumann, A. S. Perelson, W. Chen, et al., Rapid turnover of plasma virions and CD4 lymphocytes in HIV-1 infection, Nature 373 (6510) (1995) 123.

[20] A. S. Perelson, A. U. Neumann, M. Markowitz, J. M. Leonard, D. D. Ho, HIV-1 dynamics in vivo: virion clearance rate, infected cell life-span, and viral generation time, Science 271 (5255) (1996) 1582.

[21] S. M. Ciupe, R. M. Ribeiro, P. W. Nelson, G. Dusheiko, A. S. Perelson, The role of cells refractory to productive infection in acute hepatitis B viral dynamics, Proceedings of the National Academy of Sciences 104 (12) (2007) 5050–5055.

[22] H. Dahari, E. Shudo, R. M. Ribeiro, A. S. Perelson, Modeling complex decay profiles of hepatitis B virus during antiviral therapy, Hepatology 49 (1) (2009) 32–38.

[23] M. A. Nowak, S. Bonhoeffer, A. M. Hill, R. Boehme, H. C. Thomas, H. McDade, Viral dynamics in hepatitis B virus infection, Proceedings of the National Academy of Sciences 93 (9) (1996) 4398–4402.

[24] C. Koh, L. Canini, H. Dahari, X. Zhao, S. L. Uprichard, V. Haynes-Williams, M. A. Winters, G. Subramanya, S. L. Cooper, P. Pinto, et al., Oral prenylation inhibition with lonafarnib in chronic hepatitis D infection: a proof-of-concept randomised, double-blind, placebo-controlled phase 2A trial, The Lancet Infectious Diseases 15 (10) (2015) 1167–1174.

[25] J. Guedj, Y. Rotman, S. J. Cotler, C. Koh, P. Schmid, J. Albrecht, V. Haynes-Williams, T. J. Liang, J. H. Hoofnagle, T. Heller, et al., Understanding early serum hepatitis D virus and hepatitis B surface antigen kinetics during pegylated interferon-alpha therapy via mathematical modeling, Hepatology 60 (6) (2014) 1902–1910.

[26] J. Zhang, H. L. Lipton, A. S. Perelson, H. Dahari, Modeling the acute and chronic phases of Theiler murine encephalomyelitis virus infection, Journal of virology 87 (7) (2013) 4052–4059.

[27] J. T. Schiffer, L. Abu-Raddad, K. E. Mark, J. Zhu, S. Selke, A. Magaret, A. Wald, L. Corey, Frequent release of low amounts of herpes simplex virus from neurons: results of a mathematical model, Science translational medicine 1 (7) (2009) 7ra16.

[28] H. Dahari, J. E. Layden-Almer, E. Kallwitz, R. M. Ribeiro, S. J. Cotler, T. J. Layden, A. S. Perelson, A mathematical model of hepatitis C virus dynamics in patients with high baseline viral loads or advanced liver disease, Gastroenterology 136 (4) (2009) 1402–1409.

[29] H. Dahari, M. Major, X. Zhang, K. Mihalik, C. M. Rice, A. S. Perelson, S. M. Feinstone, A. U. Neumann, Mathematical modeling of primary hepatitis C infection: noncytolytic clearance and early blockage of virion production, Gastroenterology 128 (4) (2005) 1056–1066.

[30] A. U. Neumann, N. P. Lam, H. Dahari, D. R. Gretch, T. E. Wiley, T. J. Layden, A. S. Perelson, Hepatitis C viral dynamics in vivo and the antiviral efficacy of interferon-α therapy, Science 282 (1998) 103–107.

[31] P. Baccam, C. Beauchemin, C. A. Macken, F. G. Hayden, A. S. Perelson, Kinetics of influenza A virus infection in humans, Journal of virology 80 (15) (2006) 7590–7599.

[32] K. A. Pawelek, G. T. Huynh, M. Quinlivan, A. Cullinane, L. Rong, A. S. Perelson, Modeling within-host dynamics of influenza virus infection including immune responses, PLoS Comput Biol 8 (6) (2012) e1002588.

[33] C. A. Beauchemin, A. Handel, A review of mathematical models of influenza A infections within a host or cell culture: lessons learned and challenges ahead, BMC public health 11 (1) (2011) S7.

[34] V. Madelain, L. Oestereich, F. Graw, T. H. T. Nguyen, X. De Lamballerie, F. Mentré, S. Günther, J. Guedj, Ebola virus dynamics in mice treated with favipiravir, Antiviral research 123 (2015) 70–77.

