Journal of Computational Mathematics
Vol.30, No.1, 2012, 59–79.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1110-m11si12

CRITICAL ISSUES IN THE NUMERICAL TREATMENT OF THE PARAMETER ESTIMATION PROBLEMS IN IMMUNOLOGY*

Tatyana Luzyanina
Institute of Mathematical Problems in Biology, RAS, Pushchino, Moscow reg., Russia
Email: tatyana_luzyanina@impb.psn.ru
Gennady Bocharov
Institute of Numerical Mathematics, RAS, Moscow, Russia
Email: bocharov@inm.ras.ru

Abstract

A robust and reliable parameter estimation is a critical issue for modeling in immunology. We developed a computational methodology for analysis of the best-fit parameter estimates and the information-theoretic assessment of the mathematical models formulated with ODEs. The core element of the methodology is a robust evaluation of the first and second derivatives of the model solution with respect to the model parameter values. The critical issue of the reliable estimation of the derivatives was addressed in the context of inverse problems arising in mathematical immunology. To evaluate the first and second derivatives of the ODE solution with respect to parameters, we implemented the variational equations-, automatic differentiation and complex-step derivative approximation methods. A comprehensive analysis of these approaches to the derivative approximations is presented to understand their advantages and limitations.

Mathematics subject classification: 34K29, 92-08, 65K10.
Key words: Mathematical modeling in immunology, Parameter estimation, Constrained optimization.

1. Introduction

Mathematical immunology represents a rapidly growing field of applied mathematics. The key features of the immune system that make call for the application of mathematical modeling tools are: physical complexity, compartmental structure, non-linear response, threshold-type of regulation, memory or time-lag effects, inter-clonal competition and selection, redundancy [3]. Most mathematical models of immune responses are not obtained from first principles and therefore the model structure usually has no a priori proof of validity. The key elements of the data- and science-driven application of mathematical modeling to immunology are:

(i) more than one model may correspond to a particular phenomenon;

(ii) the computational techniques permit, given data of appropriate quality, to discriminate between rival mathematical models.

Given a number of candidate models, one needs for each model to determine a set of actual parameters that is in a well-defined sense optimal and to order the resulting set of optimally parameterized models to indicate which is most appropriate, given the data [3].

* Received February 9, 2011 / Revised version received May 29, 2011 / Accepted June 5, 2011 / Published online January 9, 2012 /

The last decade of research in immunology is characterized by a tremendous advance in the high-throughput experimental technologies yielding detailed information on the system state at various levels of resolution. This calls for a need to further advance the computational techniques for parameter estimation and a multiscale analysis of immunological phenomena. In our previous studies [4, 8–10, 14–16], we treated parameter estimation problems for models in immunology formulated by systems of ordinary, delay or partial differential equations, using different types of experimental data sets. Our experience led us to conclude that there is a set of rather common features of the parameter estimation problems in immunology:

(i) a lack of uniqueness of the solution to the inverse problem,

(ii) a high sensitivity to the errors in experimental data,

(iii) a poor practical identifiability of some of the model parameters.

The above difficulties, in part, result from an inconsistency between the information content of the data available and the ad hoc formulated parametric structures of the phenomenological mathematical models. The primary objective of this study is to propose an efficient computational technology for treating the parameter estimation problems appearing in mathematical immunology. Since the parameter estimation problem is essentially an optimization problem, the numerical accuracy and the computational cost of the evaluation of gradient, Jacobian- and Hessian matrices of the objective function represent the crucial factors. Therefore, an accurate and low-cost numerical approximation of the first and second derivatives of a model solution with respect to the model parameters are the key issues. So far, they have not been systematically addressed in the context of the inverse modeling in immunology.

In this study, we analyzed the existing approaches to derivative approximation: the conventional finite-differences method and the variational equations technique, which are broadly used since a long time, and the automatic differentiation (AD) and complex-step derivative approximation (CSD) methods, which came into focus of the numerical community only recently. We examined the efficiency and limitations of these methods in the context of the parameter estimation problems. The corresponding codes were implemented in Matlab. Note that during last years several AD packages have been introduced in Matlab [2]. The recently developed Matlab package PMAD [22] is meant for computing first derivatives of an analytic function by the CSD method. However, we are interested in developing a consistent and compact software which should be most efficient for solving the parameter estimation problems arising in immunology.

Most parameter estimation problems in mathematical immunology lead to minimization of a sum of squares of nonlinear functions subject to certain constraints on the estimated parameters. To solve the corresponding least-squares problems, we implemented the constrained Gauss-Newton method. To characterize the uncertainty in the parameter estimates, we used the variance-covariance and profile-likelihood-based methods. We show that the availability of the tools for computing accurate numerical derivatives allows one to perform a more thorough analysis of the parameter estimation problem, e.g. to examine the practical identifiability of the model parameters and their sensitivity to the data, to refine the model structure according to the data available.

In the next section we describe the key elements of the computational technology we propose for solving the parameter estimation problems arising in immunology. The numerical methods which can be used for approximating the derivatives of a model solution with respect to the model parameters are described in section 3. The comparative performance of the implemented methods is presented in section 4. In addition, we address the more fundamental modeling issues such as the information-theoretic model ranking and refinement. Some concluding remarks will be made in the final section.

