Published in final edited form as:

Mathematics (Basel). 2020 September ; 8(9): . doi:10.3390/math8091483.

Efficient Methods for Parameter Estimation of Ordinary and Partial Differential Equation Models of Viral Hepatitis Kinetics

Alexander Churkin[1,*], **Stephanie Lewkiewicz**[2], **Vladimir Reinharz**[3], **Harel Dahari**[4], **Danny Barash**[5,*]

[1]Department of Software Engineering, Sami Shamoon College of Engineering, Beer-Sheva 8410501, Israel

[2]Department of Mathematics, University of California at Los Angeles, Los Angeles, CA 90095, USA;

[3]Department of Computer Science, Université du Québec à Montréal, Montreal, QC H3C 3P8, Canada;

[4]Program for Experimental and Theoretical Modeling, Division of Hepatology, Department of Medicine, Stritch School of Medicine, Loyola University Medical Center, Maywoood, IL 60153, USA;

[5]Department of Computer Science, Ben-Gurion University, Beer-Sheva 8410501, Israel

Abstract

Parameter estimation in mathematical models that are based on differential equations is known to be of fundamental importance. For sophisticated models such as age-structured models that simulate biological agents, parameter estimation that addresses all cases of data points available presents a formidable challenge and efficiency considerations need to be employed in order for the method to become practical. In the case of age-structured models of viral hepatitis dynamics under antiviral treatment that deal with partial differential equations, a fully numerical parameter estimation method was developed that does not require an analytical approximation of the solution to the multiscale model equations, avoiding the necessity to derive the long-term approximation for each model. However, the method is considerably slow because of precision problems in estimating derivatives with respect to the parameters near their boundary values, making it almost impractical for general use. In order to overcome this limitation, two steps have been taken that significantly reduce the running time by orders of magnitude and thereby lead to a practical method. First, constrained optimization is used, letting the user add constraints relating to the boundary values of each parameter before the method is executed. Second, optimization is

*Correspondence: alexach3@sce.ac.il (A.C.); dbarash@cs.bgu.ac.il (D.B.); Tel.: +972-8-647-5281 (A.C.); +972-8-647-2714 (D.B.).

Author Contributions: Conceptualization, D.B.; methodology, A.C., S.L., V.R., H.D., and D.B.; software, A.C., V.R.; investigation, A.C., S.L., V.R., H.D., and D.B.; writing—original draft preparation, A.C., S.L., V.R., H.D., and D.B.; writing—review and editing, A.C., S.L., V.R., H.D., and D.B.; supervision, D.B.; funding acquisition, A.C., D.B., V.R., and H.D. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

performed by derivative-free methods, eliminating the need to evaluate expensive numerical derivative approximations. The newly efficient methods that were developed as a result of the above approach are described for hepatitis C virus kinetic models during antiviral therapy. Illustrations are provided using a user-friendly simulator that incorporates the efficient methods for both the ordinary and partial differential equation models.

Keywords

parameter estimation; constrained optimization; derivative free optimization; multiscale models; differential equations; viral hepatitis

1. Introduction

Chronic viral hepatitis (hepatitis C, hepatitis B, and hepatitis D) is a major public health concern. Approximately 500 million individuals worldwide are living with chronic viral hepatitis; above a million of those who are infected die each year, primarily from cirrhosis or liver cancer resulting from their hepatitis infection [1–3]. Deaths related to chronic hepatitis are as many as those due to human immunodeficiency virus (HIV) infection, tuberculosis, or malaria [4], and are projected to exceed the combined mortality associated with HIV infection, tuberculosis, and malaria by 2040 [5]. Only a small subset of patients are cured with currently available drugs for hepatitis B and hepatitis D. As such, a deeper understanding of hepatitis B and D infection dynamics is needed to enable the development of more curative therapeutics. Despite the significant advances in hepatitis C therapy, it is widely acknowledged that cost remains a major barrier for achieving global elimination. Thus, there still exists a need for affordable therapy with similar high efficacy and with much shorter treatment durations and vaccine development.

Mathematical models have been developed to provide insights into viral hepatitis and host dynamics during infection and the pathogenesis of infection [6–11]. The standard biphasic model for viral infection is a set of three ordinary differential equations (ODEs) with three variables. This ODE model has been used to study hepatitis C virus (HCV), hepatitis B virus (HBV), and hepatitis D virus (HDV) kinetics during antiviral treatment. It contributed to the assessment of antivirals efficacy and to our understanding of their mechanism of action [9,12–18]. The ODE model can be further simplified (termed biphasic mode) by assuming that target cells remain constant during antiviral treatment. The biphasic model has been extensively used for modeling HCV [19–27], HBV [28–31], or HDV [32–35] kinetics during antiviral treatment. Notably, we recently showed in a proof-of-concept pilot study that using the biphasic model in real time (i.e., on treatment) can shorten HCV treatment duration (and cost) with direct-acting antivirals without compromising efficacy or patient safety [36], which confirmed our retrospective biphasic modeling reports in more than 250 patients [37–41].

Modeling efforts using ODEs for understanding the intracellular viral hepatitis genome dynamics have been done in [7,42–46]. Recently, partial differential equation (termed PDE, age-structured or multiscale) models for HCV infection and treatment were developed [47–50]. These PDE models are an extension to the classical biphasic models in which the infected cell is a “black box”, producing virions but without any consideration of the intracellular viral RNA replication and degradation within the infected cell [42,43,51]. The multiscale models consider the intracellular viral RNA in an additional equation for the variable (R), with the introduction of age-dependency in addition to time-dependency, making it a PDE model. They are considerably more difficult to solve and to perform parameter estimation on compared to the biphasic model. Unlike the construction of numerical schemes in other applications, for example in the nonlinear diffusion of digital images [52–54] where accuracy can be limited, herein it is advisable to construct a stable and efficient scheme that belongs to the Runge–Kutta family with a higher accuracy than in nonlinear diffusion. Our numerical solution strategy was outlined in [55–57] and herein we continue [57] by providing an efficient parameter estimation method that follows this strategy.

Parameter estimation (or calibration) of multiscale HCV models with HCV kinetic data measured in treated patients is challenging. To overcome this, several strategies have been employed. The first strategy, employed in [48], utilizes an analytical solution named long-term approximation for solving the model equations along with calling the Levenberg–Marquardt [58,59] as a canned method for performing the fitting. The second strategy, employed in [60], transforms the multiscale model to a system of ODEs and, as such, simple parameter estimation methods can be used in the same manner as the biphasic model. The third strategy, employed in [50] that also deals with spatial models of intracellular virus replication, is based on the method of lines and utilizes canned methods for both the numerical solution of the resulting equations (Matlab’s *ode45*) and for performing the fitting (Matlab’s *fmincon*). While these strategies are adequate for specific cases, they rely on canned methods and are problematic when it comes to the user’s capability to access and control them. For these reasons, we have developed our own open source code released free of charge for the benefit of the community that allows the user to make modifications to the model and provides prospects for future development, while ensuring that it is practical in running time and enabling the user to insert constraints for the parameters that need to be estimated. In contrast to these approaches, our strategy does not rely on any canned method but fully implements our own optimization routine, thus making it suitable to other multiscale model equations by modifications inside the routine and an early preparation of the multiscale model equations by taking derivatives with respect to the parameters before the optimization procedure.

The general ideas that have led to [57], including the parameter estimation procedure described in this reference, have been laid out in order to remain self-contained. The motivation of the present work is to develop a tool that can provide similar calibration values in significantly less time. More specifically, the main contribution herein is as follows. Because of precision problems in [57] encountered with Levenberg–Marquardt that caused the parameter estimation procedure to become highly non-efficient, we developed an efficient constrained optimization procedure that is based on damped Gauss–Newton instead such that we avoid problematic use of derivatives, while alternatively offering the possibility to apply Powell’s Constrained optimization by linear approximation (COBYLA) [61] for the optimization procedure. In the following sections, we describe the model and the optimization procedure that is used in our HCVMultiscaleFit simluator. Illustrations of simultations using HCVMultiscaleFit are provided and the efficiency and practicality relative to the initial version put forth in [57] are discussed.

2. Methods

2.1. Development of Mathematical Models

2.1.1. The Standard Biphasic Model—The three variables this model keeps track of are the target cells T, in Equation (1), the infected cells I in Equation (2), and the free virus V in Equation (3). 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 cell 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:

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

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t} = \beta V(t)T(t) - \delta I(t) \tag{2}$$

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = (1 - \epsilon)pI(t) - cV(t\ ). \tag{3}$$

From the mathematical perspective, the standard biphasic model is relatively much simpler than the multiscale model. Although it is nonlinear, it can be solved analytically when assuming that T is constant (target cells remain constant during antiviral treatment).

2.1.2. The Multiscale HCV Model—A multiscale PDE model for HCV infection and treatment dynamics was introduced in [47–49]. Intracellular HCV RNA plays a biologically significant role during the HCV replication and multiscale models are considering it by additional equations for the RNA that are age-dependent, with the most complete model to date that was recently put forth in [50].

The multiscale model [47–49] can be formulated as follows:

$$\frac{\mathrm{d}T(t)}{\mathrm{d}t} = s - dT(t) - \beta V(t)T(t) \tag{4}$$

$$\frac{\partial I(a,t)}{\partial t} + \frac{\partial I(a,t)}{\partial a} = -\delta I(a,t) \tag{5}$$

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = (1 - \varepsilon_s)\int_0^\infty \rho R(a,t)I(a,t)\mathrm{d}a - cV(t) \tag{6}$$

$$\frac{\partial R(a,t)}{\partial t} + \frac{\partial R(a,t)}{\partial a} = (1-\varepsilon_\alpha)\alpha \mathrm{e}^{-\gamma t} - ((1-\varepsilon_s)\rho + \kappa\mu)R(a,t), \tag{7}$$

with the initial and boundary conditions $T(0) = \overline{T}, V(0) = \overline{\nabla}, I(0,t) = \beta V(t)T(t), I(a,0) = \overline{I}(a), R(0,t) = 1$, and $R(a,0) = \overline{R}(a)$. The initial condition $R(0, t) = 1$ reflects the assumption that a cell is infected by a single virion and therefore there is only one vRNA in an infected cell at age zero.

The four variables this model keeps track of are the target cells T in Equation (4), the infected cells I in Equation (5), the free virus V in Equation (6), and the intracellular viral RNA R in an infected cell in Equation (7).

The target cells T are produced at constant rate s, and decrease by the number of cells infected by virus in blood V at constant rate β and their death rate d. The infected cells I die at constant rate δ. The quantity of intracellular viral RNA R depends on its production α and its degradation μ and expulsion from the cell ρ. The quantity of free virus V depends on the number of assembled and released virions and their clearance rate c. The parameter γ represents the decay of replication template under therapy. 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 $\kappa \geq 1$.

The parameters that were used in the multiscale model described in [48] are depicted in Table 1. The model forms an example of our parameter estimation calibration method for PDE models developed herein that can easily be extended to include additional parameters.

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 variables are $\overline{R}(a,t)$, $\overline{I}(a,t)$, $\overline{V}$, and $\overline{T}$. The term N represents the total number of virions produced by infected cells.

These values have been previously derived in [48] and can be expressed as follows:

$$\overline{T} = c/\beta N \tag{8}$$

$$\overline{V} = (\beta N s - dc)/(\beta c) \tag{9}$$

$$\overline{I}(a) = \beta \overline{V}\,\overline{T}\mathrm{e}^{-\delta a} \tag{10}$$

$$\overline{R}(a) = \frac{\alpha}{\rho+\mu} + \left(1 - \frac{\alpha}{\rho+\mu}\right)\mathrm{e}^{-(\rho+\mu)(a)} \tag{11}$$

$$N = \frac{\rho(\alpha+\delta)}{\delta(\rho+\mu+\delta)} \tag{12}$$

It has been shown that the equations for $I(a, t)$ and $R(a.t)$ can be solved by the method of characteristics to yield:

$$I(a,t) = \begin{cases} \beta V(t-a)T(t-a)\mathrm{e}^{-\delta a} & a < t \\ \bar{I}(a-t)\mathrm{e}^{-\delta t} = \beta\bar{V}\bar{T}\mathrm{e}^{-\delta(t-a)} = (\beta Ns - dc)/(\beta N)\mathrm{e}^{-\delta a} & a > t \end{cases} \tag{13}$$

and

$$R(a,t) = \begin{cases} \frac{(1-\varepsilon_\alpha)\alpha\mathrm{e}^{-\gamma t}}{(1-\varepsilon_s)\rho+\kappa\mu-\gamma} + \left(1 - \frac{(1-\varepsilon_\alpha)\alpha\mathrm{e}^{-\gamma(t-a)}}{(1-\varepsilon_s)\rho+\kappa\mu-\gamma}\right)\mathrm{e}^{-((1-\varepsilon_s)\rho+\kappa\mu)a} & a < t \\ \frac{(1-\varepsilon_\alpha)\alpha\mathrm{e}^{-\gamma t}}{(1-\varepsilon_s)\rho+\kappa\mu-\gamma} + \left(\frac{\alpha}{\rho+\mu} + \left(1-\frac{\alpha}{\rho+\mu}\right)\mathrm{e}^{-(\rho+\mu)(a-t)} - \frac{(1-\varepsilon_\alpha)\alpha}{(1-\varepsilon_s)\rho+\kappa\mu-\gamma}\right)\mathrm{e}^{-((1-\varepsilon_s)\rho+\kappa\mu)t} & a > t \end{cases} \tag{14}$$

whereas the equations for $V(a, t)$ and $T(a, t)$ cannot be solved analytically without any approximations. The equations for $V(a, t)$ and $T(a, t)$ when using the short-term and long-term approximations can be found in [48].

2.2. Data Description

Calibration of the model was performed with data from treated patients by [48]. The data points to fit the model and on which the error is computed are only V. We assume that we start at a steady state and begin by computing the steady state given the initial parameters by using Equation (8). While the raw data are not available, we used the freely accessible tool of [62] to retrieve it from the figures directly. A visual example for one patient is available at [57].

In our method, we mostly use the default parameters from [48] that are shown in Table 2. The main difference concerns parameter s. The pre-treatment steady state viral load $\bar{V}$ in each patient is different. Since $\bar{V}$ is a necessary value in computing the long-term approximation, it was approximated as the pre-treatment viral load observed per patient. In the full model that we are implementing, we do not directly use $\bar{V}$. Instead, we have from Equation (9) that $\bar{V}$ is a function of many parameters, in particular s which is not present in the long term approximation that was outlined in [48]. Inspired by the method of [48], we chose to also fix $\bar{V}$. The counterpart in our method is that s changes per patient being, by Equation (9), equal to $(\bar{V}\beta c + dc)/(\beta N)$, where N is from Equation (12).

More details about preparing the system with data from patients and the model parameters are available in [57]. Herein, the methods are different from [57] and are significantly more efficient, but the model parameters and the system preparation are exactly the same.

2.3. Solving the Model Equations

In [48], the multiscale model equations were solved by analytical approximations but, as discussed in [56], those analytical approximations have limitations that should be alleviated. The long-term approximation is an underestimate of the PDE model since some infection events are being ignored. Moreover, for each multiscale model, the long-term approximation needs to be derived analytically, which is not a trivial task. Thus, numerical solutions provide an attractive alternative and could be easier to adjust when introducing changes to the model. A more general and comprehensive approach to parameter fitting without relying on analytical approximations would be useful. In addition, although it was shown recently that it is possible to transform the PDE multiscale model to a system of ODEs [60], this transformation problematically introduces some of the boundary conditions, e.g., ζ, as new parameters inside the model equations. A numerical approach to parameter fitting of multiscale models was recently put forth and described in [50], by the use of the method of lines and canned methods that are available in Matlab. Our new numerical approach that originated in [56] and described in [57] in detail does not rely on canned methods, with considerable benefits.

For the numerical solution of the multiscale model equations, properties such as approximation, stability, and convergence were discussed in [56] and numerical robustness was discussed in [55,56]. Future work should expand towards the advanced treatment of properties as covered in [63,64]. Concerning the numerical solution itself, we showed in [56] that the full implementation of the Rosenbrock method is preferable over the use of a canned solver in terms of efficiency and stability. Therefore, the Rosenbrock method has been implemented for the purpose of our parameter fitting method as well. In order to apply the Rosenbrock method, it is simplest to represent the system to be solved as a vector f of two functions:

$$\begin{aligned} y' = f(t, y) = \left[\frac{\mathrm{d}T}{\mathrm{d}t}, \frac{\mathrm{d}V}{\mathrm{d}t}\right]^{\top} = [s - dT - \beta VT, \\ (1 - \varepsilon_s)\int_0^{\infty} \rho R(a, t) I(a, t) \mathrm{d}a - cV\Big]^{\top}, \end{aligned} \tag{15}$$

where y is a vector with the values of $[T, V]^{\top}$ and the transpose symbol can be omitted from now on for brevity. This representation has originated in [56] for convenience with formulating the numerical schemes described in that reference. This function depends on three variables, t, V and T. While V and T are the values at the time point we are evaluating, inside the equation of I, the function $V(t - a)$ and $T(t - a)$ do depend on t directly. In our implementation, when computing the integral, we need to divide into two cases. If $a > t$, we analytically determine the values of $R(a, t)$ and $I(a, t)$ for small time steps a. When $a < t$, the system was previously solved at times $\tau_0, \ldots, \tau_n$. Therefore, we evaluate the integrals at times $a_0 = t - \tau_0, \ldots, a_n = t - \tau_n$, ensuring that the required values of $V(t - a)$ and $T(t - a)$ are already known, following the scheme presented in [56].