[35] J. Guedj, A. U. Neumann, Understanding hepatitis C viral dynamics with direct-acting antiviral agents due to the interplay between intracellular replication and cellular infection dynamics, J. Theor. Biol. 267 (2010) 330–340.

[36] L. Rong, A. S. Perelson, Mathematical analysis of multiscale models for hepatitis C virus dynamics under therapy with direct-acting antiviral agents, Math Biosci. 245 (2013) 22–30.

[37] J. Weickert, B. ter Haar Romeny, M. Viergever, Efficient and reliable schemes for nonlinear diffusion filtering, IEEE Trans. Imag. Proc. 7 (1998) 398–410.

[38] D. Barash, M. Israeli, R. Kimmel, An accurate operator splitting scheme for nonlinear diffusion

filtering, in: Proceedings of the 3rd International Conference on ScaleSpace and Morphology, LNCS Series, Springer-Verlag, 2001, pp. 281–289.

[39] D. Barash, Nonlinear diffusion filtering on extended neighborhood, Appl. Num. Math. 52 (2005) 1–11.

[40] J. Dormand, P. Prince, A family of embedded Runge-Kutta formulae, J. Comp. Appl. Math. 6 (1980) 19–26.

[41] H. Rosenbrock, Some general implicit processes for the numerical solution of differential equations, Comp. J. 5 (1963) 329–330.

[42] V. Reinharz, A. Churkin, H. Dahari, D. Barash, A robust and efficient numerical method for RNA-mediated viral dynamics, Front. Appl. Math. Stat. 3 (2017) 20.

[43] W. H. Press, S. H. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in C, 2nd Edition, Cambridge University Press, 1997.

[44] A. Iserles, A First Course in the Numerical Analysis of Differential Equations, 2nd Edition, Cambridge University Press, 2008.

[45] S. P. Nørsett, A. Wolfbrandt, Order conditions for Rosenbrock type methods, Numerische Mathematik 32 (1) (1979) 1–15.

[46] E. Fehlberg, Low-order classical Runge-Kutta formulas with step size control and their application to some heat transfer problems, Tech. rep., NASA Technical Report 315 (1969).

[47] J. Cash, A. Karp, A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides, ACM Trans. Math. Software 16 (1990) 201–222.

[48] A. C. Hindmarsh, ODEPACK, a systematized collection of ODE solvers, IMACS Trans. Sci. Comp. 1 (1983) 55–64.

[49] H. Bijl, P. Birken, A. Meister, A. van Zuijlen, A comparison of the efficiency of Rosenbrock and DIRK variants, Proc. Appl. Math. Mech. 12 (2012) 675–676.

Appendices

Appendix A. Method of characteristics

Appendix A.1. $\mathbf{R(a,t)}$

We have the relation

$$\frac{\partial R(a,t)}{\partial t} + \frac{\partial R(a,t)}{\partial a} = \alpha(a) - [\rho(a) + \mu(a)]R(a,t). \tag{25}$$

The method of characteristics consists of finding a parametrization of a and t in τ such that the coefficient of their respective derivatives allows to combine them as such:

$$\frac{\mathrm{d}R(a(\tau),t(\tau))}{\mathrm{d}\tau} = \frac{\partial R}{\partial t}\frac{\partial t}{\partial \tau} + \frac{\partial R}{\partial a}\frac{\partial a}{\partial \tau} = \alpha(a(\tau)) - [\rho(a(\tau)) + \mu(a(\tau))]R(a(\tau),t(\tau)).$$

In this case, they are at $a - t = k$, for k a constant.

Appendix A.2. $\mathbf{a} \geq t$

The parametrization in τ is such that $a(\tau) : a_0 \xrightarrow{\tau:0\to t} a$ and $t(\tau) : 0 \xrightarrow{\tau:0\to t} t$

$$t = \tau \Rightarrow \tfrac{\mathrm{d}t}{\mathrm{d}\tau} = 1 \tag{26}$$
$$a = \tau + a_0 \Rightarrow \tfrac{\mathrm{d}a}{\mathrm{d}\tau} = 1. \tag{27}$$