2. Framework for Parameter Estimation

In this section we briefly outline the computational methodology for the parameter estimation and uncertainty analysis for models in immunology formulated as systems of nonlinear ordinary differential equations (ODEs). The ODE-based models represent the most numerous class of systems used in mathematical immunology (see for review [3]). The corresponding initial value problem (IVP) can be specified as follows: to solve for $\mathbf{y}(t)$

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}(t) = \mathbf{f}(t, \mathbf{y}(t), \mathbf{p}), \quad t \geq 0, \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{y} \in \mathbb{R}^n, \quad \mathbf{p} \in \mathbb{R}^{n_p}, \tag{2.1}$$

where $\mathbf{y}_0$ is a vector of initial data and $\mathbf{p}$ is a vector of the model parameters to be estimated. To solve numerically the IVP one can use one of the solvers from the Matlab ODE Suite. Recently, a new code, `BV78`, was developed to solve ODEs by the Runge-Kutta method based on (7,8) pair [23]. Notice that the new vectorization facility of `BV78` speeds up the time integration significantly if the user codes the function $\mathbf{f}$ in such a matrix form that allows to evaluate $\mathbf{f}$ for k arguments (needed in the Runge-Kutta method used) simultaneously, i.e., if $\mathbf{f}$ is coded as an n-by-k matrix with $\mathbf{y} \in \mathbb{R}^{n\times k}$.

2.1. Maximum likelihood approach

To estimate the model parameters, one looks for a vector of best-fit parameters $\mathbf{p}^* \in \mathbb{R}^{n_p}$, for which the model solution $\mathbf{y}(t)$ of (2.1) fits the data in an optimal way. The maximum likelihood (ML) approach provides a general framework for optimal parameter estimation, uncertainty analysis and information-theoretic ranking of the parameterized models. The ML formulation allows one to compute the model parameters by maximizing the likelihood that the data did arise from the model. To implement the maximum likelihood approach, we assume that (i) the observational errors, i.e. the residuals defined as a difference between observed and model-predicted values, are normally distributed; (ii) the errors in observations at successive times are independent; (iii) the errors in the components of the state vector are independent; (iv) the variance of observation errors (σ^2) is the same for all the state variables and observation times. For a more general treatment we refer to [5]. Under these conditions, the log-likelihood function specifying the probability of observing the given data set is given by

$$\ln(\mathcal{L}(\mathbf{p};\sigma)) = -0.5\Big(n_d \ln(2\pi) + n_d \ln(\sigma^2) + \sigma^{-2}\Phi(\mathbf{p})\Big),$$
$$\text{where } \Phi(\mathbf{p}) = \sum_{j=1}^{n_d} r_j^2(\mathbf{p}) = ||\mathbf{r}(\mathbf{p})||_2^2.$$

Here n_d is the number of measurements, $\Phi(\mathbf{p})$ is an ordinary least-squares function, $r_j : \mathbb{R}^{n_p} \to \mathbb{R}$, usually called a residual, is, in general, a twice continuously differentiable function, and we assume that $n_d \geq n_p$, see [4, 5] for further details.

The problem of maximizing the likelihood function is equivalent to that of minimizing $\Phi(\mathbf{p})$, provided that σ^2 is assigned the value ${\sigma^*}^2 = \Phi(\mathbf{p}^*)/n_d$, which follows from the optimality condition $\partial(\ln(\mathcal{L}(\mathbf{p}^*;\sigma)))/\partial\sigma^2 = 0$. Here $\mathbf{p}^* = \arg\min_{\mathbf{p}} \Phi(\mathbf{p})$. If constraints on the model parameters are imposed, one needs to deal with a constrained nonlinear least-squares problem

$$\min_{\mathbf{p}\in\mathbb{R}^{n_p}} \Phi(\mathbf{p}) \quad \text{subject to} \quad c_j(\mathbf{p}) \geq 0, \quad j = 1, \cdots, n_c, \tag{2.2}$$

where $c_j(\mathbf{p})$ is a twice continuously differentiable function, $\mathbf{c}(\mathbf{p}) \in \mathbb{R}^{n_c}$.

2.2. Constrained Gauss-Newton method

To solve numerically the constrained minimization problem (2.2), we implemented the constrained Gauss-Newton method in the code GN_NLS. The Gauss-Newton method is the simplest of the methods for minimizing the nonlinear objective function $\Phi(\mathbf{p})$ that exploits the structure in the gradient $\nabla\Phi(\mathbf{p})$ and the Hessian $\nabla^2\Phi(\mathbf{p})$. It can be viewed as a modified Newton's method with line search. Below we describe the main issues of the implemented constrained Gauss-Newton method.