The Rosenbrock method additionally requires the Jacobian matrix, denoted by f'. As was shown in [57], the Jacobian can be controlled and, with some proper computational simplifications to avoid singularities that were shown to yield correct results in [57], we can implement the Rosenbrock method convincingly for both solution and parameter estimation of the multiscale models.

2.4. Parameter Estimation

2.4.1. Preliminaries—As outlined in [48], the HCV multiscale model has 12 parameters (Table 1) and the nonlinear differential equations that comprise it are stiff [56]. In addition, the integral term in the equation complicates matters, as described in [56,57]. Parameter fitting is known to be a difficult problem in general and for multiscale models, in particular, one needs to approach it carefully with the use of robust techniques for the optimization, but, at the same time, these techniques can be made highly efficient for practical computations. The novelty in this work is described next.

For efficiency reasons, we revert from the Levenberg–Marquardt method for optimization that was used in albeit different ways in both [48,57] and implement significant improvements. Already in [57], we have noticed that more difficult fitting cases take several hours to perform, and this situation needs to be remedied for a practical use of our simulator. The reason for the lengthy running times was non-trivial and only after a considerable period of time, having tried the simplest numerical method for the solution of the equations (the Euler method instead of the Rosenbrock method) and not noticing a significant time reduction in the parameter estimation calculation, we began to understand that the problem lies in the optimization method being used. We then examined interior point methods for performing constrained optimization instead of the Levenberg–Marquardt method we used in [57] and found out that the Hessian calculations in these interior point methods are problematic, causing precision problems near the parameter boundaries that are the source of running time accumulations. There was definitely a need to avoid the use of derivatives and therefore two alternative approaches were taken. The first was to try a constrained damped Gauss–Newton strategy, which can also be looked at as a simple version of Levenberg–Marquardt without gradient descent, or alternatively Levenberg–Marquardt is a pseudo second-order method with added derivatives to approximate the Hessian and thereby adds complications that should better be avoided. While in general Levenberg–Marquardt is considered more robust than Gauss–Newton, for our constrained application, the simplicity of the damped Gauss–Newton in terms of derivative calculations relative to Levenberg–Marquardt, in which also the Lagrange parameter needs to be calculated at each step, makes the damped Gauss–Newton significantly preferable. The second approach taken was that, while developing our own damped Gauss–Newton method for the constrained application, we also examined a completely derivative-free approach based on COBYLA (Constrained Optimization by Linear Approximation). These two approaches turned out to be complementary to each other as by default the quicker and sometimes somewhat more accurate damped Gauss–Newton can be tried first, but, when it fails, COBYLA can provide a good alternative or it can even be used from the start and all along a research study as the difference in the calculated error that has been minimized is quite small. This contribution allows for reaching an overall procedure for parameter estimation that is practical and by orders of magnitude less demanding in computing time relative to [57], which provides a technical breakthrough from the computational standpoint.

Thus, two newly developed methods have been introduced to perform constrained optimization for this application in an efficient manner: LSF (Least Squares Fitter using Gauss–Newton) with a flowchart shown in Figure 1 and Powell's COBYLA (Constrained Optimization by Linear Approximation) with a pseudocode shown in Algorithm 1. The latter is a derivative-free optimization method that solves the constrained optimization by linear programming. The former is a constrained optimization that performs linearization in the manner described herein.

In both approaches, the objective function to be minimized is described as follows. The objective consists of adjusting the parameters of a model function to best fit a data set. A simple data set consists of *n* points (data pairs) (x_i, y_i), $i = 1, \ldots, n$, where x_i is an independent variable and y_i is a dependent variable whose value is found by observation. The model function has the form $f(x, p)$, where *m* adjustable parameters are held in the vector *p*. The goal is to find the parameter values for the model that best fits the data. The fit of a model to a data point is measured by its residual, defined as the difference between the actual value of the dependent variable and the value predicted by the model:

$$r_i = y_i - f(x_i, p) \tag{16}$$

The least-squares method obtains the optimal parameter values by minimizing the sum of squared residuals:

$$S = \sum_{i=1}^{n} (r_i)^2 = \sum_{i=1}^{n} (y_i - f(x_i, p))^2 \tag{17}$$

2.4.2. Optimization by a Constrained Version of Nonlinear Least Squares (Gauss–Newton Method)—If we assume that $f(x)$ is twice continuously differentiable, then we can utilize Newton's method to solve the system of nonlinear equations:

$$\nabla f(x) = J(x)^T r(x) = 0, \tag{18}$$

which provides local stationary points for $f(x)$, where $r(x)$ is the vector of residuals associated with data points as functions of parameter vector *x* and *J* is the Jacobian. Written in terms of derivatives of $r(x)$ and starting from an initial guess x_0, this version of the Newton iteration scheme takes the form:

$$x_{k+1} = x_k - \left[J(x_k)^T J(x_k) + S(x_k)\right]^{-1} J(x_k)^T r(x_k), \quad k = 0, 1, 2, \ldots \tag{19}$$

where $S(x_k)$ denotes the matrix:

$$S(x_k) = \sum_{i=1}^{m} r_i(x_k) \nabla^2 r_i(x_k) . \tag{20}$$

In order to obtain the correction $\Delta x_k = x_{k+1} - x_k$, a linear system is solved by a direct or iterative method:

$$\left[J(x_k)^T J(x_k) + S(x_k)\right] \Delta x_k = -J(x_k)^T r(x_k) . \tag{21}$$

For our application, we use the Gauss–Newton method, which neglects the second term $S(x_k)$ of the Hessian, and the computation of the step Δx_k involves the solution of the linear system:

$$\left[J(x_k)^T J(x_k)\right]\Delta x_k = -J(x_k)^T r(x_k), \tag{22}$$

and $x_{k+}1 = x_k + \Delta x_k$.

In our application, we use the following steps that comprise a damped Gauss–Newton strategy:

- Start with an initial guess x_0 and iterate for $k = 0, 1, 2, \ldots$
- Solve $min_{\Delta x_k} \|J(x_k)\Delta x_k + r(x_k)\|_2$. to compute the correction Δx_k.
- Choose a step length α_k so that there is enough descent.
- Calculate the new iterate $x_{k+1} = x_k + \alpha_k \Delta x_k$.
- Check for convergence.

We choose α_k to be 1.0 at the beginning of the algorithm and decrease it by dividing by two each time the error increases relative to the previous iteration. More sophisticated damping strategies such as the Armijo–Goldstein step-length principle are not suitable in our application because of constraints violation that is described next. We also extend the Damped Gauss–Newton method to be constrained in the following way: in the case that one of the parameters, during the convergence process, exceeds its bounds (constraints provided in the GUI by the user), the algorithm assigns to this parameter its corresponding bound value instead. For this reason, we cannot apply the Armijo–Goldstein condition and need to revert to a simple damping strategy that is suitable with our constrained modification of a damped Gauss–Newton method. A flowchart of our method is shown in Figure 1.

2.4.3. Optimization by Derivative-Free Methods (COBYLA Method)—Should the Gauss–Newton method fail to carry out the optimization of Equation (18), a helpful alternative is the COBYLA algorithm, a derivative-free simplex method originally developed by Powell [65]. The parameters in the algorithm have mathematical meanings that are outside the scope of the model employed, as will be shown herein, and a pseudocode of the algorithm is available in Algorithm 1. In general, a simplex method seeks to minimize an objective function using *simplices*, where *simplex* refers to the convex hull of a set of $n + 1$ points in n-dimensional space. Such an algorithm begins by evaluating the objective function at the vertices of an initial simplex, and then strategically adjusting the simplex so that the objective function attains generally smaller values at the vertices of the new simplex than it did at those of the previous simplex. At each iteration, a vertex of the simplex may be altered, or the simplex itself rescaled, so as to guide the simplex into a region at which the objective function is minimized. When sufficient accuracy is attained, the vertex of the final simplex at which the objective function is smallest is returned as the function's minimizer.

A major benefit of both the Gauss–Newton and COBYLA algorithms is in reducing and even abolishing the use of derivatives of the objective function. In our model, the Hessian matrix associated with our objective function imposes a heavy computational burden on the optimization problem, and methods that do not require it are preferable. Numerical results indicate that COBYLA is generally very effective when the Gauss–Newton method fails; the latter, however, is quicker and more accurate than COBYLA. By default, we use the Gauss–Newton method, and, when it fails, the user is prompted to initiate COBYLA. The details of the COBYLA algorithm are described in Appendix A, beginning with a description of the Nelder and Mead simplex method from which it is derived.

Algorithm 1: COBYLA method.

```
begin
    $\rho \longleftarrow \rho_{beg}$
    $\mu \longleftarrow 0$
    $Branch \longleftarrow (*)$
    Form the initial simplex
    loop
        Ensure that $x_0$ is the optimal vertex and that $Flag = ON$ iff the simplex is acceptable
        if $Branch = (*)$ or $Flag = ON$ then
            Generate $x^*$
            if $\|x^* - x_0\|_2 \geq 0.5\rho$ then
                if $\mu$ is not large enough then
                    Revise $\mu$
                    if $x_0$ is not optimal vertex then
                        Continue
                    end
                end
                Calculate $f(x^*)$ and $\{ c_i(x^*) : i = 1, 2, \ldots, m \}$
                if $\Phi(x^*) < \Phi(x_0)$ or the change may help acceptability then
                    Revise the simplex
                end
                if $(\Phi(x^*) - \Phi(x_0))/(\hat{\Phi}(x^*) - \hat{\Phi}(x_0)) \geq 0.1$ then
                    $Branch \longleftarrow (*)$
                    Continue
                end
            end
            if $Flag = ON$ then
                if $\rho \leq \rho_{end}$ then
                    Break
                else
                    $Branch \longleftarrow (*)$
                    Update $\rho$ and $\mu$
                end
            else
                $Branch \longleftarrow (\Delta)$
            end
        else
            Calculate $x^\Delta$, $f(x^\Delta)$ and $\{ c_i(x^\Delta) : i = 1, 2, \ldots, m \}$
            Make $x^\Delta$ a vertex of the simplex
            $Branch \longleftarrow (*)$
        end
    endloop
end
```

2.5. Method Scope and Other Approaches

The strategy that was introduced in [57] and also implemented herein prepares the multiscale model equations for parameter fitting by working on them directly as an initial step. This strategy is beneficial in postponing approximations to later steps and ensuring full control of the user during the whole fitting procedure. It should be noted that, for each parameter introduced in future multiscale models, the derivative with respect to the new parameter needs to be taken and more equations need to be derived, as illustrated in this section. However, this technical procedure is significantly less complicated than deriving analytical approximations to a modified model with a change in the parameters. In our package, the code is written in Java and at present the method is hard coded for the model; therefore, some technical expertise is needed if a new model is given and the method needs to be hard coded in Java for the new model. In future work we plan to separate the model from the method and make it generic, which needs to be done only once and then it can easily handle various modifications to the model and become modeler friendly. Until that time, we do rely on some amount of expert knowledge, but, overall, it should still be easier than deriving analytical approximations to a modified model.

The importance of parameter estimation to the model was already noted in previous studies. It was addressed in [48] and attempts to come up with improved strategies were tried thereafter in [60] and in [50]. Here, we briefly relate to each of these approaches in order to remain self-contained. More information can be found in [57].

2.5.1. Parameters Change When Transforming a PDE Multiscale Model to a System of ODEs—An approach taken in [60] showed how a PDE multiscale model of hepatitis C virus can be transformed to a system of ODEs. In principle, parameter estimation should then become easier, avoiding the complications in dealing with the PDE multiscale model. However, there are side effects introduced in such a transformation, as can be noticed in Equation (9) of [60] where the boundary condition $R(t, 0) = \zeta$ gets inside the differential equations. Consequently, as admitted in the discussion of that reference, all parameters in Equations (7)–(10) must be estimated including ζ. The inclusion of boundary conditions as new parameters inside the model equations is a drawback compared to parameter estimation performed on the original multiscale model equations before the transformation. Another drawback from the perspective of parameters change is the fact that the simplest PDE multiscale model appearing in [47] was used in the transformation to ODEs, but important additions such as the inclusion of parameter γ as in [48] are not taken into account. It is not obvious how to include the parameter γ and other developments to the multiscale model inside the system of ODEs. Finally, any information regarding the age of the cell since infection is lost. Thus, if one would wish, for example, to vary the parameter α from infection to a certain time; this is not possible. In summary, while the transformation works for the simplest multiscale model, it is limited in considering developments to the multiscale model and the parameters in the system of ODEs are not the same as the parameters in the multiscale model.

2.5.2. Problematic Issues in Strategies Relying on Canned Methods—The previous approaches for parameter fitting of the multiscale model with age are all relying on canned methods. The two main strategies are the ones worked out in [48,50]. In [48], the long-term approximation is used for the solution of the multiscale model equations and Levenberg–Marquardt is used as a canned method. One drawback of such an approach is that it is limited to the multiscale model under treatment. In addition, the analytical approximation would change when various multiscale models are introduced and the elaborative derivations would need to be carried for each one, with restrictions that are incorporated by the approximation being used. Finally, as elaborated in [57], the use of a canned method is distancing the user away from having control over the main optimization procedure and the ability to tune it from the programming standpoint.

3. Results

Having described the newer and significantly more efficient methods for parameter estimation relative to [57], we present the new results obtained for both the biphasic model [26] and multiscale models [47–50]. We first provide a basic illustration with the mutliscale model in which run-time and performance comparisons between methods are generated. Then, in Appendix B, for each type of model, some examples are described. The results are presented using a newer (efficient) version of the user-friendly simulator that we have initially developed in [55,57] for both biphasic and multiscale models. We start from the biphasic model in Appendix B and end with the multiscale model in Appendix C. The simulator with a GUI is freely available at http://www.cs.bgu.ac.il/~dbarash/Churkin/SCE/Efficient/Parameter_Estimation (the efficient version, with the option to either select the biphasic or the multiscale model).

For a basic comparison between all relevant parameter estimation methods, we apply our new methods on the difficult case of the retrieved data points that was also used for this purpose in [57] to compare our efficient methods with the previous ones. Results were obtained after a few minutes instead of the several hours that was reported in [57], making our tool practical also for difficult cases. As in [57], we fitted the four treatment parameters κ, ε_s, ε_a and γ and all other parameters were selected with the values of Table 2.

We show in Table 3 the different values of those four parameters and sum of squared-errors fitted with the various methods (new efficient ones vs. previously published ones) to the data emanating from a patient. In the rightmost column, we fitted the long-term approximation with the retrieved data points using the scipy.optimize.curve_fit method, which is a Python implementation of a simple Levenberg–Marquardt scheme as a canned method. The next column to the right are the values obtained previously by the use of Levenberg–Marquardt along with the numerical method to solve the model equations as outlined in [57]. In the left columns are the values obtained by our new efficient methods. The small differences assure us that the significant efficiency achieved, thereby making our simulator a practical and useful tool, did not result in less accuracy.

To further illustrate the tool we provide, we show in Figure 2 the starting configuration after the data was inserted as input. The shown fitting curve is the one for default parameters (not considering data points) before running any fitting method. In Figures 3 and 4, the final results are shown when selecting LSF and COBYLA, respectively. In Figures 5 and 6, we present the curves of all methods shown in the same simulator window and in a separate graph, to which Table 3 corresponds.

4. Discussion

A practical and user-guided automatic procedure for parameter estimation is an important goal to achieve for mathematical models that are based on differential equations. It enables users to test a variety of fitting scenarios, either for the model calibration or model calibration with validation, by inserting different available data points of patients used for the fitting and fixed parameter values. The motivation is to use the parameters obtained by the fitting procedure to perform successful predictions for other data, where other data are data of new patients that form initial conditions to the model and successful predictions mean that the solution of the model equations yields a correct extraction of important quantities such as time to cure. In the context of viral dynamic models, even a simple model such as the biphasic model [26] that is beneficial to be tested by users requires a nonlinear method for the least squares minimization because a linear method is not sufficient [57]. The development of more complicated models such as viral dynamic models that consider intracellular viral RNA replication, namely age-structured PDE multiscale models to study viral hepatitis dynamics during antiviral therapy [47–50], presents a need for even more sophisticated strategies that perform parameter estimation while solving the model equations simultaneously. Efficient methods as developed herein are crucial such that the parameter estimation can be performed in a reasonable time.