Therefore the equations (25) and (27) are consistent together. We now have to solve the `ODE`:

$$R' + [\rho(a(\tau)) + \mu(a(\tau))]R(a(\tau),t(\tau)) = \alpha(a(\tau)).$$

Using the methods of constants Appendix B we need to solve

$$\begin{aligned} K' &= \alpha(a(\tau))\mathrm{e}^{\int[\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta} \\ &\Downarrow \\ K &= \int_0^t \alpha(a(u))\mathrm{e}^{\int_0^u[\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta}\,\mathrm{d}u + C, \end{aligned}$$

which implies that:

$$\begin{aligned} R(a(\tau),t(\tau)) &= \left(\int_0^t \alpha(a(u))\mathrm{e}^{\int_0^u[\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta}\,\mathrm{d}u + C\right)\mathrm{e}^{-\int_0^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} \\ &= C\mathrm{e}^{-\int_0^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} + \mathrm{e}^{-\int_0^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau}\int_0^t \alpha(a(u))\mathrm{e}^{\int_0^u[\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta}\,\mathrm{d}u. \end{aligned}$$

In particular, when $\tau = 0$ we have $R(a(0),t(0)) = R(a_0,0) = C$ and therefore:

$$R(a(\tau),t(\tau)) = R(a_0,0)\mathrm{e}^{-\int_0^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} + \mathrm{e}^{-\int_0^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau}\int_0^t \alpha(a(u))\mathrm{e}^{\int_0^u[\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta}\,\mathrm{d}u.$$

Given the steady state distribution $\bar{R}(a)$, we have as assumption the parametrization (27) and that $R(a,0) = \bar{R}(a) \Rightarrow R(a_0,0) = R(a-t,0) = \bar{R}(a-t)$ so $R(a,t), a \geq t$ is :

$$R(a,t) = \bar{R}(a-t)\mathrm{e}^{-\int_0^t [\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} + \mathrm{e}^{-\int_0^t [\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} \int_0^t \alpha(a(u))\mathrm{e}^{\int_0^u [\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta}\,\mathrm{d}u.$$

We will finish by re-writting those equations using the parametrization (27).
First notice that:

$$\int_0^t [\rho(a(\tau)) + \mu(a(\tau))]\,\mathrm{d}\tau = \int_{a_0}^a [\rho(\zeta) + \mu(\zeta)]\,\mathrm{d}\zeta$$

and

$$\int_{a_0}^a [\rho(\zeta) + \mu(\zeta)]\,\mathrm{d}\zeta = \int_0^a [\rho(\zeta) + \mu(\zeta)]\,\mathrm{d}\zeta - \int_0^{a_0} [\rho(\zeta) + \mu(\zeta)]\,\mathrm{d}\zeta,$$

therefore

$$\mathrm{e}^{-\int_0^t [\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} = \frac{\mathrm{e}^{-\int_0^a [\rho(\zeta)+\mu(\zeta)]\,\mathrm{d}\zeta}}{\mathrm{e}^{-\int_0^{a_0} [\rho(\zeta)+\mu(\zeta)]\,\mathrm{d}\zeta}}. \tag{28}$$

Let be $\pi(s) = \mathrm{e}^{-\int_0^s [\rho(\zeta)+\mu(\zeta)]\,\mathrm{d}\zeta}$ then $\mathrm{e}^{-\int_0^t [\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} = \frac{\pi(a)}{\pi(a_0)} = \frac{\pi(a)}{\pi(a-t)}$

$$R(a,t) = \bar{R}(a-t)\frac{\pi(a)}{\pi(a-t)} + \frac{\pi(a)}{\pi(a-t)} \int_0^t \alpha(a(u))\mathrm{e}^{\int_0^u [\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta}\,\mathrm{d}u.$$

We can use once more the result of Eq. (28) to arrive at:

$$\begin{aligned}
R(a,t) &= \bar{R}(a-t)\frac{\pi(a)}{\pi(a-t)} + \frac{\pi(a)}{\pi(a-t)} \int_0^t \alpha(a(u))\frac{\pi(a_0)}{\pi(a-u)}\,\mathrm{d}u \\
R(a,t) &= \bar{R}(a-t)\frac{\pi(a)}{\pi(a-t)} + \frac{\pi(a)}{\pi(a-t)} \int_0^t \alpha(a(u))\frac{\pi(a-t)}{\pi(a-u)}\,\mathrm{d}u \\
R(a,t) &= \bar{R}(a-t)\frac{\pi(a)}{\pi(a-t)} + \int_0^t \frac{\pi(a)\alpha(a(u))}{\pi(a-u)}\,\mathrm{d}u.
\end{aligned}$$