Let $\tilde{\mathbf{p}} \in \mathbb{R}^{n_p}$ be an approximation to the solution of problem (2.2). One term of the Taylor expansion of the residual r_i and the constrain c_j at $\tilde{\mathbf{p}}$ reads

$$\begin{aligned} r_i(\tilde{\mathbf{p}} + \mathbf{d}) &\approx r_i(\tilde{\mathbf{p}}) + \Big(\frac{\partial r_i(\tilde{\mathbf{p}})}{\partial p_1}, \cdots, \frac{\partial r_i(\tilde{\mathbf{p}})}{\partial p_{n_p}}\Big)\mathbf{d}, \\ c_j(\tilde{\mathbf{p}} + \mathbf{d}) &\approx c_j(\tilde{\mathbf{p}}) + \Big(\frac{\partial c_j(\tilde{\mathbf{p}})}{\partial p_1}, \cdots, \frac{\partial c_j(\tilde{\mathbf{p}})}{\partial p_{n_p}}\Big)\mathbf{d}. \end{aligned}$$

Hence the approximating linear problem at a point $\mathbf{p}_k$ can be written as

$$\min_{\mathbf{d}\in\mathbb{R}^{n_p}} ||J(\mathbf{p}_k)\mathbf{d} + \mathbf{r}(\mathbf{p}_k)||_2^2 \quad \text{subject to} \quad Q(\mathbf{p}_k)\mathbf{d} + \mathbf{c}(\mathbf{p}_k) \geq 0, \tag{2.3}$$

where the Jacobian matrices $J(\mathbf{p}_k) \in \mathbb{R}^{n_d \times n_p}$ and $Q(\mathbf{p}_k) \in \mathbb{R}^{n_c \times n_p}$ are defined as

$$J_{i,j}(\mathbf{p}_k) = \frac{\partial r_i(\mathbf{p}_k)}{\partial p_j}, \quad Q_{l,j}(\mathbf{p}_k) = \frac{\partial c_l(\mathbf{p}_k)}{\partial p_j},$$
$$i = 1, \cdots, n_d, \quad l = 1, \cdots, n_c, \quad j = 1, \cdots, n_p.$$

While in the unconstrained Gauss-Newton method the search direction $\mathbf{d}_k$ at a point $\mathbf{p}_k$ is computed as

$$\mathbf{d}_k = -\Big(J(\mathbf{p}_k)^T J(\mathbf{p}_k)\Big)^{-1} J(\mathbf{p}_k)^T \mathbf{r}(\mathbf{p}_k), \tag{2.4}$$

in the constrained Gauss-Newton method the search direction $\mathbf{d}_k$ is computed as a minimizer of the linear approximating problem at a point $\mathbf{p}_k$

$$\mathbf{d}_k = \arg\min_{\mathbf{d}\in\mathbb{R}^{n_p}} \|J(\mathbf{p}_k)\mathbf{d} + \mathbf{r}(\mathbf{p}_k)\|_2^2 \quad \text{subject to} \quad Q(\mathbf{p}_k)\mathbf{d} + \mathbf{c}(\mathbf{p}_k) \geq 0. \tag{2.5}$$

Problem (2.5) is a quadratic programming (QP) problem. In Matlab a QP problem of the form

$$\min_{\mathbf{d}} \frac{1}{2}\mathbf{d}^T B\mathbf{d} + \mathbf{g}^T\mathbf{d} \quad \text{subject to} \quad A\mathbf{d} \leq \mathbf{b} \tag{2.6}$$

can be solved by the code quadprog. In our case,

$$B = J^T J \in \mathbb{R}^{n_p \times n_p}, \quad \mathbf{g} = J^T\mathbf{r} \in \mathbb{R}^{n_p \times 1}, \quad A = -Q \in \mathbb{R}^{n_c \times n_p}, \quad \mathbf{b} = \mathbf{c} \in \mathbb{R}^{n_c \times 1}. \tag{2.7}$$

As soon as a search direction $\mathbf{d}_k$ is computed, we have to decide how far to move along that direction. For this, we implemented the line search algorithm for the strong Wolfe conditions [19] (Chapter 3, Algorithms 3.5 and 3.6). Namely, we compute

$$\mathbf{p}_{k+1} = \mathbf{p}_k + s_k \mathbf{d}_k,$$

where $s_k \in \mathbb{R}^+$ is the step length which should satisfy the strong Wolfe conditions

$$\Phi(\mathbf{p}_k + s_k\mathbf{d}_k) \le \Phi(\mathbf{p}_k) + \alpha_1 s_k \nabla\Phi(\mathbf{p}_k)^T \mathbf{d}_k, \tag{2.8a}$$
$$|\nabla\Phi(\mathbf{p}_k + s_k\mathbf{d}_k)^T \mathbf{d}_k| \le \alpha_2 |\nabla\Phi_k^T \mathbf{d}_k|, \quad 0 < \alpha_1 < \alpha_2 < 1. \tag{2.8b}$$

2.3. Computing confidence intervals for the estimated parameters

To estimate the confidence in the identified parameters we implemented two approaches: the variance-covariance matrix based technique and the profile-likelihood-based method.