From the parameter estimation standpoint, as previously outlined [57] and briefly mentioned in the Introduction, multiscale models are even more challenging than the biphasic model. Not only is conducting a search in at least a 10-parameter search space more difficult than in a 4-parameter search space, but also the task of solving the model equations themselves and how to connect the equations solution to the optimization procedure requires more sophistication. Previously, this was approached in [48] by using the long-term approximation along with a canned method for Levenberg–Marquardt, and in [50] by the method of lines and then employing Matlab's 4th order Runge–Kutta solver along with a canned method available in Matlab called *fmincon* for the optimization. While these strategies work sufficiently well for specific cases because of their use of canned methods, they are problematic from the standpoint of the user's capability to access and control them. Thus far, to the best of our knowledge, no specific source code for viral hepatitis kinetics besides our initial attempt at [57] (more general software such as DUNE, DuMuX, and UG4 for the solution of PDE models are available at [66–68] and would be worthwhile exploring in the future) has been released free of charge for the benefit of the community and while these strategies were described coherently in the context of presenting multiscale models, they were not intended to provide to the user a comprehensive solution of their own. There is clearly a need to provide the user with a free of charge simulator that is effortless to operate and a code that can be accessed for dissemination and future development. Furthermore, it should be practical in running time and allow inserting constraints for the parameters that need be estimated, which is not available in our initial attempt of [57] because of reverting to the standard non-constrained Levenberg–Marquardt method for the optimization and encountering numerical precision problems that were difficult to detect when developing the complete strategy for parameter estimation in our initial attempt.

The strategy we presented herein is a direct continuation to [57] and requires no canned methods utilization. It works directly on the multiscale model equations, preparing them in advance for the optimization procedure by taking their derivatives with respect to the parameters, in contrast to solving them first by an analytical approximation or performing the method of lines as a first step. For the solution of the model equations, the Rosenbrock method described in [55] is employed, as was shown to be advantageous in comparison to other solution schemes in [56]. For the constrained optimization procedure, as a departure from [57], either the Gauss–Newton or COBYLA are employed in full (not as a canned method) such that the user has access to the source code at each point in the procedure. Both Gauss–Newton and COBYLA are significantly more efficient in their constrained optimization procedure relative to the Levenberg–Marquardt employed in [57]. More complicated patient cases that took several hours of run time simulation in [57] (19.48 h reported in Table 3 for Levenberg–Marquardt) are now calculated in a few minutes (3.23 min reported in Table 3 for Gauss–Newton) on a standard PC, and simpler cases that took several minutes are now performed in seconds. Thus, the obtained results are much faster to compute than the existing solutions without sacrificing accuracy. The whole method is provided in a form of a model simulator with a user-friendly GUI, letting the user insert parameter constraints.

We note by passing that the aforementioned general software DUNE, DuMuX, and UG4 [66–68] are written in C++ using the MPI library allowing for massively parallel evaluations in the context of HPC, which might allow significantly more extended data sets to consider in the future. Thus, High Performance Computing (HPC) might also be an option for future development.

The code is open source and is divided into several packages: two fitter packages and a third default package with solver (Solver.java) class and GUI (GUI.java) class. The default package also contains different helper classes, like a class with all parameters and adapter classes to define the objective function for the fitters. The code is flexible and it is easy to add any new model solver class or any new parameters fitting class, library, or package. We use adapter design pattern to connect between the model solver and the parameters fitting algorithm. Thus, to add a new solver or fitter, one should add the solver/fitter code to the project and implement the adapter class that matches the interface of the model solver to the objective function interface of the parameters fitting method. In addition, one should change 2–3 rows in the GUI.java class to make use of a new solver/fitter from the GUI interface.

5. Conclusions and Future Work

The efficient methods described herein make the simulator a practical tool that is distributed free of charge for the benefit of the community and the dissemination of viral hepatitis models. Furthermore, the methods for parameter estimation employed can conceptually be used in other mathematical models in biomedicine.

Future work would include the development of the code in several directions. First, the code can be made more modular such that the modeler can easily implement the method for a different model or a modified version of one of these models. In this way, portability of the method to other models can be achieved such that a significant modification of the code is not needed as a consequence of a change in the model, ensuring that the modification is relatively straightforward. Second, at present, individual fits to individual time course profiles is available, which is useful when one wants to describe viral dynamics within one patient. The code can be developed for use also for fitting the in vitro time course profiles of pooled patient datasets. Thus, as future work, having the option to import and fit the models to repeated/multiple measurements would be useful.

From the numerical perspective, it might be possible to try weak methods for the solution of the model equations such as finite elements, finite volumes, or discontinuous Galerkin as described in [63,64]. The two time scales might present different challenges as compared to PDEs that are dependent on time and space in their partial derivatives. Independently, much of the computations are present in the optimization stage as compared to the solution stage and therefore efforts centered on the model equations solution could focus on simplified strategies, if at all possible, for the benefit of gaining more efficiency.

Finally, machine learning methods can be used to improve parameter estimation. There are already enough patient cases, as more than 250 patients have been modeled, which can be used to prepare the data for the parameter estimation of our simulator. Machine learning can then be used for outliers' removal, replacement of the incorrect and missed data with the correct one (currently done manually), and correction of the data for the parameters and time to cure estimation. The machine learning algorithm can then be integrated with the parameter estimation method to yield an overall improved procedure.

Funding:

This research was funded by the U.S. National Institutes of Health Grant Nos. R01-AI078881, R01-AI144112, R01-AI146917, and R01-GM121600.

Appendix A.: Details of the COBYLA Method

The original simplex method, devised by Spendley, Hext, and Himsworth [61,69], seeks to advance a simplex towards a minimizer of $f(x): \mathbb{R}^n \to \mathbb{R}$ by reflecting vertices at which f is large across opposite faces of the simplex in the hopes of reducing f. If the initial simplex has vertices $x_0, x_1, \ldots, x_n$, ordered so that $f(x_0) \leq f(x_1) \leq \ldots \leq f(x_n)$, then x_n is the vertex at which f is largest, and is considered for revision. The vector $\hat{x}$, defined below, is proposed as a replacement:

$$\hat{x} := \frac{2}{n}\left(\sum_{i=0}^{n-1} x_i\right) - x_n, \tag{A1}$$

$$= \frac{1}{n}\left(\sum_{i=0}^{n-1} x_i\right) + \left(\frac{1}{n}\left(\sum_{i=0}^{n-1} x_i\right) - x_n\right), \tag{A2}$$

$$:= \bar{x} + (\bar{x} - x_n)\,. \tag{A3}$$

Note that the vector $\bar{x}$ is the centroid of the convex hull of the points $x_0, x_1, \ldots, x_{n-1}$, so that $\hat{x}$ is the reflection of the vertex through the face of the simplex opposite x_n. As such, the volume of the simplex is preserved if the vertex exchange $x_n \to \hat{x}$ is made, with the swap occurring if $f(\hat{x}) < f(x_{n-1})$. If $f(\hat{x})$ is comparable to $f(x_n)$, the assumption is made that f is minimized in the area between x_n and $\hat{x}$, so that the change $x_n \to \hat{x}$ simply bypasses this region. To access this interior region, the simplex is rescaled without replacing any of the vertices with $\hat{x}$. The optimal vertex, x_0, is left alone, and each vertex x_k, for $k = 1, \ldots, n$, is replaced with $(1/2)(x_0 + x_k)$. Both such changes to the simplex are illustrated in the case $n = 2$ in Figure A1.

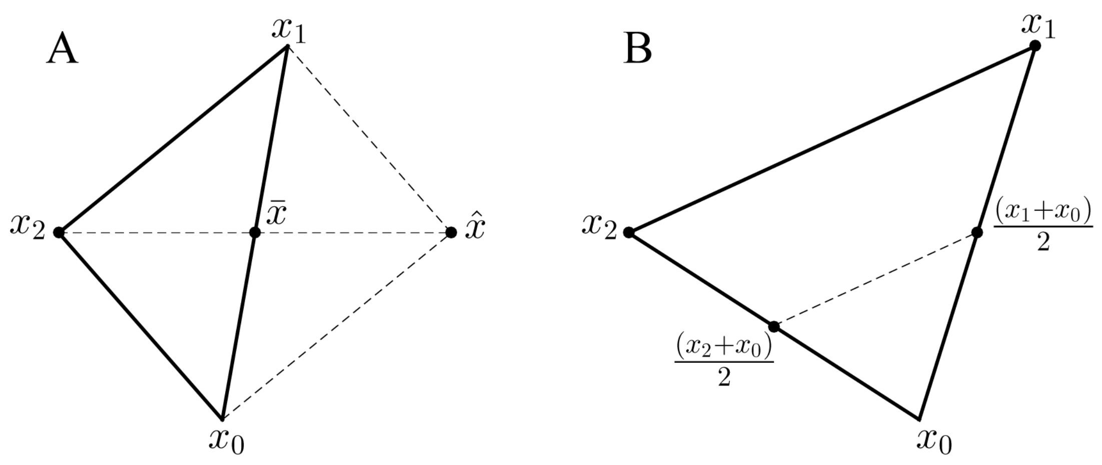


Figure A1.
Illustration of the original simplex method. The points x_0, x_1, and x_2 form the initial simplex. (**A**) The point $\bar{x}$ is the midpoint of the line joining x_0 and x_1, and $\hat{x}$ is the reflection of x_2 through this line. If $f(\hat{x}) < f(x_1), x_2$ is replaced by $\hat{x}$, shifting the location of the simplex. (**B**) If $f(\hat{x}) \geq f(x_1), x_2$ is replaced with $(1/2)(x_2 + x_0)$ and x_1 is replaced with $(1/2)(x_1 + x_0)$, reducing the volume of the simplex.

The Nelder–Mead method [70] expands on this basic simplex method, removing much of the inefficiency that arises when rescalings result in a small simplex that takes longer to converge to the function's minimizing region. It does so by moving the vertex x_n to a new point along the line joining x_n and $\hat{x}$, strategically chosen to reduce f as much as possible. A generic expression for the new vertex, x_{new}, is now,

$$x_{\text{new}} := \bar{x} + \theta(\bar{x} - x_n), \tag{A4}$$

where $\theta > 0$ may be different at each iteration. In the case of a linear f, we have that

$$f(x_{\text{new}}) = f(\bar{x}) + \theta(f(\bar{x}) - f(x_n)) \leq f(\bar{x}), \tag{A5}$$

where the last equality above follows from the fact that $f(x_n) \geq f(x_k)$ for all $k = 0, 1, \ldots, n - 1$, which implies $f(\bar{x}) \leq f(x_n)$. Regarding the linear case as a proxy for the more general case, we expect such a vector x_{new} to decrease the value of f, if θ is chosen well. One particular implementation bases the choice of θ—and thus the new vertex—on the value of $f(\hat{x})$ relative to the value of f at other vertices. The new vertex, $\check{x}$, is defined as such:

$$\begin{cases} 2\hat{x} - \bar{x} & \text{if } f(\hat{x}) < f(x_0), \quad \text{(A6a)} \\ \frac{1}{2}(\bar{x} + \hat{x}) & \text{if } f(x_0) \leq f(\hat{x}) < f(x_{n-1}), \quad \text{(A6b)} \\ \frac{1}{2}(x_n + \bar{x}) & \text{if } f(x_{n-1}) \leq f(\hat{x}). \quad \text{(A6c)} \end{cases}$$

The choices in Equations (A6a)–(A6c) are obtained by taking $\theta = 2$, (1/2), (−1/2), respectively, in Equation (A4), and are depicted in Figure A2. If $f(\hat{x}) < f(x_0)$, as in Equation

(A6a), then f decreases so significantly along the line from x_n to $\hat{x}$ that choosing a new vertex, $2\hat{x} - \bar{x}$, even further down the line is presumed to result in a greater reduction. If $f(x_0) \le f(\hat{x}) < f(x_{n-1})$, as in Equation (A6b), the reduction in f is less significant, and the new vertex is placed between $\hat{x}$ and the face of the simplex opposite x_n. If $f(x_{n-1}) \le f(\hat{x})$, as in Equation (A6c), the reduction in f at $\hat{x}$ is minimal, and $\check{x}$ is placed between x_n and the face of the opposite simplex, as placement of the vertex near $\hat{x}$ is not warranted.

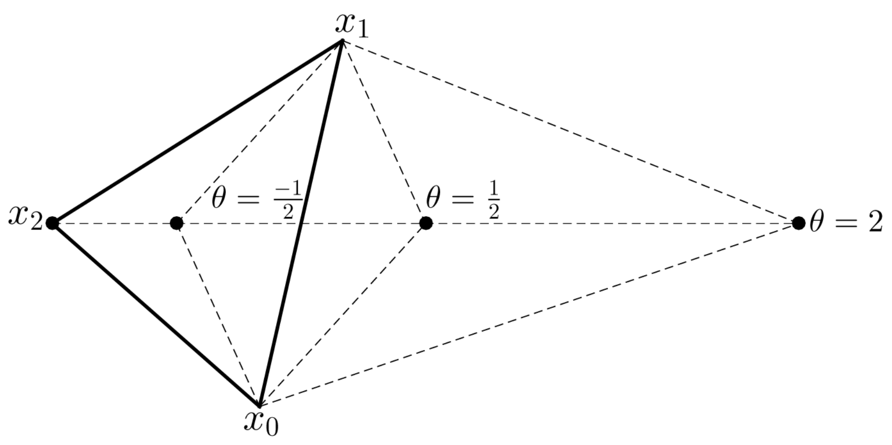


Figure A2.
Illustration of the Nelder–Mead method. The points x_0, x_1 and x_2 form the initial simplex. The vertex x_2 is replaced with a vertex of the form $x_{\text{new}} = \bar{x} + \theta(\bar{x} - x_2)$. If $f(\hat{x}) < f(x_0), \theta = 2$, if $f(x_0) \le f(\hat{x}) < f(x_{n-1}), \theta = 1/2$, and if $f(x_{n-1}) \le f(\hat{x}), \theta = -1/2$.

The COBYLA algorithm is used to minimize an objective function $f(x): \mathbb{R}^n \to \mathbb{R}$ subject to the set of $m \in \mathbb{N}$ constraints,

$$\{c_i(x) \ge 0 : i = 1, 2, \ldots, m\}, \tag{A7}$$

where $c_i(x): \mathbb{R}^n \to \mathbb{R}$ for each $i = 1, \cdots, m$. As derivatives are omitted from the algorithm entirely, no smoothness assumptions are required for the functions f, c_i; they must simply be well-defined on $\mathbb{R}^n$. After generating the initial simplex from an initial guess as to the location of the minimizer, the optimal vertex is identified and labeled x_0. In this case, a vertex of the simplex is considered optimal if $\Phi(x_0) \le \Phi(x_k)$, for $k = 1, \cdots, n$, where the x_k are the other n vertices of the simplex, and $\Phi(x)$ is defined by

$$\Phi(x) = f(x) + \mu[\max\{-c_i(x) : i = 1, \cdots, m\}]_+, \tag{A8}$$

with $[x]_+ = \max\{x, 0\}$, and $\mu \ge 0$ a constant parameter. The optimality of a point x is thus affected by both the value of $f(x)$ and how closely it satisfies the constraints in Equation (A7). If $c_i(x) \ge 0$ for all $i = 1, \cdots, m$, then $\Phi(x) = f(x)$, but if at least one constraint is violated, then $\Phi(x) > f(x)$, lessening the "worth" of the point x as an approximation to the minimizer. From there, each iteration of the algorithm generates a new candidate vertex, designed to either replace an existing vertex with one that decreases $\Phi(x)$ or improves the shape of the simplex. The shape of the simplex is particularly crucial in this algorithm because its vertices are used to define linear programming problems, from which new candidate vertices designed to improve the optimality condition are derived. Specifically, if $\{x_k : k = 0, \cdots, n\}$ are the vertices of the current simplex, we let $\hat{f}(x) : \mathbb{R}^n \to \mathbb{R}$ be the unique affine function that passes through points $(x_k, f(x_k)) \in \mathbb{R}^{n+1}$, and, analogously, we let $\hat{c}_i(x) : \mathbb{R}^n \to \mathbb{R}$ be the unique affine function that passes through the points $(x_k, c_i(x_k))$. Should the shape of the simplex be "acceptable"—in a way to be defined later—a new candidate vertex is chosen to improve optimality by minimizing $\hat{f}(x)$ subject to the constraints $\{\hat{c}_i(x) \ge 0\}$. Should the simplex be of an unacceptable shape, the linear programming problem may be ill-defined or fail to provide a reasonable approximation to the functions $f(x)$, $c_i(x)$. If this newly generated vector improves the value of Φ, it replaces a vertex of the simplex. This process continues until a pre-determined final trust region radius, which represents the desired accuracy of the approximation to the minimizer, is achieved.