Assuming that α, ρ and μ are constant we can solve the integral:

$$\begin{aligned}
\int_0^t \alpha\frac{\pi(a)}{\pi(a-u)}\,\mathrm{d}u &= \int_0^t \alpha\frac{\mathrm{e}^{-\int_0^a (\rho+\mu)\,\mathrm{d}v}}{\mathrm{e}^{-\int_0^{a-u} (\rho+\mu)\,\mathrm{d}v}}\,\mathrm{d}u \\
&= \alpha\int_0^t \frac{\mathrm{e}^{-(\rho+\mu)a}}{\mathrm{e}^{-(\rho+\mu)(a-u)}}\,\mathrm{d}u \\
&= \alpha\int_0^t \mathrm{e}^{-(\rho+\mu)a+(\rho+\mu)(a-u)}\,\mathrm{d}u \\
&= \alpha\int_0^t \mathrm{e}^{-(\rho+\mu)u}\,\mathrm{d}u \\
&= \alpha\left(\frac{1}{-(\rho+\mu)}\mathrm{e}^{-(\rho+\mu)t} - \frac{1}{-(\rho+\mu)}\right) \\
&= \frac{\alpha}{(\rho+\mu)}(1-\mathrm{e}^{-(\rho+\mu)t}).
\end{aligned}$$

We thus obtain:

$$R(a,t) = \bar{R}(a-t)\frac{\pi(a)}{\pi(a-t)} + \frac{\alpha}{(\rho+\mu)}(1-\mathrm{e}^{-(\rho+\mu)t}).$$

Let us re-write

$$\begin{aligned}\frac{\pi(a)}{\pi(a-t)} &= \frac{\mathrm{e}^{-\int_0^a(\rho+\mu)\,\mathrm{d}v}}{\mathrm{e}^{-\int_0^{a-t}(\rho+\mu)\,\mathrm{d}v}}\\ &= \frac{\mathrm{e}^{-(\rho+\mu)a}}{\mathrm{e}^{-(\rho+\mu)(a-t)}}\\ &= \mathrm{e}^{-(\rho+\mu)t},\end{aligned}$$

then

$$\begin{aligned}R(a,t) &= \bar{R}(a-t)\frac{\pi(a)}{\pi(a-t)} + \frac{\alpha}{(\rho+\mu)}(1-\mathrm{e}^{-(\rho+\mu)t})\\ &= \bar{R}(a-t)\mathrm{e}^{-(\rho+\mu)t} + \frac{\alpha}{(\rho+\mu)} - \frac{\alpha}{(\rho+\mu)}\mathrm{e}^{-(\rho+\mu)t}\\ &= \frac{\alpha}{(\rho+\mu)} + \left(\bar{R}(a-t) - \frac{\alpha}{(\rho+\mu)}\right)\mathrm{e}^{-(\rho+\mu)t}.\end{aligned} \tag{29}$$

Appendix A.3. $\mathbf{a} < t$

In this case, we have $a < t$, then let us parametrize in function of τ such that $a(\tau) : 0 \xrightarrow{\tau:t_0\to t} a$ and $t(\tau) : t_0 \xrightarrow{\tau:t_0\to t} t$

$$t = \tau \Rightarrow \tfrac{\mathrm{d}t}{\mathrm{d}\tau} = 1 \tag{30}$$

$$a = \tau - t_0 \Rightarrow \tfrac{\mathrm{d}a}{\mathrm{d}\tau} = 1 \tag{31}$$

The change in conditions is first important in the general equation, where we can notice a change of bounds to the integrals:

$$R(a(\tau),t(\tau)) = C\mathrm{e}^{-\int_{t_0}^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} + \mathrm{e}^{-\int_{t_0}^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau}\int_{t_0}^t \alpha(a(u))\mathrm{e}^{\int_{t_0}^u[\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta}\,\mathrm{d}u$$