2.3.1. Variance-covariance analysis

The variance-covariance method is based upon a parabolic approximation of the objective function around the best-fit parameter estimate $\mathbf{p}^*$ [6]. The $100 \cdot \theta\%$ confidence interval (CI) for the parameter of interest, e.g. for p_k, is approximated by the standard interval

$$CI_{p_k} = [p_k^* - \sigma_{p_k} w(\theta, n_f),\ p_k^* + \sigma_{p_k} w(\theta, n_f)], \quad k = 1, \cdots, n_p, \tag{2.9}$$

where p_k^* is the best-fit parameter estimate, σ_{p_k} is the standard deviation for p_k, $w(\theta, n_f)$ is the $100 \cdot \theta$ percentage point of the Student's t-distribution with $n_f = n_d - n_p$ degrees of freedom. An estimate of the standard deviation of p_k is given by the corresponding diagonal element $\Xi_{k,k}$ of the covariance matrix Ξ,

$$\sigma_{p_k} = \sqrt{\Xi_{k,k}(\mathbf{p}^*)}, \quad \Xi(\mathbf{p}^*) = \frac{2\Phi(\mathbf{p}^*)}{n_f} H^{-1}(\mathbf{p}^*) \in \mathbb{R}^{n_p \times n_p}, \tag{2.10}$$

where $H \in \mathbb{R}^{n_p \times n_p}$ is the Hessian matrix with its (k, m)-th element equal to

$$H_{k,m}(\mathbf{p}) = \frac{\partial^2}{\partial p_k \partial p_m}\Phi(\mathbf{p}) = 2\sum_{j=1}^{n_d}\Big(\frac{\partial r_j(\mathbf{p})}{\partial p_k}\frac{\partial r_j(\mathbf{p})}{\partial p_m} + r_j(\mathbf{p})\frac{\partial^2 r_j(\mathbf{p})}{\partial p_k \partial p_m}\Big). \tag{2.11}$$

The accuracy of the confidence intervals computed by the variance-covariance method depends on how consistent the local parabolic approximation of the objective function in the vicinity of the best-fit parameters is. If the objective function is far from being parabolic, other methods need to be applied to assess the validity of the variance analysis. For many inverse problems in mathematical immunology the Hessian matrix H appears to be nearly singular and, hence, the CIs obtained are not reliable, cf. examples in section 4. In such cases, other methods should be used, in particular, those which do not involve the Hessian evaluation.

2.3.2. Profile-likelihood-based method

The profile-likelihood-based approach provides a method for computing the confidence intervals of the maximum likelihood parameter estimates by following 'a global' behavior of the objective function, cf. e.g. [25]. To compute the approximations to the $100 \cdot \theta\%$ CIs of the estimates, we proceed as follows. Let $\mathbf{p}^*$ be the vector of the computed best-fit values of the model parameters. For a parameter of interest, p_k^*, we search for the interval $[p_k^{\min},\ p_k^{\max}]$ of maximal width and containing p_k^* such that

$$|\ln(\mathcal{L}(\tilde{\mathbf{p}})) - \ln(\mathcal{L}(\mathbf{p}^*))| \le \frac{1}{2}\mathcal{X}^2_{1,\theta} \quad \text{whenever } p_k \in [p_k^{\min},\ p_k^{\max}]. \tag{2.12}$$

In (2.12), $\mathcal{L}(\mathbf{p}^*)$ stands for the likelihood function,

$$\begin{gathered}\ln(\mathcal{L}(\mathbf{p}^*;\sigma^*)) = -0.5\Big(n_d\ln(2\pi) + n_d\ln({\sigma^*}^2) + {\sigma^*}^{-2}\Phi(\mathbf{p}^*)\Big), \quad {\sigma^*}^2 = \frac{\Phi(\mathbf{p}^*)}{n_d}, \\ \mathcal{L}(\tilde{\mathbf{p}}) := \max_{\mathbf{p}\in S(p_k)} \mathcal{L}(\mathbf{p}), \quad S(p_k) := \Big\{[p_1, p_2, ..., p_{k-1}, p, p_{k+1}, ..., p_{n_p}]|_{p \text{ fixed}}\Big\},\end{gathered} \tag{2.13}$$

and $\mathcal{X}^2_{1,\theta}$ is the θ-th quantile of the $\mathcal{X}^2$-distribution for 1 degree of freedom.

Using the relationship between the maximum likelihood and the least-squares objective function, the expression (2.12) is equivalent to

$$\begin{gathered}|\ln(\Phi(\tilde{\mathbf{p}})) - \ln(\Phi(\mathbf{p}^*))| \le \frac{1}{n_d}\mathcal{X}^2_{1,\theta}, \\ \text{whenever } \ p_k \in [p_k^{\min},\ p_k^{\max}] \ \text{ and } \ \Phi(\tilde{\mathbf{p}}) = \min_{\mathbf{p}\in S(p_k)} \Phi(\mathbf{p}).\end{gathered} \tag{2.14}$$

Hence, to compute CI_{p_k} we search for $p_k^{\min}$ and $p_k^{\max}$ such that

$$\Phi(\tilde{\mathbf{p}}) \in [\Phi(\mathbf{p}^*),\ \Phi(\mathbf{p}^*)\mathrm{e}^{\mathcal{X}^2_{1,\theta}/n_d}] \quad \text{whenever } p_k \in [p_k^{\min},\ p_k^{\max}]. \tag{2.15}$$

In [25], an algorithm, based on the computation of the Hessian matrix H, is proposed to estimate CIs by the profile-likelihood-based method. Since H might be singular, we did not implement this algorithm. Instead, we suggest the following approach to estimate CIs by the profile-likelihood method. We fix $p_k = p_k^* + \delta$ with $|\delta|$ small compared to p_k^* and solve the minimization problem (2.2) with the code `GN_NLS` for the remaining $n_p - 1$ parameters. Then we increase (if $\delta > 0$) or decrease (if $\delta < 0$) p_k, solve (2.2), and repeat this procedure until p_k reaches a value for which inequality (2.14) is violated. In this way we estimate the upper (if $\delta > 0$) or the low (if $\delta < 0$) boundary of CI_{p_k}. Although this approach is rather time consuming, it is robust and it allows one to analyze the sensitivity of the objective $\Phi(\mathbf{p})$ to the model parameters.