The algorithm takes as inputs an initial guess, x_0, as to the location of the minimizer, and the constants ρ_{beg}, $\rho_{\text{end}} > 0$, which represent the initial and final trust region radii. Additionally, μ is set to zero. At the start, the initial simplex is generated from x_0 and ρ_{beg}. The vector x_0 is one vertex, with the other vertices, x_k, $k = 1, \ldots, n$, defined by $x_k = x_0 + \rho_{\text{beg}} e_k$, for $k = 1, \ldots, n$. Here, e_k is the k-th coordinate vector. After each x_k is generated, $f(x_k)$ is computed, and the labels of the vectors x_0 and x_k are swapped if $f(x_k) < f(x_0)$, to ensure that f is minimized at x_0. After the initial simplex is defined, the algorithm proceeds to advance the simplex at each iteration by either generating a new candidate vertex, denoted x^*, to decrease $\Phi(x)$, or an alternate vertex, denoted x^Δ, to improve the shape of the simplex. Each iteration begins by ensuring that x_0 is the optimal vertex–and relabeling vertices if it is not–before assessing the suitability of the simplex. For this, denote by σ^k the Euclidean distance from the vertex x_k to the face of the simplex opposite x_k, and, by η^k, the length of the segment joining x_k to x_0. The simplex is deemed to have an acceptable shape if and only if $\sigma^k \ge \alpha\rho$ and $\eta^k \le \beta\rho$ for all $k = 1, \ldots, n$, where α, β satisfy $0 < \alpha < 1 < \beta$. These conditions prevent the development of flat, degenerate simplices, which would result in poorly-formulated linear programming problems. If the simplex is of an unacceptable shape, a vertex x^Δ is generated; otherwise, a candidate vertex x^* is computed.

The iteration immediately following formation of the initial simplex always generates a vertex x^*, as the initial simplex satisfies $\sigma^k = \eta^k = \rho_{\text{beg}}$ for all $k = 1, \ldots, n$, and is thus always acceptable. The loop that generates such an x^* in general is described below; in the very first iteration, $\rho = \rho_{\text{beg}}$. The vector x^* is generated by minimizing $\hat{f}(x)$ subject to the constraints in Equation (A7) and the trust region condition,

$$\|x - x_0\|_2 \le \rho, \tag{A9}$$

as illustrated in Figure A3. Should the constraints in Equations (A7) and (A9) be inconsistent with one another, the candidate x^* is chosen by minimizing the "greatest constraint violation" function, $\widehat{M}(x)$, defined by,

$$\widehat{M}(x) := \max\{-\hat{c}_i(x) : i = 1, \ldots, m\} \tag{A10}$$

subject to the trust region constraint in Equation (A9). This process is illustrated in Figure A4. If there exist multiple x^* that satisfy $\widehat{M}(x^*) = \min\left\{\widehat{M}(x) : \|x - x^{(0)}\|_2 \leq \rho\right\}$, then the x^* that minimizes $\hat{f}(x)$ is chosen from among those that minimized $\widehat{M}(x)$. If there are multiple such minimizers of $\hat{f}(x)$, then the one that minimizes $\|x - x_0\|_2$ is chosen.

When the appropriate x^* is identified, the condition $\|x^* - x_0\|_2 < \frac{1}{2}\rho$ is tested. If $\|x^* - x_0\|_2 \geq \frac{1}{2}\rho$, the relative "size" of the parameter μ is evaluated. To this end, denote by $\bar{\mu}$ the smallest value of μ for which $\widehat{\Phi}(x^*) \leq \widehat{\Phi}(x_0)$, where

$$\widehat{\Phi}(x) = \hat{f}(x) + \mu[\max\{-\hat{c}_i(x) : i = 1, \ldots, m\}]_+ = \hat{f}(x) + \mu[\widehat{M}(x)]_+ . \qquad \text{(A11)}$$

If Equations (A7) and (A9) are consistent with one another, then x^* had been chosen to minimize $\hat{f}(x)$, and it follows that $\widehat{\Phi}(x^*) = \hat{f}(x^*) \leq \hat{f}(x_0) = \widehat{\Phi}(x_0)$. In this case, $\bar{u} = 0$. Should Equations (A7) and (A9) be inconsistent, then x^* had been chosen to minimize $\widehat{M}(x)$ as in Equation (A10), implying that $\widehat{M}(x_0) - \widehat{M}(x^*) \geq 0$. If $\widehat{M}(x_0) - \widehat{M}(x^*) = 0$, then $\widehat{M}$ has at least two minimizers, and x^* had been chosen to minimize $\hat{f}(x)$, implying $\widehat{\Phi}(x^*) \leq \widehat{\Phi}(x_0)$ from Equation (A11). If $\widehat{M}(x_0) - \widehat{M}(x^*) > 0$, there exists a $\bar{\mu}$ sufficiently large to guarantee $\hat{f}(x_0) - \hat{f}(x^*) + \mu\left(\widehat{M}(x_0) - \widehat{M}(x^*)\right) > 0$, even if $\hat{f}(x_0) - \hat{f}(x^*) < 0$, implying $\widehat{\Phi}(x^*) \leq \widehat{\Phi}(x_0)$. for $\mu \geq \bar{\mu}$ If the current value of μ satisfies $\mu \geq \frac{2}{3}\bar{\mu}$, then μ is considered "sufficiently large" and its value is left alone. Otherwise, μ is increased to 2μ.

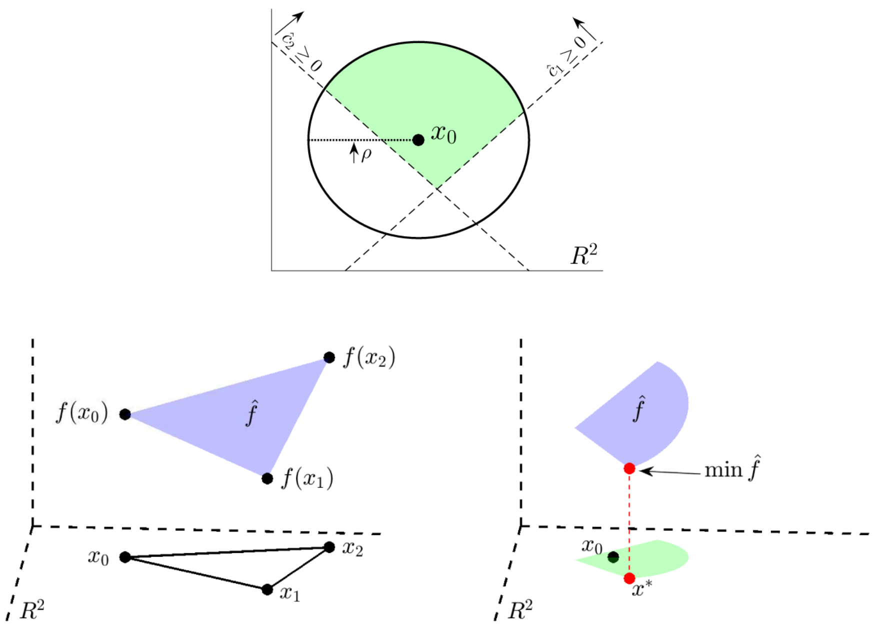


Figure A3.
Minimization of $\hat{f}$ The candidate vertex x^* is computed by minimizing $\hat{f}(x)$ subject to the constraints $\hat{c}_1$, $\hat{c}_1 \geq 0$ within the trust region $\|x - x_0\|_2 \leq \rho$. (top) The region of optimization (green) is the intersection of the trust region $\|x - x_0\|_2 \leq \rho$ with the half planes defined by the

affine constraints $\hat{c}_1 \geq 0$ and $\hat{c}_2 \geq 0$. (bottom left) The function, $\hat{f}$, to be minimized is represented graphically by the plane (blue) passing through points $(x_0, f(x_0))$, $(x_1, f(x_1))$, $(x_2, f(x_2))$. (bottom right) The vertex x^* is defined to be the point within the region of optimization (green) at which $\hat{f}$ (blue) is minimized.

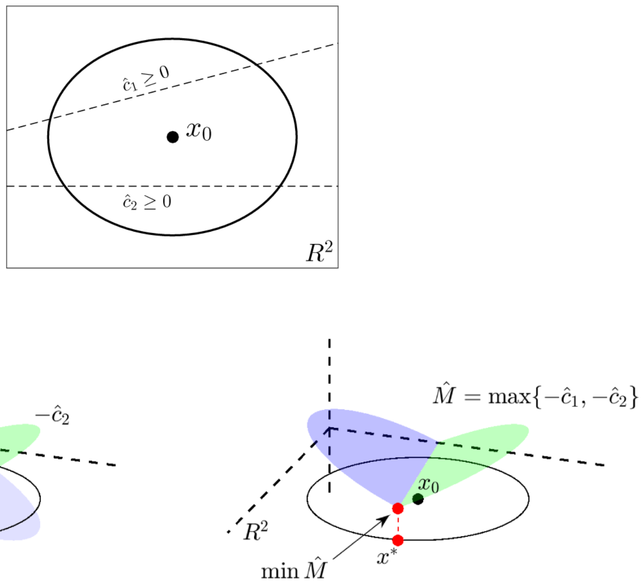


Figure A4.
Minimization of $\widehat{M}$ Should the constraints $\hat{c}_i(x) \geq 0$ be inconsistent with one another within the trust region $\|x - x_0\|_2 \leq \rho$, the candidate vertex x^* is chosen to minimize $\widehat{M} := \max\{-\hat{c}_i(x) : i = 1, \ldots, m\}$. (top) The constraints $\hat{c}_1(x) \geq 0$ and $\hat{c}_2(x) \geq 0$ are inconsistent within the region $\|x - x_0\|_2 \leq \rho$. (bottom left) Graphs of the affine functions $-\hat{c}_1(x)$ (blue) and $-\hat{c}_2(x)$ (green). (bottom right) The vertex x^* is defined to be the point within the trust region (black circle) at which $\widehat{M}$ is minimized.

If μ is increased, it may no longer be the case that x_0 is optimal, in the sense that there may exist some k between 1 and n for which $\Phi(x_k) < \Phi(x_0)$. From the form of Φ in Equation (A8), this reversal of the original order relation $\Phi(x_k) \geq \Phi(x_0)$ as μ increases can only occur if $[M(x_0)]_+ > [M(x_k)]_+$, with $M(x)$ defined as:

$$M(x) := \max\{-c_i(x) : i = 1, \ldots, m\}. \tag{A12}$$

If x_0 is no longer optimal, the process returns to the start of the loop, from which it first rearranges the labeling of the vertices so as to label the optimal one x_0. No change was made to the vertices of the simplex beyond this relabeling, and if the simplex had been acceptable previously, it will remain acceptable. In this case, another candidate vertex x^* will be computed with respect to the new x_0. The process of generating x^*, increasing μ, and then relabeling the vertices can only happen a finite number of times; the labels of the vertices x_0 and x_k are only switched if $M(x_0) > M(x_k)$, meaning that the value of M decreases each time a vertex exchange is made. Thus, increasing μ can only change the order relation between the values $\Phi(x_k)$ until the optimal vertex x_0 also satisfies $M(x_0) = \min\{M(x_k) : k = 0, \ldots, n\}$, at the latest.

If μ is sufficiently large and x_0 is optimal, or once these conditions have been achieved, the possible replacement of one vertex with the new candidate vertex x^* is considered. The values of $f(x^*)$ and $c_i(x^*)$ are computed, and the simplex is revised to incorporate x^* as a vertex if $\Phi(x^*) < \Phi(x_0)$, or if such a revision will improve acceptability of the simplex. The choice of which existing vertex should be replaced with x^* is determined by the predicted effect of the change on the acceptability conditions and the volume of the simplex. For this, consider the quantity $\bar{\sigma}^k$, defined to be the Euclidean distance from x^* to the face of the simplex opposite x_k. If x_k is replaced with x^*, but all other vertices are left unchanged, the volume, V^*, of the new simplex is related to the volume, V^k, of the original simplex by the formula,

$$V^* = \frac{\bar{\sigma}^k}{\sigma^k} V^k . \tag{A13}$$

This follows from the standard formula for the volume of a simplex in $\mathbb{R}^n$, which yields $V^k = (1/n)V^H \sigma^k, V^* = (1/n)V^H \bar{\sigma}^k$, where V^H is the volume of the convex hull of the n points $x_0, x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n$. It is considered advantageous for a change in the vertices to increase the volume of the simplex, and, as such, vertices x_k for which $\bar{\sigma}^k > \sigma^k$ are sought for replacement by x^*. Specifically, consider the set J defined by

$$\boldsymbol{J} := \left\{ j : \bar{\sigma}^j \geq \sigma^j \right\} \cup \left\{ j : \bar{\sigma}^j \geq \alpha\rho \right\}, \tag{A14}$$

thus consisting of indices j for which x^j could be replaced with x^* either without decreasing the volume of the simplex or without disturbing the acceptability condition $\sigma^j \geq \alpha\rho$. To consider the effect of a change on the acceptability condition $\eta^k \leq \beta\rho$, note that, if a vertex is replaced by x^*, the optimal vertex of the new simplex will be either the previous optimal vertex, x_0, or the vertex x^* itself. Denoting the new optimal vertex by $\bar{x}_0$, if J is nonempty, let $\ell \in \mathbb{N}$ be defined as

$$\ell := \min\left\{ k \in \mathbb{N} \cap [1, n] : \|x_k - \bar{x}_0\|_2 = \max\left\{ \|x_j - \bar{x}_0\|_2 : j \in \boldsymbol{J} \right\} \right\}, \tag{A15}$$

so that $x_{(\ell)}$ is the vertex with smallest index at a maximal distance from the optimal vertex. If $\|x^\ell - \bar{x}_0\|_2 > \delta\rho$, for $1 < \delta \leq \beta$, then x_k is replaced by x^*. If these conditions are not met, the simplex is revised regardless, as long as either $\Phi(x^*) < \Phi(x_0)$ or there exists an index k for which $\bar{\sigma}^k > \sigma^k$—that is, if Φ is smaller at the new candidate vertex, or a change increases the simplex's volume. In this case, x_0 is replaced by x_ℓ, where ℓ is now defined as

$$\ell := \min\left\{ k \in \mathbb{N} \cap [1, n] : \bar{\sigma}^k / \sigma^k = \max\left\{ \bar{\sigma}^j / \sigma^j : j \in \mathbb{N} \cap [1, n] \right\} \right\}. \tag{A16}$$

Note that, even though the simplex is updated whenever $\Phi(x^*) < \Phi(x_0)$, the vertex to discard is chosen based on the effect the change will have on the volume and acceptability of the simplex, with no regard for the values of Φ at different vertices. This process almost always results in an update to the simplex, with an update failing to occur only if Φ is not decreased at x^*, no vertex swap would increase the volume of the simplex, and either J is empty or $\|x_j - \bar{x}_0\|_2 \leq \delta\rho$ for all $j = 1, \ldots, n$. The set J is empty if each hypothetical vector swap actually decreases the volume of the simplex and fails to maintain the first acceptability condition, $\bar{\sigma}^j \geq \alpha\rho$.

After potentially making a vector replacement, the progress made in reducing Φ as the simplex advances is examined. If sufficient progress is not made, the trust region radius, ρ, will be reduced. To determine if such a reduction should be made, the change in Φ at x^* is compared to the change in $\widehat{\Phi}$. The condition

$$\frac{\Phi(x^*) - \Phi(x_0)}{\widehat{\Phi}(x^*) - \widehat{\Phi}(x_0)} \leq 0.1 \tag{A17}$$

is tested. If Cond. (A17) fails, then the improvement to Φ is at least 10% of the improvement to $\widehat{\Phi}$. This improvement is considered significant enough, and the iteration returns to the start of the loop to generate a successive x^*, assuming the simplex is still acceptable. If Cond. (A17) fails, the improvement to x^* will be less than 10% of the improvement to $\widehat{\Phi}$, and a decrease in the trust region radius is called for. Before the trust region radius is assessed, the acceptability of the simplex is checked. If the simplex is unacceptable, the iteration returns to the start of the loop, and generates an alternate vertex x^{Δ}. If the simplex is acceptable, the condition $\rho \leq \rho_{\text{end}}$ is tested. If $\rho \leq \rho_{\text{end}}$, the final trust region radius has been reached, and the algorithm terminates. If $\rho > \rho_{\text{end}}$, further advances to the simplex are required to reduce the trust region radius to its final value. The iteration returns to the start of the loop, and will generate a new candidate, x^*, as the acceptability of the simplex has already been verified. Before this, ρ and μ are updated. If $\rho > 3\rho_{\text{end}}$, then ρ is decreased by half. If $\rho \leq 3\rho_{\text{end}}$, then ρ is set to ρ_{end} itself.