At the boundary condition $R(a(0),t(0)) = R(0,t_0) = 1$ (by assumption) and as all the integrals go to zero we have $C = 1$. Let us now notice that the $\int_{t_0}^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau = \int_0^a[\rho(\zeta)+\mu(\zeta)]\,\mathrm{d}\zeta$, which implies that $\mathrm{e}^{-\int_{t_0}^t[\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau} = \pi(a)$ and we can therefore rewrite

$$R(a(\tau),t(\tau)) = \pi(a) + \pi(a)\int_{t_0}^t \alpha(a(u))\frac{1}{\pi(u)}\,\mathrm{d}u,$$

since $\mathrm{e}^{\int_{t_0}^u[\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta} = \frac{1}{\mathrm{e}^{-\int_{t_0}^u[\rho(a(\eta))+\mu(a(\eta))]\,\mathrm{d}\eta}} = \frac{1}{\pi(u)}$. If we assume that α, ρ, μ are constant we can solve the equations and obtain:

$$\begin{aligned}\pi(a) &= \mathrm{e}^{-\int_0^a [\rho(a(\tau))+\mu(a(\tau))]\,\mathrm{d}\tau}\\ &= \mathrm{e}^{-\int_0^a [\rho+\mu]\,\mathrm{d}\tau}\\ &= \mathrm{e}^{-[\rho+\mu]a}.\end{aligned}$$

It remains to solve

$$\begin{aligned}\int_{t_0}^{t} \alpha(a(u))\frac{1}{\pi(u)}\,\mathrm{d}u &= \int_{t_0}^{t} \alpha \mathrm{e}^{[\rho+\mu]u}\,\mathrm{d}u\\ &= \int_0^a \alpha \mathrm{e}^{[\rho+\mu]\zeta}\,\mathrm{d}\zeta\\ &= \frac{\alpha}{\rho+\mu}\left(\mathrm{e}^{[\rho+\mu]a}-1\right).\end{aligned}$$

Combining it all together:

$$\begin{aligned}R(a,t) &= \mathrm{e}^{-[\rho+\mu]a} + \mathrm{e}^{-[\rho+\mu]a}\frac{\alpha}{\rho+\mu}\left(\mathrm{e}^{[\rho+\mu]a}-1\right)\\ &= \mathrm{e}^{-[\rho+\mu]a} + \frac{\alpha}{\rho+\mu}\left(1-\mathrm{e}^{-[\rho+\mu]a}\right)\\ &= \mathrm{e}^{-[\rho+\mu]a} + \frac{\alpha}{\rho+\mu} - \frac{\alpha}{\rho+\mu}\mathrm{e}^{-[\rho+\mu]a}\\ &= \frac{\alpha}{\rho+\mu} + \left(1-\frac{\alpha}{\rho+\mu}\right)\mathrm{e}^{-[\rho+\mu]a}.\end{aligned} \tag{32}$$

Appendix A.4. $\mathbf{R(a,t)}$ *solution*

Combining the two previous sub-sections, we can obtain the following general closed form for $R(a,t)$:

$$R(a,t) = \begin{cases} \frac{\alpha}{\rho+\mu} + \left(1-\frac{\alpha}{\rho+\mu}\right)\mathrm{e}^{-[\rho+\mu]a} & a<t\\ \frac{\alpha}{(\rho+\mu)} + \left(\bar{R}(a-t)-\frac{\alpha}{(\rho+\mu)}\right)\mathrm{e}^{-(\rho+\mu)t} & a\geq t.\end{cases} \tag{33}$$

Appendix B. Variation of Constants / Parameters methods

Given the `ODE`:

$$f' + a(x)f = 0,$$

one can solve it as follows:

$$\begin{aligned}\frac{f'}{f} + a(x) &= 0\\ \ln f + \int_x a(y)\,\mathrm{d}y &= K_0\end{aligned}$$

$$\begin{aligned} f &= \mathrm{e}^{K_0}\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y} \\ f &= K\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y}. \end{aligned}$$

If instead of 0 on the right-hand side there is a function $g(x)$:

$$f' + a(x)f = g(x),$$

then the only thing that changes is the parameter K, which is now a function of x. Therefore we must solve

$$f = K(x)\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y} \Rightarrow f' = K'(x)\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y} - a(x)K(x)\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y}.$$

It follows that

$$\begin{aligned} f' + a(x)K(x)\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y}\,\mathrm{d}y &= K'(x)\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y} \\ f' + a(x)f &= K'(x)\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y} \\ g(x) &= K'(x)\mathrm{e}^{-\int_x a(y)\,\mathrm{d}y} \\ g(x)\mathrm{e}^{\int_x a(y)\,\mathrm{d}y} &= K'(x). \end{aligned}$$

The last equation needs to be solved for the specific problem.