3. Numerical Derivatives

Numerical algorithms for solving the parameter estimation problems require the evaluation of the gradient, the Jacobian- and the Hessian matrices of the objective function associated with the underlying problem and, therefore, the corresponding first- and second-order derivatives of the model solution $\mathbf{y}(t;\mathbf{p})$ with respect to the parameters $\mathbf{p}$. The methods for accurate and inexpensive computations of the first and second derivatives of $\mathbf{y}(t;\mathbf{p})$ play a crucial role in the development of accuracy and cost efficient parameter estimation problems.

A widely spread finite-difference (FD) approximation of derivatives suffers from the so-called step-size dilemma: choosing a small step size to minimize the truncation error can cause subtractive cancelation error to dominate. As a result, this approximation often does not allow one to get an accuracy which is required.

Another broadly used approach to estimate the first derivatives of $\mathbf{y}(t;\mathbf{p})$ with respect to $\mathbf{p}$ is to solve the variational equations (VEs), obtained by differentiating (2.1) with respect to $\mathbf{p}$,

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial \mathbf{y}}{\partial p_i}\Big) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\Big(\frac{\partial \mathbf{y}}{\partial p_i}\Big) + \frac{\partial \mathbf{f}}{\partial p_i}, \quad \frac{\partial \mathbf{y}}{\partial p_i}(t_0) = \mathbf{y}_{0,i}, \quad i = 1, 2, \cdots, n_p, \tag{3.1}$$

together with (2.1). To computed the second derivatives of $\mathbf{y}$, we further differentiate (3.1),

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial^2 \mathbf{y}}{\partial p_j \partial p_i}\Big) = \frac{\partial^2 \mathbf{f}}{\partial \mathbf{y}^2}\Big(\frac{\partial \mathbf{y}}{\partial p_j}\Big)\Big(\frac{\partial \mathbf{y}}{\partial p_i}\Big) + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\Big(\frac{\partial^2 \mathbf{y}}{\partial p_j \partial p_i}\Big) + \frac{\partial^2 \mathbf{f}}{\partial p_j \partial p_i}, \tag{3.2a}$$

$$\frac{\partial^2 \mathbf{y}}{\partial p_j \partial p_i}(t_0) = \mathbf{y}_{0,i,j}, \tag{3.2b}$$

$i, j = 1, \cdots, n_p$, and solve (2.1) appended by (3.1) and (3.2). We name the system (3.2) as DVEs. The main drawbacks of this approach are: (i) coding of the right hand side of the VEs and DVEs requires significant effort; (ii) the corresponding IVPs might be expensive to solve.

In recent years, two techniques for accurate approximation of derivatives gained interest in computational practice - automatic differentiation techniques (AD) [19] and the complex-step derivative approximation (CSD) [24]. Theoretically, AD gives exact results in infinite precision arithmetic. In terms of accuracy the CSD approximation to the first derivatives is competitive with AD: it is free from the step size dilemma, the truncation error can be eliminated almost completely by choosing a very small step size. However, the published results indicated that the CSD approximations to the second derivatives are sensitive to the step size, though, compared to FD, this sensitivity seems to be weaker so that cancelation error can be reduced.

To evaluate the first and second derivatives of $\mathbf{y}(\mathbf{t};\mathbf{p})$ with respect to $\mathbf{p}$, we implemented the variational equations-, automatic differentiation and complex-step derivative approximation methods. We compared these three approaches to understand their advantages and limitations. As a basic code for implementing the AD and CSD methods, we used the code `BV78` [23]. Below we describe the corresponding implementation details.

3.1. Automatic differentiation

Automatic differentiation techniques allow one to compute derivatives analytically [19]. To approximate the derivatives of $\mathbf{y}(t;\mathbf{p})$ with respect to $\mathbf{p}$ using this approach, we apply the forward mode of AD directly on the numerical integrator implemented in the code `BV78`. Namely, we differentiate (once or twice, depending on the order of the derivatives required) each line of the Runge-Kutta scheme, which involves the computed approximation to $\mathbf{y}(t;\mathbf{p})$, except the formulas determining the integration step size. The latter needs to be clarified. In case of a constant step size, the resulting numerical scheme for approximate derivatives is exactly the original Runge-Kutta scheme applied to the VEs (3.1) or to the DVEs (3.2), cf., e.g. [21]. However, the Runge-Kutta scheme in `BV78` is a variable step size algorithm. The step size is determined by the error in the computed solution and hence it depends on the model parameters. If, applying the AD formally, we differentiate the step size with respect to $\mathbf{p}$, the resulting scheme is not the Runge-Kutta scheme anymore and, hence, its stability and convergence properties will change. For this reason, we did not differentiate the step size and left the step-size selection scheme unchanged. The difference between the obtained numerical scheme and the one used by the code `BV78` to solve the VEs (and the DVEs) together with (2.1) is that in the first case the step size is adjusted to the error in the computed solution of (2.1), while in the later case the step size is adjusted to the error in the computed solution and its derivatives. The scheme obtained by AD allows one to compute accurate (up to machine precision) derivatives of the numerical solution to (2.1). These derivatives closely approximate the derivatives of the exact solution as the step size of time integration tends to zero. Hence, one can control the accuracy of the computed derivatives by assigning certain tolerances on the computed solution.