It is considered sensible to update μ whenever ρ is updated, as the value of μ can become quite large. A constraint function $c_i(x)$ is regarded as "significant" to Φ if $i \in I$, where

$$I := \left\{ i \in \mathbb{N} \cap [1, m] : c_i^{\min} < (1/2)c_i^{\max} \right\}, \tag{A18}$$

with $c_i^{\min}$ and $c_i^{\max}$ representing the minimum and maximum values of c_i at the vertices of the current simplex. If I is empty, $\mu = 0$, and if I is nonempty, μ is set to the value,

$$\frac{\left\{ \max_{k = 0, 1, \ldots, n} f(x_k) - \min_{k = 0, 1, \ldots, n} f(x_k) \right\}}{\min\left\{ [c_i^{\max}]_+ - c_i^{\min} : i = 1, \ldots, m \right\}}, \tag{A19}$$

assuming that the quantity in Equation (A19) is less than the current value of μ.

If after generating the candidate vector x^* the condition $\|x^* - x_0\|_2 < \frac{1}{2}\rho$ is met, much of the previously described process is omitted. The algorithm proceeds as it did after advancing the simplex and verifying Cond. (A17), by first checking the acceptability of the simplex and revising, if necessary, and then checking the condition $\rho \le \rho_{\text{end}}$. As before, if $\rho \le \rho_{\text{end}}$, the algorithm terminates, and, if $\rho > \rho_{\text{end}}$, ρ and μ are updated as previously described and the process returns to the start of the loop to generate a new x^*.

It only remains to describe the process of updating the simplex to improve its acceptability. This is done by generating an alternate vertex, x^Δ, to replace one of the vertices of the simplex. Recall that, if the simplex is unacceptable, then there either exists a $j \in \mathbb{N} \cap [1, n]$ such that $\sigma^k < \alpha\rho$ or such that $\eta^k > \beta\rho$. If the latter is true, define $\ell \in \mathbb{N} \cap [1, n]$ to be the index that satisfies

$$\eta^\ell = \max\left\{\eta^k : k = 1, \ldots, n\right\}. \tag{A20}$$

Otherwise, define ℓ to be the index that satisfies

$$\sigma^\ell = \min\left\{\sigma^k : k = 1, \ldots, n\right\}. \tag{A21}$$

The vertex x_ℓ is then one that violates one of the acceptability conditions most egregiously, either by being the closest to the optimal vertex or the farthest from its opposite face. The vertex x_ℓ will be replaced by, x^Δ, where x^Δ is defined by

$$x^\Delta = x_0 \pm \gamma\rho v_\ell, \tag{A22}$$

where v_ℓ is the unit vector perpendicular to the face of the simplex opposite the outgoing vertex x_ℓ, and $\gamma \in (\alpha, 1)$. The + or − is chosen to minimize $\widehat{\Phi}$. The new vertex x_Δ, illustrated in Figure A5, maintains the general position of x_{ℓ} relative to the opposite face of the simplex, while satisfying $\sigma^\Delta = \eta^\Delta = \|x^\Delta - x_0\|_2 = \gamma\rho \in (\alpha\rho, \beta\rho)$. Even though the acceptability condition is not violated at the new vertex x^Δ, acceptability may still be violated at other vertices. After replacing x_ℓ with x_Δ, the process always generates a vertex of the type x^*, as opposed to conducting several replacements to improve acceptability. As described previously, the algorithm continues to advance the simplex by replacing current vertices with improvements of the form x^* and x^Δ until the condition $\rho \le \rho_{\text{end}}$ is achieved.

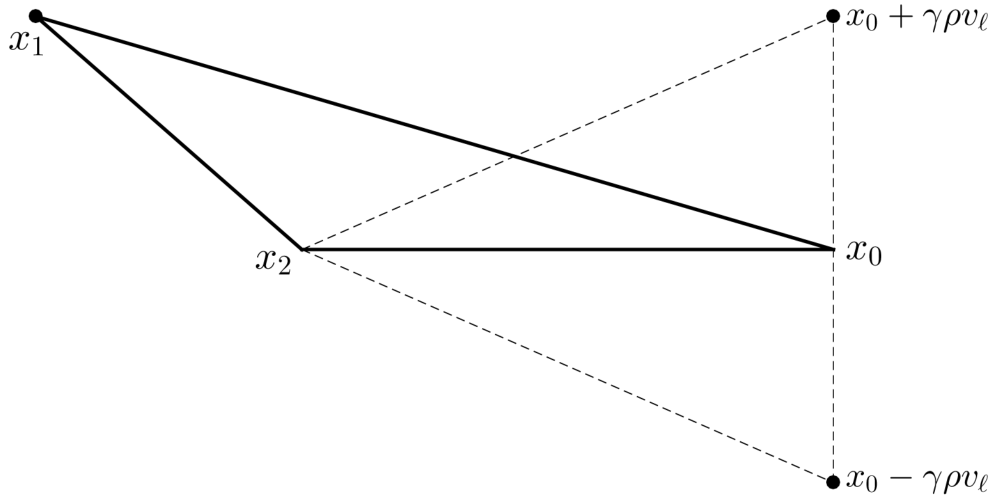
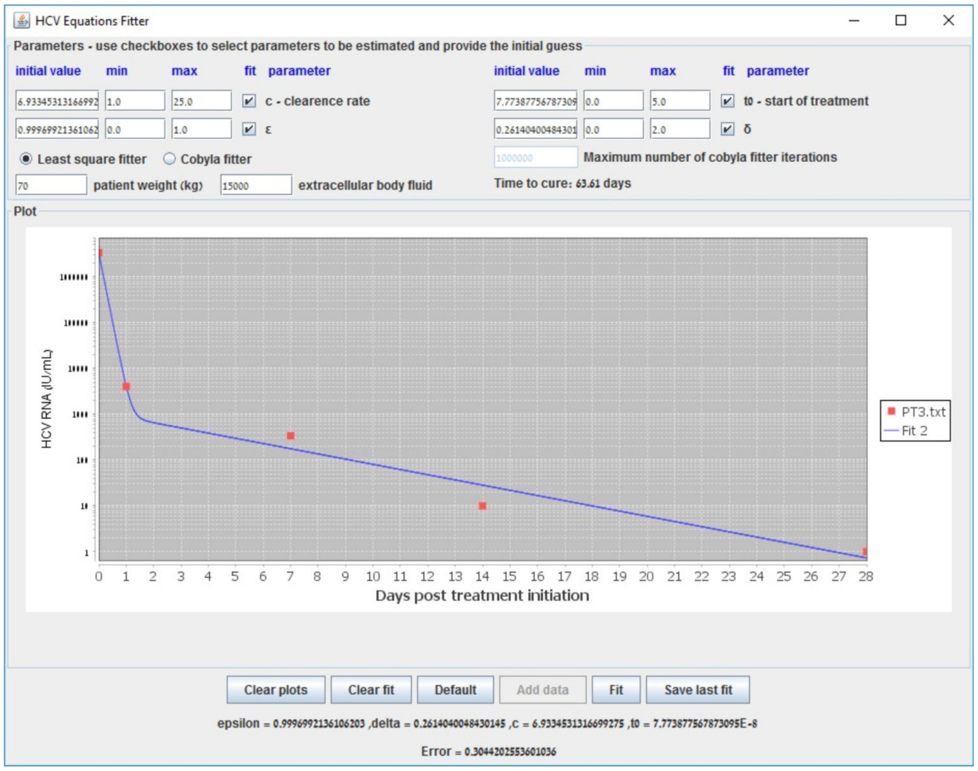


Figure A5.
Illustration of the new vector x^{Δ}, generated to improve the shape of the simplex. The vertex x_1 is replaced with either $x^{\Delta} = x_0 + \gamma\rho v_\ell$ or $x^{\Delta} = x_0 - \gamma\rho v_\ell$, whichever point results in a smaller value of $\widehat{\Phi}$.

Appendix B.: Parameter Estimation in the Biphasic Model

We begin with the standard model for HCV dynamics, the biphasic model of Neumann et al. [26]. Although the model is nonlinear, it can be solved analytically when assuming that the target cells T variable is constant. We incorporated the analytical solution described in [26] to our simulator and performed parameter estimation using a constrained optimization Gauss–Newton solver to solve the minimum least squares problem, which is more efficient and stable than the non-constrained nonlinear solver with a damping factor described in [57]. There can be instances in which the Gauss–Newton solver fails, although it is the simplest and most efficient, in which case the COBYLA solver should be used instead by selecting its option in the simulator or it can be used in all instances from the start. Figures A6 and A7 present fitting results from two patients (Pts). Figure A6 corresponds to Pt3 who was treated with mavyret [36] where LSF was selected for the optimization (default). In a case that corresponds to Pt285003 who was treated with epclusa [36] where LSF was selected for the optimization (default), a warning appeared because of a failure, after which Figure A7 corresponds to the same case, but this time COBYLA was selected for the optimization and succeeded to yield a fit. Our current method has recently been used in [71,72]. A webpage with user instructions is available at http://www.cs.bgu.ac.il/~dbarash/Churkin/SCE/Efficient/Parameter_Estimation/Biphasic.



Parameters - use checkboxes to select parameters to be estimated and provide the initial guess

initial value	min	max	fit	parameter	initial value	min	max	fit	parameter
6.93345313166992	1.0	25.0	✓	c - clearence rate	7.77387756787309	0.0	5.0	✓	t0 - start of treatment
0.99969921361062	0.0	1.0	✓	ε	0.26140400484301	0.0	2.0	✓	δ

[● Least square fitter] ○ Cobyla fitter

1000000 Maximum number of cobyla fitter iterations

70 patient weight (kg) 15000 extracellular body fluid

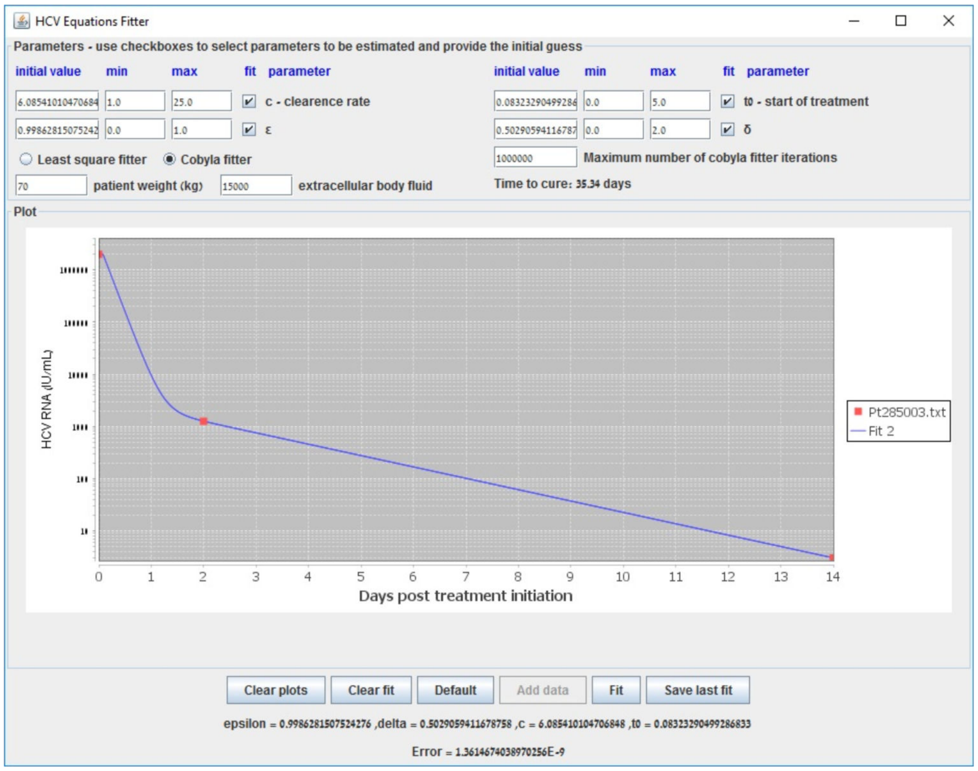
Time to cure: 63.61 days

epsilon = 0.9996992136106203 ,delta = 0.2614040048430145 ,c = 6.9334531316699275 ,t0 = 7.773877567873095E-8

Error = 0.3044202553601036

Figure A6.

Biphasic model fitting example with data taken from [36] of a patient who was treated with mavyret. The LSF method (default) is recommended for use.



Parameters - use checkboxes to select parameters to be estimated and provide the initial guess

initial value	min	max	fit	parameter	initial value	min	max	fit	parameter
6.08541010470684	1.0	25.0	✓	c - clearence rate	0.08323290499286	0.0	5.0	✓	t0 - start of treatment
0.99862815075242	0.0	1.0	✓	ε	0.50290594116787	0.0	2.0	✓	δ

Least square fitter [✓ Cobyla fitter]

1000000 Maximum number of cobyla fitter iterations

70 patient weight (kg) 15000 extracellular body fluid

Time to cure: 35.34 days

epsilon = 0.9986281507524276 ,delta = 0.5029059411678758 ,c = 6.085410104706848 ,t0 = 0.08323290499286833

Error = 1.3614674038970256E-9

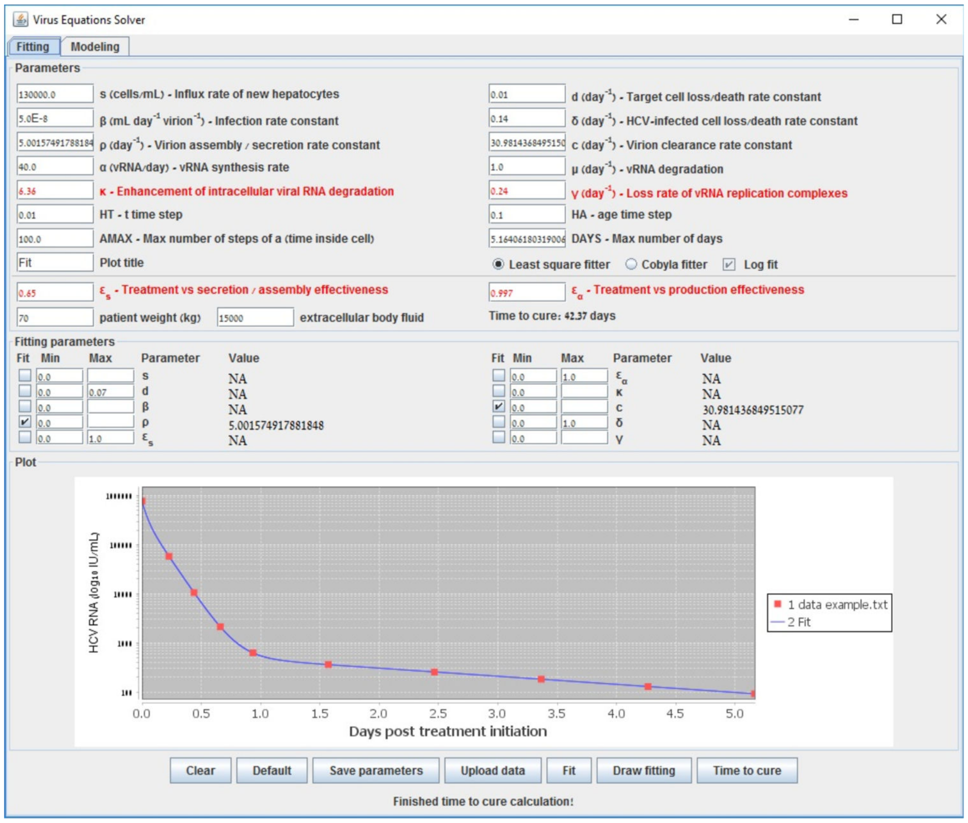
Figure A7.

Biphasic model fitting example with data taken from [36] of a patient who was treated with epclusa. In this particular case, COBYLA was selected instead of LSF and succeeded to yield a fit.

Appendix C.: Parameter Estimation in the Multiscale Model

Extending to the multiscale model for HCV dynamics, we perform parameter estimation on the model taken from [48]. As previously described, in our simulator, we solved the model equations using the Rosenbrock method [55] and performed parameter estimation after preparing the derivative equations using a full implementation of our developed Gauss–Newton (LSF) and COBYLA methods that are suitable to our application domain without reverting to canned methods.

To illustrate the tool we provide, we first show in Figure A8 the result of fitting to generated data points from the default values of the parameters $c\,\rho$, when starting away from their real values (with initial guesses of $\rho = 7.95$, $c = 22.5$) and selecting the LSF method. The predicted values after running LSF ($\rho = 5.0$, $c = 31.0$) are very close to their real values (error of $2.58 \cdot 10^{-6}$) and run-time was 40 s. In Figure A9, we show the result of the selecting the COBYLA method for the case as in the previous figure. The predicted values after running COBYLA are even slightly closer to their real values (error of $1.34 \cdot 10^{-8}$) for this particular case and run-time was 109 s. Thus, starting from initial guesses that are far from the real values can be handled, indicating the robustness of our methods. Run-times of our methods were significantly faster than a run-time of 1680 s (with even a larger error of around 0.5 showing much less robustness) when using the Levenberg–Marquardt method that was implemented in [57]. A webpage with user instructions is available at http://www.cs.bgu.ac.il/~dbarash/Churkin/SCE/Efficient/Parameter_Estimation/Multiscale.