The corresponding AD-modifications of `BV78`, named by `ode78_ad1` and `ode78_ad2`, can be used to evaluate the first, respectively, the first and second derivatives of $\mathbf{y}(t;\mathbf{p})$ with respect to $\mathbf{p}$. To use `ode78_ad1` and `ode78_ad2`, the user has to code, together with the function $\mathbf{f}(\mathbf{y}(\mathbf{p}),\mathbf{p})$, the first, respectively, the first and second derivatives of $\mathbf{f}(\mathbf{y}(\mathbf{p}),\mathbf{p})$ with respect to $\mathbf{p}$, as in the case with the VEs and DVEs approaches.

3.2. Complex-step derivative approximation

If $F(x)$ is a real function of a real variable and it is also analytic, we can expand it as a Taylor series about a real point x as

$$F(x+ih) = F(x) + ihF'(x) - \frac{h^2}{2}F''(x) - i\frac{h^3}{6}F'''(x) + \cdots . \tag{3.3}$$

Hence,

$$F'(x) = \frac{\Im(F(x+ih))}{h} + \mathcal{O}(h^2), \quad F(x) = \Re(F(x+ih)) + \mathcal{O}(h^2). \tag{3.4}$$

The first equality in (3.4) is called the complex-step derivative approximation. Since this estimate for the first derivative does not involve a difference operation (on the contrary to FD), it is not subject to subtractive cancelation errors and hence a very small step h can be used. Fundamental results on the complex-step approximation to the first derivatives can be found, e.g., in [17, 22, 24] and the references therein.

The following approximation of the second derivative of an analytic function $F(\mathbf{x})$ by the CSD method has been proposed in the literature [1, 12, 20],

$$\frac{\partial^2 F(\mathbf{x})}{\partial x_j \partial x_k} \approx \frac{1}{2h_j h_k}\Im[F(\mathbf{x}+ih_j\mathbf{e}_j + h_k\mathbf{e}_k) - F(\mathbf{x}+ih_j\mathbf{e}_j - h_k\mathbf{e}_k)] =: H_{j,k}, \tag{3.5}$$

where $\mathbf{e}_j$ is the vector with jth element equal to one and other elements equal to zero, and the step sizes h_j and h_k can be either equal or different. To the best of our knowledge, no formulas avoiding subtraction for approximating second derivatives by the CSD method have been suggested in the literature. Clearly, the differencing operation in (3.5) may reduce the accuracy of this formula by several orders of magnitude.

Another drawback of approximation (3.5) is that in general, $H_{i,j} \neq H_{j,i}$, i.e., the Hessian H for $F(\mathbf{x})$ is not symmetric. In [1], it was suggested to use $(H + H')/2$. However, this approach does not necessarily guarantee a more accurate approximation of the Hessian.

An optimal choice of the step sizes h_j and h_k in (3.5) is important to get a higher accuracy. As follows from (3.5), h_j can be very small since it contributes to truncation error only, while very small h_k leads to a high subtractive cancelation error. A number of selection schemes have been proposed. In [1] it was recommended to use $h_j \ll h_k$ and the specific implementation was based on $h_j = 10^{-35}$, $h_k = \sqrt{\epsilon}$, $\epsilon = 2^{-52}$. In [12] the following values were chosen: $h_j = h_k = \sqrt{\epsilon}$. In [20] an "optimal" step size was considered to be $h_j = h_k = \epsilon^{1/5}$. Our results clearly indicate that the use of the pair (h_j, h_k) equal to $(\epsilon^{1/5}p_j,\ \epsilon^{1/5}p_k)$ is preferable if p_j is not too small, otherwise the pair $(p_j,\ \epsilon^{1/5}p_k)$ can be used. We further refer to [20] for a detailed analysis of the approximation (3.5).

Table 4.1: Quantitative dynamics of human peripheral blood T lymphocytes following stimulation with phytohemagglutinin in vitro. At various times, lymphocytes labeled with a fluorescent dye (CFSE) were characterized by flow cytometry analysis. The total numbers of dead lymphocytes, D^i, and the distribution of lymphocytes with respect to the number of divisions they have undergone, N_j^i, $j = 0, 1, \cdots, 7$, were followed from day 3 to day 7 at the indicated times t_i, $i = 0, 1, \cdots, 4$.

Time hours t_i	Total number of dead cells D^i	Numbers of cells w.r.t. the number of divisions (j) they undergone N_j^i							
		0	1	2	3	4	5	6	7
72	1.6×10^4	29358	22876	43372	39970	5208	98	14	0
96	2.4×10^4	16050	12600	22650	57025	96350	46950	2500	25
120	6.0×10^4	14476	14784	25344	58652	141460	156290	32076	440
144	1.2×10^5	13500	12150	24150	55000	137850	188950	69450	2150
168	1.3×10^5	13509	12198	21603	51927	140560	232160	96102	3420

The corresponding CSD-modifications of `BV78`, named by `ode78_csd1` and `ode78_csd2`, deal with complex functions $\mathbf{f}$ and $\mathbf{y}$ and allow one to estimate the first, respectively the first and second derivatives of $\mathbf{y}(t, \mathbf{p})$ with respect to $\mathbf{p}$. To compute the derivative $\partial\mathbf{y}(\mathbf{p}^*)/\partial p_j$, we set

$$\mathtt{p} = \mathtt{p}^*; \quad \mathtt{p(j)=p(j)} + \mathtt{eps} * \mathtt{p(j)} * \mathtt{complex(0,1)}; \tag{3.6}$$

in `ode78_csd1` and evaluate a (complex) solution `y(p)` of the given model. Then

$$\mathbf{y}(\mathbf{p}^*) \approx \Re(\mathtt{y(p)}), \quad \frac{\partial\mathbf{y}(\mathbf{p}^*)}{\partial p_j} \approx \frac{\Im(\mathtt{y(p)})}{\mathtt{eps} * \mathtt{p(j)}}. \tag{3.7}$$

In (3.6), `eps`$= 2^{-52} \approx 2.2 \times 10^{-16}$ is the floating-point relative accuracy in Matlab. To evaluate the second derivatives, we implemented formula (3.5) with (h_j, h_k) equal to $(\epsilon^{1/5}p_j,\ \epsilon^{1/5}p_k)$ if $p_j > 10^{-10}$ and $(p_j,\ \epsilon^{1/5}p_k)$ otherwise. Since the computed solution is complex, we modified the error control so that the integration step size is adjusted according to the error in the modulus of all the variables involved in its determination in the original code.

4. Results

We analyzed the computational efficiency of the derivatives approximation schemes outlined above to understand their performance in the context of parameter estimation problems in immunology. Two representative models are considered, the first represents a basic mathematical tool for the kinetic parameters analysis of the cell proliferation CFSE assay [14] and the other describes the dynamics of antiviral cytotoxic T lymphocytes (CTL) in hepatitis B infection [8]. These models were calibrated using experimental and clinical data, respectively.

Model 1. This system models the rates of change of the population sizes of live T lymphocytes having undergone j divisions (N_j) and dead but not disintegrated lymphocytes (D) [14],

$$\frac{\mathrm{d}N_0}{\mathrm{d}t}(t) = -(\alpha + \beta)N_0(t), \tag{4.1a}$$

$$\frac{\mathrm{d}N_j}{\mathrm{d}t}(t) = 2\alpha N_{j-1}(t) - (\alpha + \beta)N_j(t), \quad j = 1, \cdots, 7, \tag{4.1b}$$

$$\frac{\mathrm{d}D}{\mathrm{d}t}(t) = \sum_{j=0}^{7} \beta N_j(t) - \delta D(t), \quad t \geq t_0. \tag{4.1c}$$

Table 4.2: Clinical data on the virus-CTL dynamics in blood during the acute phase of HBV infection.

Time days t_i	Virus population density HBV-DNA (copy/ml) V_i	T lymphocytes population density (cell/ml) E_i
70	7.2×10^8	77
77	6.1×10^9	97
84	6.1×10^9	
91	1.83×10^9	
98	1.4×10^8	
105	2.5×10^5	2273
119		1059
140		584

Here, for simplicity the model assumes that the per capita proliferation and death rates of lymphocytes, α and β, do not depend on the number of divisions the lymphocytes have completed, which naturally define the generation or division-age (compartmental) structure of the cell population. The first term on the right of equations for $N_j(t)$ represents the cell birth (influx from the previous generation cell compartment due to division), while the last term represents cell loss (outflux from the current generation compartment) due to division and death. In the equation for dead cells, δ denotes the specific decay rate of dead lymphocytes due to disintegration and phagocytosis. The initial conditions $N_j(t_0)$ and $D(t_0)$ are defined by the experimental data presented in Table 4.1 at time $t_0 = 72$ hours. The objective is to estimate the model parameters $\mathbf{p} = [\alpha, \beta, \delta] \in \mathcal{P}$ so that the model solution with the estimated parameters best fits the data in Table 4.1. Here $\mathcal{P}$ is a 3-D(imensional) space of nonnegative real numbers.