Virus Equations Solver

Fitting | Modeling

Parameters

130000.0	s (cells/mL) - Influx rate of new hepatocytes	0.01	d (day^{-1}) - Target cell loss/death rate constant
5.0E-8	β (mL day^{-1} virion^{-1}) - Infection rate constant	0.14	δ (day^{-1}) - HCV-infected cell loss/death rate constant
5.00157491788184	ρ (day^{-1}) - Virion assembly / secretion rate constant	30.9814368495150	c (day^{-1}) - Virion clearance rate constant
40.0	α (vRNA/day) - vRNA synthesis rate	1.0	μ (day^{-1}) - vRNA degradation
6.36	κ - Enhancement of intracellular viral RNA degradation	0.24	γ (day^{-1}) - Loss rate of vRNA replication complexes
0.01	HT - t time step	0.1	HA - age time step
100.0	AMAX - Max number of steps of a (time inside cell)	5.16406180319006	DAYS - Max number of days
Fit	Plot title		[✓ Least square fitter] Cobyla fitter [✓ Log fit]
0.65	ε_s - Treatment vs secretion / assembly effectiveness	0.997	ε_α - Treatment vs production effectiveness
70	patient weight (kg) 15000 extracellular body fluid		Time to cure: 42.37 days

Fitting parameters

Fit	Min	Max	Parameter	Value	Fit	Min	Max	Parameter	Value
	0.0		s	NA		0.0	1.0	ε_α	NA
	0.0	0.07	d	NA		0.0		κ	NA
	0.0		β	NA	✓	0.0		c	30.981436849515077
✓	0.0		ρ	5.001574917881848		0.0	1.0	δ	NA
	0.0	1.0	ε_s	NA		0.0		γ	NA

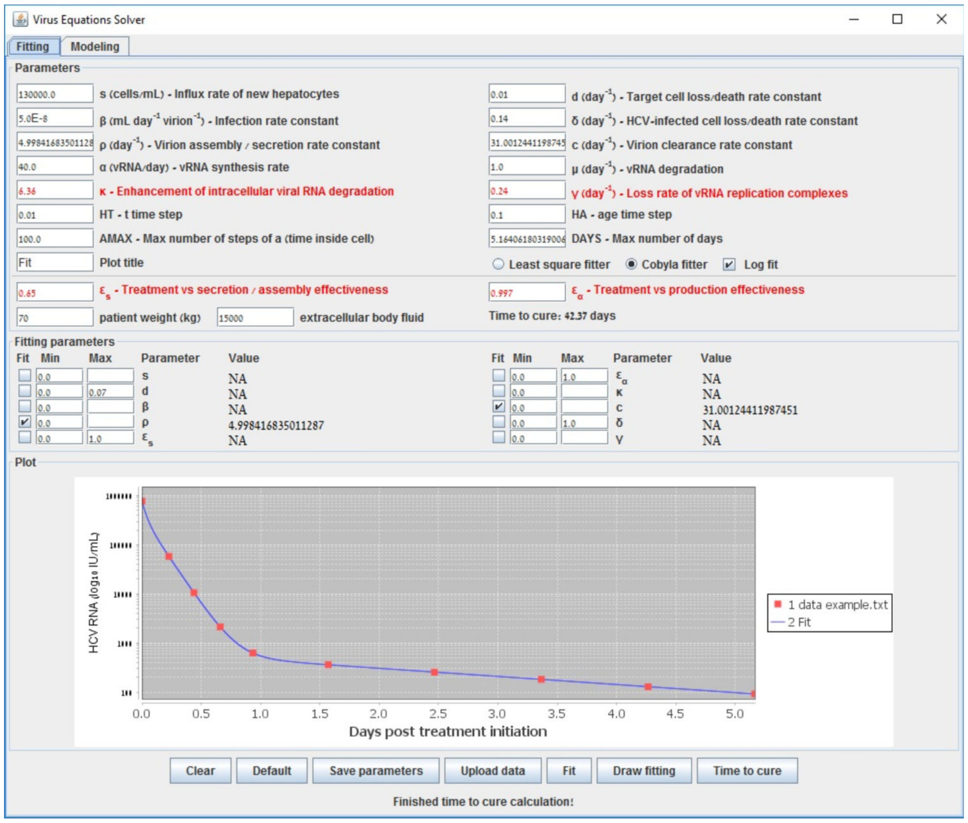
Plot

Clear | Default | Save parameters | Upload data | Fit | Draw fitting | Time to cure

Finished time to cure calculation!

Figure A8.

Fitting the parameters c and ρ of the multiscale model to generated data points using the LSF method.



Parameters

130000.0	s (cells/mL) - Influx rate of new hepatocytes	0.01	d (day^{-1}) - Target cell loss/death rate constant
5.0E-8	β (mL day^{-1} virion^{-1}) - Infection rate constant	0.14	δ (day^{-1}) - HCV-infected cell loss/death rate constant
4.99841683501128	ρ (day^{-1}) - Virion assembly / secretion rate constant	31.0012441198745	c (day^{-1}) - Virion clearance rate constant
40.0	α (vRNA/day) - vRNA synthesis rate	1.0	μ (day^{-1}) - vRNA degradation
6.36	κ - Enhancement of intracellular viral RNA degradation	0.24	γ (day^{-1}) - Loss rate of vRNA replication complexes
0.01	HT - t time step	0.1	HA - age time step
100.0	AMAX - Max number of steps of a (time inside cell)	5.16406180319006	DAYS - Max number of days
Fit	Plot title		○ Least square fitter [✓ ● Cobyla fitter] [✓ Log fit]
0.65	ε_s - Treatment vs secretion / assembly effectiveness	0.997	ε_α - Treatment vs production effectiveness
70	patient weight (kg) 15000 extracellular body fluid		Time to cure: 42.37 days

Fitting parameters

Fit	Min	Max	Parameter	Value	Fit	Min	Max	Parameter	Value
	0.0		s	NA		0.0	1.0	ε_α	NA
	0.0	0.07	d	NA		0.0		κ	NA
	0.0		β	NA	✓	0.0		c	31.00124411987451
✓	0.0		ρ	4.998416835011287		0.0	1.0	δ	NA
	0.0	1.0	ε_s	NA		0.0		γ	NA

Plot

Finished time to cure calculation!

Figure A9.

Fitting the parameters c and ρ of the multiscale model to generated data points using the COBYLA method.

References

1. World Health Organization. Global Hepatitis Report 2017: Web Annex A: Estimations of Worldwide Prevalence of Chronic Hepatitis B Virus Infection: A Systematic Review of Data Published between 1965 and 2017; Technical Report; World Health Organization: Geneva, Switzerland, 2018.
2. Gilman C; Heller T; Koh C Chronic hepatitis delta: A state-of-the-art review and new therapies. World J. Gastroenterol 2019, 25, 4580. [PubMed: 31528088]
3. Blach S; Zeuzem S; Manns M; Altraif I; Duberg AS; Muljono DH; Waked I; Alavian SM; Lee MH; Negro F; et al. Global prevalence and genotype distribution of hepatitis C virus infection in 2015: A modelling study. Lancet Gastroenterol. Hepatol 2017, 2, 161–176. [PubMed: 28404132]
4. Stanaway JD; Flaxman AD; Naghavi M; Fitzmaurice C; Vos T; Abubakar I; Abu-Raddad LJ; Assadi R; Bhala N; Cowie B; et al. The global burden of viral hepatitis from 1990 to 2013: Findings from the Global Burden of Disease Study 2013. Lancet 2016, 388, 1081–1088. [PubMed: 27394647]
5. Foreman KJ; Marquez N; Dolgert A; Fukutaki K; Fullman N; McGaughey M; Pletcher MA; Smith AE; Tang K; Yuan CW; et al. Forecasting life expectancy, years of life lost, and all-cause and cause-specific mortality for 250 causes of death: Reference and alternative scenarios for 2016–2040 for 195 countries and territories. Lancet 2018, 392, 2052–2090. [PubMed: 30340847]
6. Ciupe SM Modeling the dynamics of hepatitis B infection, immunity, and drug therapy. Immunol. Rev 2018, 285, 38–54. [PubMed: 30129194]
7. Means S; Ali MA; Ho H; Heffernan J Mathematical Modeling for Hepatitis B Virus: Would Spatial Effects Play a Role and How to Model It? Front. Physiol 2020, 11, 146. [PubMed: 32158403]

8. Dahari H; Major M; Zhang X; Mihalik K; Rice CM; Perelson AS; Feinstone SM; Neumann AU Mathematical modeling of primary hepatitis C infection: Noncytolytic clearance and early blockage of virion production. Gastroenterology 2005, 128, 1056–1066. [PubMed: 15825086]

9. Dahari H; Layden-Almer JE; Kallwitz E; Ribeiro RM; Cotler SJ; Layden TJ; Perelson AS A mathematical model of hepatitis C virus dynamics in patients with high baseline viral loads or advanced liver disease. Gastroenterology 2009, 136, 1402–1409. [PubMed: 19208338]

10. Goyal A; Murray JM Dynamics of in vivo hepatitis D virus infection. J. Theor. Biol 2016, 398, 9–19. [PubMed: 27012516]

11. Goyal A; Ribeiro RM; Perelson AS The role of infected cell proliferation in the clearance of acute HBV infection in humans. Viruses 2017, 9, 350.

12. Neumann AU; Phillips S; Levine I; Ijaz S; Dahari H; Eren R; Dagan S; Naoumov NV Novel mechanism of antibodies to hepatitis B virus in blocking viral particle release from cells. Hepatology 2010, 52, 875–885. [PubMed: 20593455]

13. Dahari H; de Araujo ESA; Haagmans BL; Layden TJ; Cotler SJ; Barone AA; Neumann AU Pharmacodynamics of PEG-IFN-α−2a in HIV/HCV co-infected patients: Implications for treatment outcomes. J. Hepatol 2010, 53, 460–467. [PubMed: 20561702]

14. Dahari H; Ribeiro RM; Perelson AS Triphasic decline of hepatitis C virus RNA during antiviral therapy. Hepatology 2007, 46, 16–21. [PubMed: 17596864]

15. Dahari H; Lo A; Ribeiro RM; Perelson AS Modeling hepatitis C virus dynamics: Liver regeneration and critical drug efficacy. J. Theor. Biol 2007, 247, 371–381. [PubMed: 17451750]

16. Dahari H; Shudo E; Ribeiro RM; Perelson AS Modeling complex decay profiles of hepatitis B virus during antiviral therapy. Hepatology 2009, 49, 32–38. [PubMed: 19065674]

17. Koh C; Dubey P; Han MAT; Walter PJ; Garraffo HM; Surana P; Southall NT; Borochov N; Uprichard SL; Cotler SJ; et al. A randomized, proof-of-concept clinical trial on repurposing chlorcyclizine for the treatment of chronic hepatitis C. Antivir. Res 2019, 163, 149–155. [PubMed: 30711416]

18. Dubey P; Koh C; Surana P; Uprichard SL; Han MAT; Fryzek N; Kapuria D; Etzion O; Takyar VK; Rotman Y; et al. Modeling hepatitis delta virus dynamics during ritonavir boosted lonafarnib treatment-the LOWR HDV-3 study. Hepatology 2017, 66, 21A.

19. Pawlotsky JM; Dahari H; Neumann AU; Hezode C; Germanidis G; Lonjon I; Castera L; Dhumeaux D Antiviral action of ribavirin in chronic hepatitis C. Gastroenterology 2004, 126, 703–714. [PubMed: 14988824]

20. Neumann AU; Lam NP; Dahari H; Davidian M; Wiley TE; Mika BP; Perelson AS; Layden TJ Differences in viral dynamics between genotypes 1 and 2 of hepatitis C virus. J. Infect. Dis 2000, 182, 28–35. [PubMed: 10882578]

21. Dahari H; Shudo E; Ribeiro RM; Perelson AS Mathematical modeling of HCV infection and treatment. Methods Mol. Biol 2009, 510, 439–453. [PubMed: 19009281]

22. DebRoy S; Hiraga N; Imamura M; Hayes CN; Akamatsu S; Canini L; Perelson AS; Pohl RT; Persiani S; Uprichard SL; et al. hepatitis C virus dynamics and cellular gene expression in uPA-SCID chimeric mice with humanized livers during intravenous silibinin monotherapy. J. Viral Hepat 2016, 23, 708–717. [PubMed: 27272497]

23. Canini L; DebRoy S; Mariño Z; Conway JM; Crespo G; Navasa M; D'Amato M; Ferenci P; Cotler SJ; Forns X; et al. Severity of liver disease affects HCV kinetics in patients treated with intravenous silibinin monotherapy. Antivir. Ther 2015, 20, 149. [PubMed: 24912382]

24. Goyal A; Lurie Y; Meissner EG; Major M; Sansone N; Uprichard SL; Cotler SJ; Dahari H Modeling HCV cure after an ultra-short duration of therapy with direct acting agents. Antivir. Res 2017, 144, 281–285. [PubMed: 28673800]

25. Dahari H; Shteingart S; Gafanovich I; Cotler SJ; D'Amato M; Pohl RT; Weiss G; Ashkenazi YJ; Tichler T; Goldin E; et al. Sustained virological response with intravenous silibinin: individualized IFN-free therapy via real-time modelling of HCV kinetics. Liver Int. 2015, 35, 289–294. [PubMed: 25251042]

26. Neumann AU; Lam NP; Dahari H; Gretch DR; Wiley TE; Layden TJ; Perelson AS Hepatitis C viral dynamics in vivo and the antiviral efficacy of interferon-α therapy. Science 1998, 282, 103–107. [PubMed: 9756471]

27. Guedj J; Dahari H; Pohl RT; Ferenci P; Perelson AS Understanding silibinin’s modes of action against HCV using viral kinetic modeling. J. Hepatol 2012, 56, 1019–1024. [PubMed: 22245888]

28. Lewin SR; Ribeiro RM; Walters T; Lau GK; Bowden S; Locarnini S; Perelson AS Analysis of hepatitis B viral load decline under potent therapy: complex decay profiles observed. Hepatology 2001, 34, 1012–1020. [PubMed: 11679973]

29. Ribeiro RM; Germanidis G; Powers KA; Pellegrin B; Nikolaidis P; Perelson AS; Pawlotsky JM hepatitis B virus kinetics under antiviral therapy sheds light on differences in hepatitis Be antigen positive and negative infections. J. Infect. Dis 2010, 202, 1309–1318. [PubMed: 20874517]

30. Nowak MA; Bonhoeffer S; Hill AM; Boehme R; Thomas HC; McDade H Viral dynamics in hepatitis B virus infection. Proc. Natl. Acad. Sci. USA 1996, 93, 4398–4402. [PubMed: 8633078]

31. Tsiang M; Rooney JF; Toole JJ; Gibbs CS Biphasic clearance kinetics of hepatitis B virus from patients during adefovir dipivoxil therapy. Hepatology 1999, 29, 1863–1869. [PubMed: 10347131]

32. Canini L; Koh C; Cotler SJ; Uprichard SL; Winters MA; Han MAT; Kleiner DE; Idilman R; Yurdaydin C; Glenn JS; et al. Pharmacokinetics and pharmacodynamics modeling of lonafarnib in patients with chronic hepatitis delta virus infection. Hepatol. Commun 2017, 1, 288–292. [PubMed: 29404459]

33. Koh C; Canini L; Dahari H; Zhao X; Uprichard SL; Haynes-Williams V; Winters MA; Subramanya G; Cooper SL; Pinto P; et al. Oral prenylation inhibition with lonafarnib in chronic hepatitis D infection: A proof-of-concept randomised, double-blind, placebo-controlled phase 2A trial. Lancet Infect. Dis 2015, 15, 1167–1174. [PubMed: 26189433]

34. Guedj J; Rotman Y; Cotler SJ; Koh C; Schmid P; Albrecht J; Haynes-Williams V; Liang TJ; Hoofnagle JH; Heller T; et al. Understanding early serum hepatitis D virus and hepatitis B surface antigen kinetics during pegylated interferon-alpha therapy via mathematical modeling. Hepatology 2014, 60, 1902–1910. [PubMed: 25098971]

35. Shekhtman L; Cotler SJ; Hershkovich L; Uprichard SL; Bazinet M; Pantea V; Cebotarescu V; Cojuhari L; Jimbei P; Krawczyk A; et al. Modelling hepatitis D virus RNA and HBsAg dynamics during nucleic acid polymer monotherapy suggest rapid turnover of HBsAg. Sci. Rep 2020, 10, 1–7.

36. Etzion O; Dahari H; Yardeni D; Issachar A; Nevo-Shor A; Naftaly-Cohen M; Uprichard SL; Arbib OS; Munteanu D; Braun M; et al. Response-Guided Therapy with Direct-Acting Antivirals Shortens Treatment Duration in 50% of HCV Treated Patients. Hepatology 2018, 68, 1469A–1470A.

37. Dahari H; Canini L; Graw F; Uprichard SL; Araújo ES; Penaranda G; Coquet E; Chiche L; Riso A; Renou C; et al. HCV kinetic and modeling analyses indicate similar time to cure among sofosbuvir combination regimens with daclatasvir, simeprevir or ledipasvir. J. Hepatol 2016, 64, 1232–1239. [PubMed: 26907973]

38. Canini L; Imamura M; Kawakami Y; Uprichard SL; Cotler SJ; Dahari H; Chayama K HCV kinetic and modeling analyses project shorter durations to cure under combined therapy with daclatasvir and asunaprevir in chronic HCV-infected patients. PLoS ONE 2017, 12, e0187409. [PubMed: 29216198]

39. Gambato M; Canini L; Lens S; Graw F; Perpiñan E; Londoño MC; Uprichard SL; Mariño Z; Reverter E; Bartres C; et al. Early HCV viral kinetics under DAAs may optimize duration of therapy in patients with compensated cirrhosis. Liver Int. 2019, 39, 826–834. [PubMed: 30499631]

40. Sandmann L; Manns MP; Maasoumy B Utility of viral kinetics in HCV therapy—It is not over until it is over? Liver Int. 2019, 39, 815–817. [PubMed: 31020775]

41. Deng B; Lou S; Dubey P; Etzion O; Chayam K; Uprichard S; Sulkowski M; Cotler S; Dahari H Modeling time to cure after short-duration treatment for chronic HCV with daclatasvir, asunaprevir, beclabuvir and sofosbuvir: the FOURward study. J. Viral Hepat 2018, 25, 58.

42. Dahari H; Sainz B; Perelson AS; Uprichard SL Modeling subgenomic hepatitis C virus RNA kinetics during treatment with alpha interferon. J. Virol 2009, 83, 6383–6390. [PubMed: 19369346]

43. Dahari H; Ribeiro RM; Rice CM; Perelson AS Mathematical modeling of subgenomic hepatitis C virus replication in Huh-7 cells. J. Virol 2007, 81, 750–760. [PubMed: 17035310]

44. Murray JM; Goyal A In silico single cell dynamics of hepatitis B virus infection and clearance. J. Theor. Biol 2015, 366, 91–102. [PubMed: 25476731]

45. Murray JM; Wieland SF; Purcell RH; Chisari FV Dynamics of hepatitis B virus clearance in chimpanzees. Proc. Natl. Acad. Sci. USA 2005, 102, 17780–17785. [PubMed: 16306261]

46. Packer A; Forde J; Hews S; Kuang Y Mathematical models of the interrelated dynamics of hepatitis D and B. Math. Biosci 2014, 247, 38–46. [PubMed: 24513247]

47. Guedj J; Dahari H; Rong L; Sansone ND; Nettles RE; Cotler SJ; Layden TJ; Uprichard SL; Perelson AS 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 2013, 110, 3991–3996. [PubMed: 23431163]

48. Rong L; Guedj J; Dahari H; Coffield DJJ; Levi M; Smith P; Perelson AS Analysis of hepatitis C virus decline during treatment with the protease inhibitor danoprevir using a multiscale model. PLoS Comput. Biol 2013, 9, e1002959. [PubMed: 23516348]

49. Rong L; Perelson AS Mathematical analysis of multiscale models for hepatitis C virus dynamics under therapy with direct-acting antiviral agents. Math. Biosci 2013, 245, 22–30. [PubMed: 23684949]

50. Quintela BM; Conway JM; Hyman JM; Guedj J; dos Santos RW; Lobosco M; Perelson AS A New Age-Structured Multiscale Model of the Hepatitis C Virus Life-Cycle During Infection and Therapy with Direct-Acting Antiviral Agents. Front. Microbiol 2018, 9, 601. [PubMed: 29670586]

51. Guedj J; Neumann AU Understanding hepatitis C viral dynamics with direct-acting antiviral agents due to the interplay between intracellular replication and cellular infection dynamics. J. Theor. Biol 2010, 267, 330–340. [PubMed: 20831874]

52. Weickert J; ter Haar Romeny B; Viergever M Efficient and reliable schemes for nonlinear diffusion filtering. IEEE Trans. Imag. Proc 1998, 7, 398–410.

53. Barash D; Israeli M; Kimmel R An Accurate Operator Splitting Scheme for Nonlinear Diffusion Filtering. In Proceedings of the 3rd International Conference on ScaleSpace and Morphology, Vancouver, BC, Canada, 7–8 July 2001; pp. 281–289.

54. Barash D Nonlinear Diffusion Filtering on Extended Neighborhood. Appl. Num. Math 2005, 52, 1–11.

55. Reinharz V; Churkin A; Dahari H; Barash D A Robust and Efficient Numerical Method for RNA-mediated Viral Dynamics. Front. Appl. Math. Stat 2017, 3, 20. [PubMed: 30854378]

56. Reinharz V; Dahari H; Barash D Numerical schemes for solving and optimizing multiscale models with age of hepatitis C virus dynamics. Math. Biosci 2018, 300, 1–13. [PubMed: 29550297]

57. Reinharz V; Churkin A; Lewkiewicz S; Dahari H; Barash D A Parameter Estimation Method for Multiscale Models of hepatitis C Virus Dynamics. Bull. Math. Biol 2019, 81, 3675–3721. [PubMed: 31338739]

58. Levenberg K A method for the solution of certain nonlinear problems in least squares. Q. Appl. Math 1944, 2, 164–168.

59. Marquardt DW An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math 1963, 11, 431–441.

60. Kitagawa K; Nakaoka S; Asai Y; Watashi K; Iwami S A PDE Multiscale Model of hepatitis C virus infection can be transformed to a system of ODEs. J. Theor. Biol 2018, 448, 80–85. [PubMed: 29634960]

61. Powell MJD A View of Algorithms for Optimization Without Derivatives. Math. Today 2007, 43, 170–174.

62. Rohatgi A WebPlotDigitizer: Web Based Tool to Extract Data from Plots, Images, and Maps. V 4.1. 2018 Available online: https://automeris.io/WebPlotDigitizer (accessed on 22 August 2020).

63. Quarteroni A; Valli A Springer Series in Computational Mathematics, 1994; Springer: Berlin/Heidelberg, Germany, 1994.

64. Knabner P; Angermann L Numerical Methods for Elliptic and Parabolic Partial Differential Equations: An Applications-Oriented Introduction; Springer: Berlin/Heidelberg, Germany, 2004.

65. Powell MJD A Direct Search Optimization Method That Models Objective and Constraint Functions by Linear Interpolation In Advances in Optimization and Numerical Analysis.

Mathematics and its Applications; Gomez S, Hennart J, Eds.; Springer: Dordrecht, The Netherlands, 1994; Volume 275.

66. Bastian P; Heimann F; Marnach S Generic implementation of finite element methods in the distributed and unified numerics environment (DUNE). Kybernetika 2010, 46, 294–315.
67. Flemisch B; Darcis M; Erbertseder K; Faigle B; Lauser A; Mosthaf K; Müthing S; Nuske P; Tatomir A; Wolff M; et al. DuMux: DUNE for multi-{phase, component, scale, physics,… } flow and transport in porous media. Adv. Water Resour 2011, 34, 1102–1112.
68. Vogel A; Reiter S; Rupp M; Nägel A; Wittum G UG 4: A novel flexible software system for simulating PDE based models on high performance computers. Comput. Vis. Sci 2013, 16, 165–179.
69. Spendley W; Hext GR; Himsworth FR Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation. Technometrics 1962, 4, 441–461.
70. Nelder J; Mead R A Simplex Method for Function Minimization. Comput. J 1965, 7, 308–313.
71. Dasgupta S; Imamura M; Gorstein E; Nakahara T; Tsuge M; Churkin A; Yardeni D; Etzion O; Uprichard SL; Barash D; et al. Modeling-Based Response-Guided Glecaprevir-Pibrentasvir Therapy for Chronic Hepatitis C to Identify Patients for Ultrashort Treatment Duration. J. Infect. Dis 2020, jiaa219, doi:10.1093/infdis/jiaa219.
72. Gorstein E; Martinello M; Churkin A; Dasgupta S; Walsh K; Applegate T; Yardeni D; Etzion O; Uprichard SL; Barash D; et al. Modeling based response guided therapy in subjects with recent hepatitis C infection. Antivir. Res 2020, doi:10.1016/j.antiviral.2020.104862.

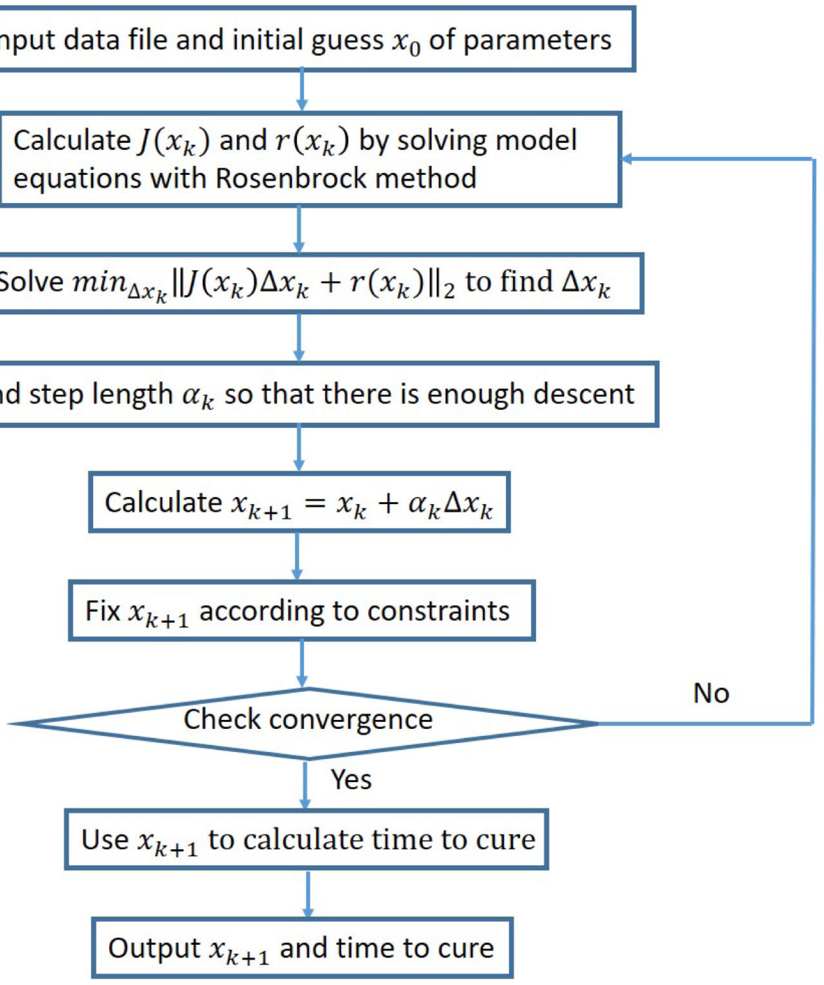
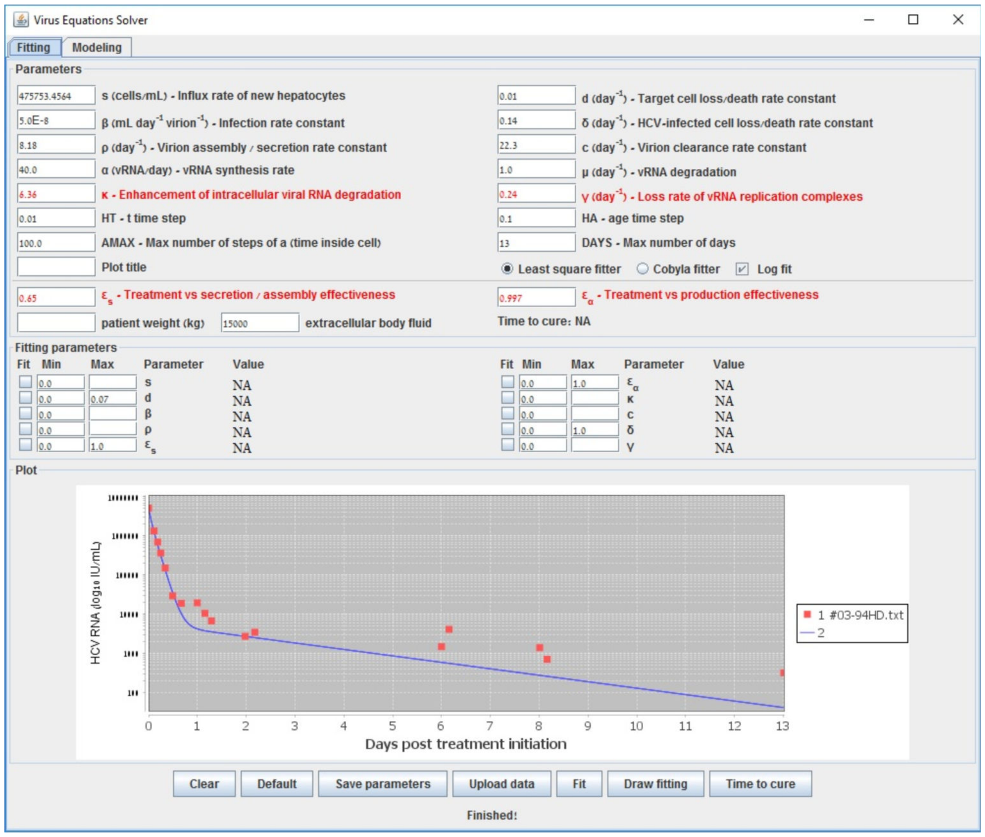

Input data file and initial guess x_0 of parameters
Calculate $J(x_k)$ and $r(x_k)$ by solving model equations with Rosenbrock method
Solve $min_{\Delta x_k} \|J(x_k)\Delta x_k + r(x_k)\|_2$ to find Δx_k
Find step length α_k so that there is enough descent
Calculate $x_{k+1} = x_k + \alpha_k \Delta x_k$
Fix x_{k+1} according to constraints
Check convergence
No
Yes
Use x_{k+1} to calculate time to cure
Output x_{k+1} and time to cure

Figure 1.
A flowchart of our constrained damped Gauss–Newton method.



Parameters

475753.4564	s (cells/mL) - Influx rate of new hepatocytes	0.01	d (day^{-1}) - Target cell loss/death rate constant
5.0E-8	β (mL day^{-1} $virion^{-1}$) - Infection rate constant	0.14	δ (day^{-1}) - HCV-infected cell loss/death rate constant
8.18	ρ (day^{-1}) - Virion assembly / secretion rate constant	22.3	c (day^{-1}) - Virion clearance rate constant
40.0	α (vRNA/day) - vRNA synthesis rate	1.0	μ (day^{-1}) - vRNA degradation
6.36	κ - Enhancement of intracellular viral RNA degradation	0.24	γ (day^{-1}) - Loss rate of vRNA replication complexes
0.01	HT - t time step	0.1	HA - age time step
100.0	AMAX - Max number of steps of a (time inside cell)	13	DAYS - Max number of days
	Plot title		[✓ Least square fitter] Cobyla fitter [✓ Log fit]
0.65	ε_s - Treatment vs secretion / assembly effectiveness	0.997	ε_α - Treatment vs production effectiveness
	patient weight (kg) 15000 extracellular body fluid		Time to cure: NA

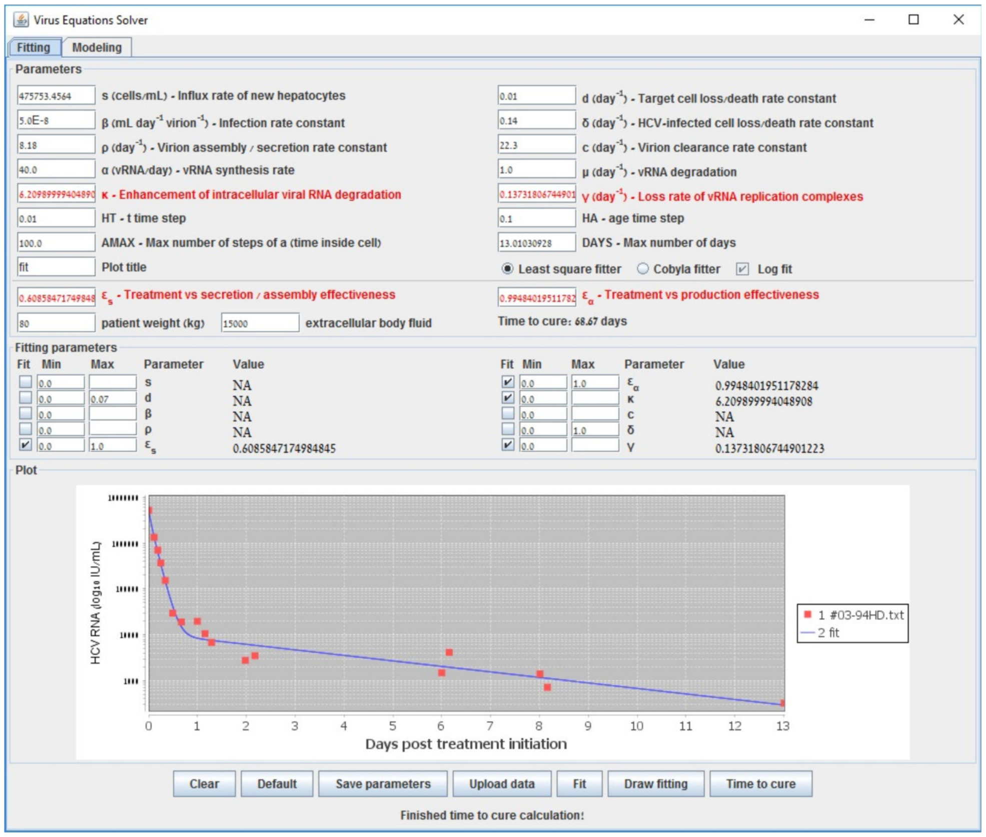
Fitting parameters

Fit	Min	Max	Parameter	Value	Fit	Min	Max	Parameter	Value
☐	0.0		s	NA	☐	0.0	1.0	ε_α	NA
☐	0.0	0.07	d	NA	☐	0.0		κ	NA
☐	0.0		β	NA	☐	0.0		c	NA
☐	0.0		ρ	NA	☐	0.0	1.0	δ	NA
☐	0.0	1.0	ε_s	NA	☐	0.0		γ	NA

Finished!

Figure 2.

Start fit that emanates from data of a patient reported in [48]. The fitting curve corresponds to default parameters before fitting with our methods. The multiscale model is used.



Parameters

475753.4564	s (cells/mL) - Influx rate of new hepatocytes	0.01	d (day^{-1}) - Target cell loss/death rate constant
5.0E-8	β (mL day^{-1} virion^{-1}) - Infection rate constant	0.14	δ (day^{-1}) - HCV-infected cell loss/death rate constant
8.18	ρ (day^{-1}) - Virion assembly / secretion rate constant	22.3	c (day^{-1}) - Virion clearance rate constant
40.0	α (vRNA/day) - vRNA synthesis rate	1.0	μ (day^{-1}) - vRNA degradation
6.20989999404890	κ - Enhancement of intracellular viral RNA degradation	0.13731806744901	γ (day^{-1}) - Loss rate of vRNA replication complexes
0.01	HT - t time step	0.1	HA - age time step
100.0	AMAX - Max number of steps of a (time inside cell)	13.01030928	DAYS - Max number of days
fit	Plot title		[✓ Least square fitter] Cobyla fitter [✓ Log fit]
0.60858471749848	ε_s - Treatment vs secretion / assembly effectiveness	0.99484019511782	ε_α - Treatment vs production effectiveness
80	patient weight (kg) 15000 extracellular body fluid		Time to cure: 68.67 days

Fitting parameters

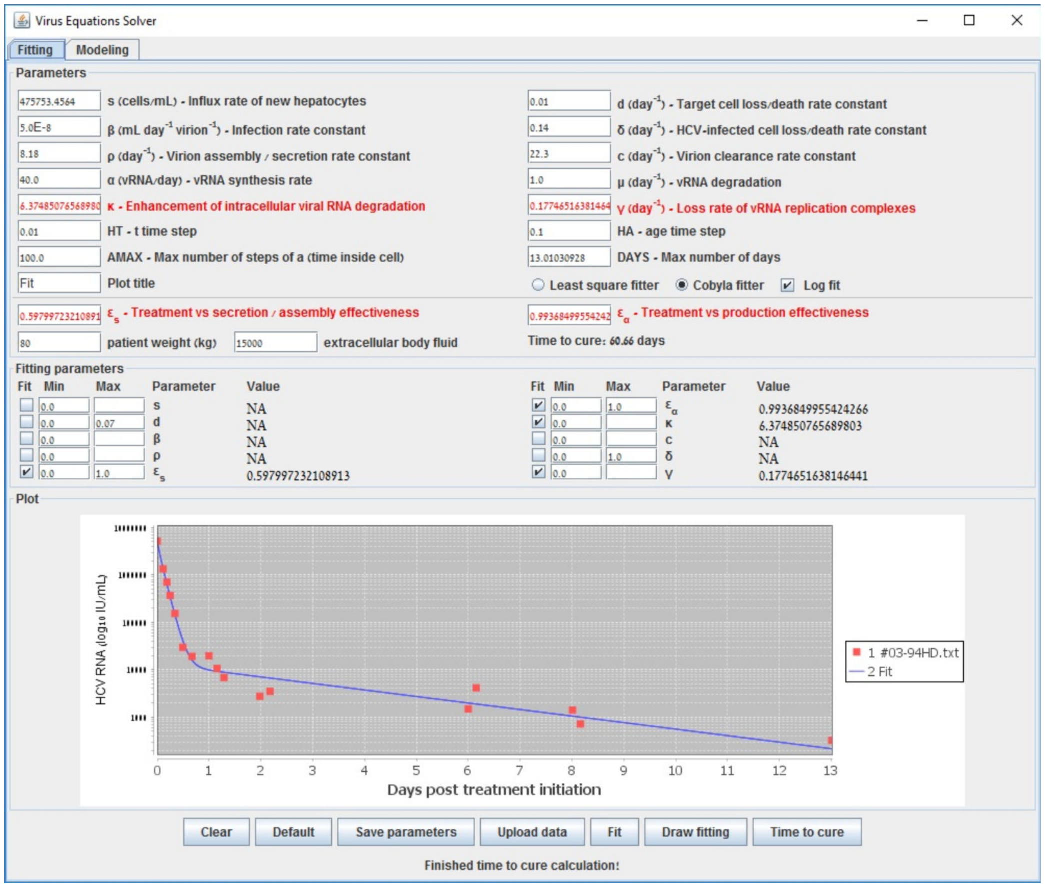
Fit	Min	Max	Parameter	Value	Fit	Min	Max	Parameter	Value
	0.0		s	NA	✓	0.0	1.0	ε_α	0.9948401951178284
	0.0	0.07	d	NA	✓	0.0		κ	6.209899994048908
	0.0		β	NA		0.0		c	NA
	0.0		ρ	NA		0.0	1.0	δ	NA
✓	0.0	1.0	ε_s	0.6085847174984845	✓	0.0		γ	0.13731806744901223

Plot

Finished time to cure calculation!

Figure 3.

End fit using Gauss–Newton (LSF) that emanates from data of a patient reported in [48].



Parameters

475753.4564	s (cells/mL) - Influx rate of new hepatocytes	0.01	d (day^{-1}) - Target cell loss/death rate constant
5.0E-8	β (mL day^{-1} virion^{-1}) - Infection rate constant	0.14	δ (day^{-1}) - HCV-infected cell loss/death rate constant
8.18	ρ (day^{-1}) - Virion assembly / secretion rate constant	22.3	c (day^{-1}) - Virion clearance rate constant
40.0	α (vRNA/day) - vRNA synthesis rate	1.0	μ (day^{-1}) - vRNA degradation
6.37485076568980	κ - Enhancement of intracellular viral RNA degradation	0.17746516381464	γ (day^{-1}) - Loss rate of vRNA replication complexes
0.01	HT - t time step	0.1	HA - age time step
100.0	AMAX - Max number of steps of a (time inside cell)	13.01030928	DAYS - Max number of days
Fit	Plot title		○ Least square fitter [✓ Cobyla fitter] ✓ Log fit
0.59799723210891	ε_s - Treatment vs secretion / assembly effectiveness	0.99368499554242	ε_α - Treatment vs production effectiveness
80	patient weight (kg) 15000 extracellular body fluid		Time to cure: 60.66 days

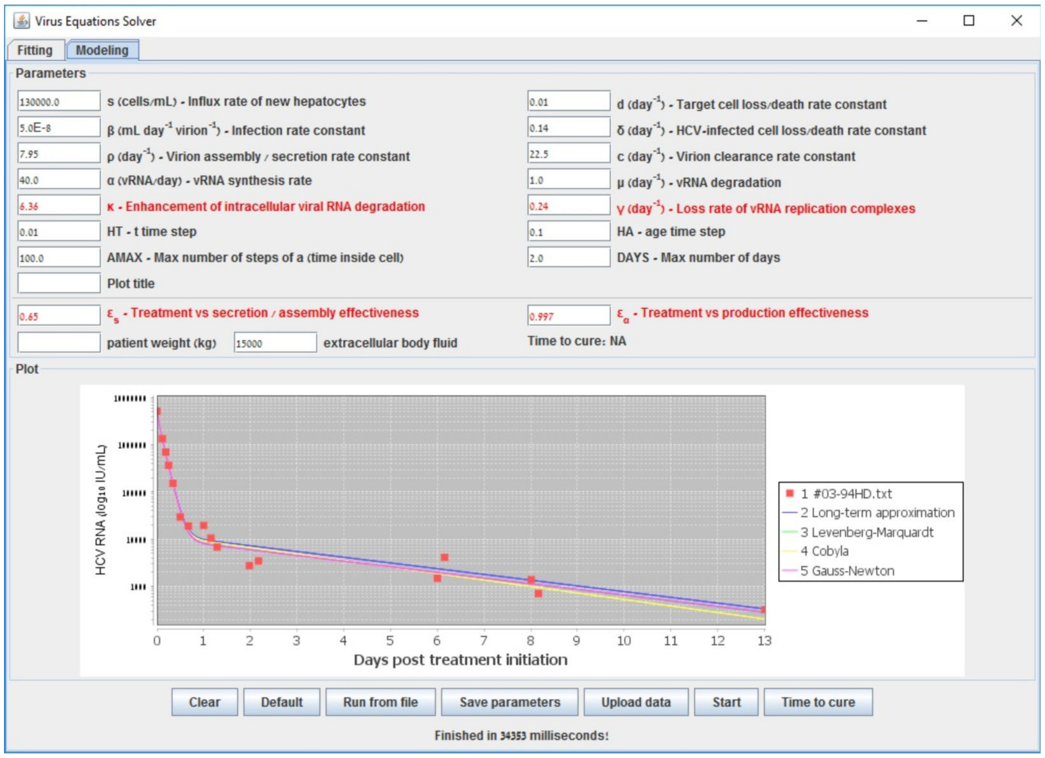
Fitting parameters

Fit	Min	Max	Parameter	Value	Fit	Min	Max	Parameter	Value
	0.0		s	NA	✓	0.0	1.0	ε_α	0.9936849955424266
	0.0	0.07	d	NA	✓	0.0		κ	6.374850765689803
	0.0		β	NA		0.0		c	NA
	0.0		ρ	NA		0.0	1.0	δ	NA
✓	0.0	1.0	ε_s	0.597997232108913	✓	0.0		γ	0.1774651638146441

Finished time to cure calculation!

Figure 4.

End fit using COBLYA that emanates from data of a patient reported in [48].



130000.0	s (cells/mL) - Influx rate of new hepatocytes	0.01	d (day^{-1}) - Target cell loss/death rate constant
5.0E-8	β (mL day^{-1} $virion^{-1}$) - Infection rate constant	0.14	δ (day^{-1}) - HCV-infected cell loss/death rate constant
7.95	ρ (day^{-1}) - Virion assembly / secretion rate constant	22.5	c (day^{-1}) - Virion clearance rate constant
40.0	α (vRNA/day) - vRNA synthesis rate	1.0	μ (day^{-1}) - vRNA degradation
6.36	κ - Enhancement of intracellular viral RNA degradation	0.24	γ (day^{-1}) - Loss rate of vRNA replication complexes
0.01	HT - t time step	0.1	HA - age time step
100.0	AMAX - Max number of steps of a (time inside cell)	2.0	DAYS - Max number of days
	Plot title		
0.65	ε_s - Treatment vs secretion / assembly effectiveness	0.997	ε_α - Treatment vs production effectiveness
	patient weight (kg)	15000	extracellular body fluid

Time to cure: NA

Finished in 34353 milliseconds!

Figure 5.

Comparison between the line fits of different methods inside the simulator window for the retrieved data points of patient HD that was reported in [48].

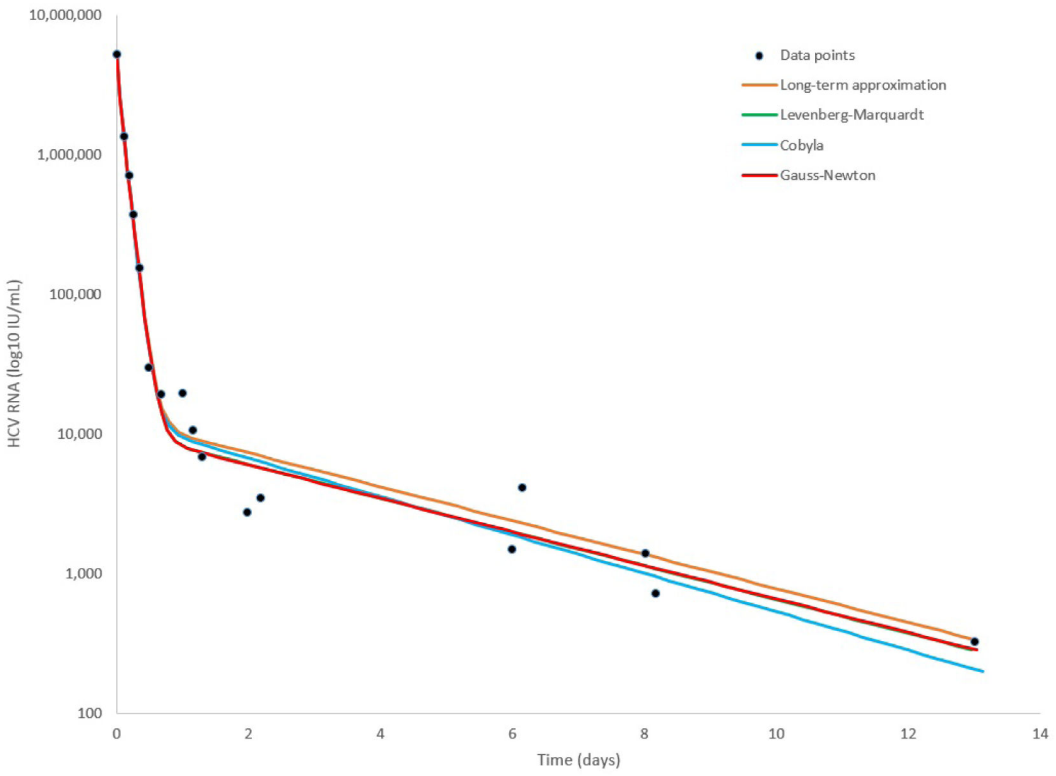


Figure 6.
Comparison between the line fits of different methods for the retrieved data points of patient HD that was reported in [48].

Table 1.

The 12 parameters of the model.

s (cells mL^{-1})	Influx rate of new hepatocytes
d (d^{-1})	Target cell loss/death rate constant
β (mL d^{-1} virion^{-1})	Infection rate constant
δ (d^{-1})	HCV-infected cell loss/death rate constant
ρ (d^{-1})	Virion assembly/secretion rate constant
c (d^{-1})	Virion clearance rate constant
α (vRNAd^{-1})	vRNA synthesis rate
μ (d^{-1})	vRNA degradation
κ	Enhancement of intracellular viral RNA degradation
γ (d^{-1})	Loss rate of vRNA replication complexes
ε_s	Treatment vs. secretion/assembly effectiveness
ε_α	Treatment vs. production effectiveness

Table 2.

Default parameters that are used herein. Parameter s comes from Equation (9), taking $\overline{V}$ as the max Virions value.

α	40 d^{-1}	β	5×10^{-8} mL d^{-1}
c	22.3 d^{-1}	δ	0.14 d^{-1}
μ	1 d^{-1}	d	0.01 d^{-1}
ρ	8.18 d^{-1}	s	$(\overline{\nabla}\beta c + dc)/(\beta N)$ cells /mL

Table 3.

Values of the parameters when fitted to the patient digitized data. The rightmost column has the values when the retrieved data points are fitted to the long-term approximation as in [57]. The left columns contain the fitted parameter values by our efficient methods. Except for the rightmost column, all methods are combined with the Rosenbrock numerical scheme. The fixed parameters have the values shown in Table 2. Run-time comparison is reported in seconds in the last row.

	Gauss-Newton (LSF)	COBYLA	Levenberg-Marquardt	Long-Term
ε_s	0.609	0.598	0.602	0.600
ε_a	0.995	0.994	0.995	0.994
κ	6.210	6.375	6.219	6.160
γ (d^{-1})	0.137	0.177	0.139	0.140
accuracy (sum error^2)	0.538	0.582	0.538	0.587
run-time (s)	194	3698	70118	<1