Model 2. This system describes the rate of changes of the population densities of the hepatitis B virus (HBV) $V(t)$ and virus-specific cytotoxic T lymphocytes (CTLs) $E(t)$ in the acute immune response to the infection [8],

$$\frac{\mathrm{d}V}{\mathrm{d}t}(t) = \beta V(t)(1 - V(t)/K) - \gamma V(t)E(t), \tag{4.2a}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t}(t) = bV(t)E(t)/(\theta + V(t)) - \alpha E(t) + C, \quad t \geq 0, \tag{4.2b}$$

$$V(0) = 10, \quad E(0) = C/\alpha. \tag{4.2c}$$

Here the per capita parameters have the following biological meaning: β is the replication rate of viruses, γ is the rate of virus clearance due to CTLs, K is the virus carrying capacity, b is the rate of CTL stimulation, θ is the viral load saturation in CTL expansion rate, α is the CTL death rate and C is the rate of CTL influx from thymus. The clinical data on hepatitis B virus infection, presented in Table 4.2, were used to estimate the model parameters $\mathbf{p} = [\beta, \gamma, K, b, \theta, \alpha, C] \in \mathcal{P}$, where $\mathcal{P}$ is a 7-D space of real numbers such that $\beta \geq 0$, $\gamma \geq 0$, $K > 0$, $b \geq 0$, $\theta \geq 0$, $\alpha > 0$, $C \geq 0$.

To avoid dealing with very large and small numbers, we rewrite this model, using the logarithmic transformation $\tilde{V}(t) = \ln(V(t))$ and $\tilde{E}(t) = \ln(E(t))$. From now on we will use the transformed system omitting the tilde sign for simplicity,

$$\frac{\mathrm{d}V}{\mathrm{d}t}(t) = \beta(1 - \exp(V(t))/K) - \gamma \exp(E(t)), \tag{4.3a}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t}(t) = b\exp(V(t))/(\theta + \exp(V(t))) - \alpha + C/\exp(E(t)), \quad t \geq 0, \tag{4.3b}$$

$$V(0) = \ln(10), \; E(0) = \ln(C/\alpha). \tag{4.3c}$$

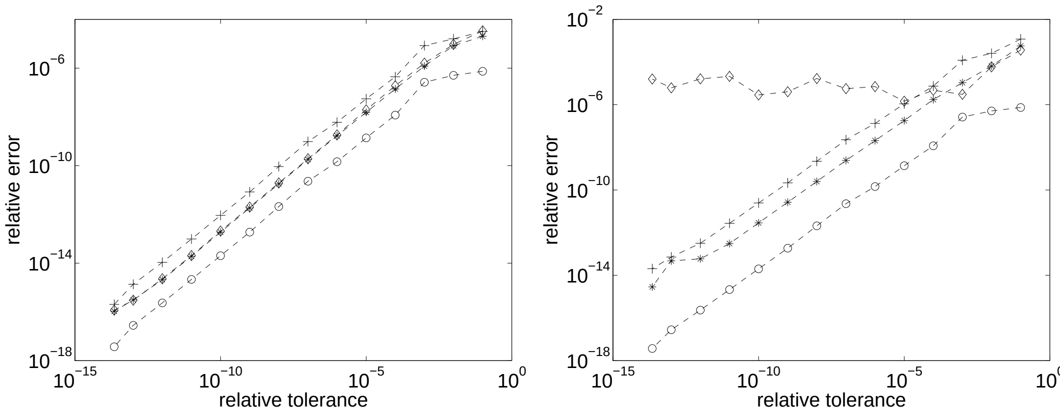

Fig. 4.1. The relative error (4.4) of the first (left) and second (right) derivatives computed using the implemented VEs, DVEs ($*$), CSD ($\diamond$) and AD ($+$) approaches versus the relative tolerance of the time integrator. The error of the computed solution ($\circ$) is shown for comparison. $\mathbf{p} = [0.3,\ 10^{-8},\ 0.2]$.

4.1. Accuracy and computational expense of numerical derivatives

Due to the availability of the closed-form solution to model (4.1), we estimate the accuracy of the computed derivatives using all the approaches described in the previous section. Next, we compare the CPU time needed to evaluate the derivatives for both illustrative models.

First derivatives. To estimate the accuracy of the numerical derivatives obtained by the VEs, the AD and CSD approaches, we computed the maximal absolute relative error at the time $t_{final} = 168$ hours,

$$E_{VE} = \max \frac{|D - D_{VEs}|}{(I + |D|)}, \quad E_{AD} = \max \frac{|D - D_{AD}|}{(I + |D|)}, \tag{4.4a}$$

$$E_{CSD} = \max \frac{|D - D_{CSD}|}{(I + |D|)}, \tag{4.4b}$$

where the elements of the matrices $D, D_{VE}, D_{AD}, D_{CSD} \in \mathbb{R}^{n \times n_p}$ are the analytical first derivatives of the model solution with respect to the model parameters, respectively, the derivatives computed by the VEs, the AD and CSD methods, $I \in \mathbb{R}^{n \times n_p}$ is a matrix of ones, and the maximum is taken with respect to all entries of the matrices.

Since the integration step size in the implemented VEs and CSD approaches is adjusted to the error in the computed solution and its first derivatives, although in different ways, we expect that $E_{VE} \approx E_{CSD}$, while the accuracy of the AD approach might be poorer (since the step size is adjusted to the error in the computed solution only). Indeed, for a number of vectors of the model parameters, we obtained that E_{VE} and E_{CSD} have the same order of magnitude, while the order of E_{AD} was either equal to the one of E_{VE} or lower by one. For example, using the relative and absolute tolerances 10^{-12}, respectively, 10^{-14} for the time integration, we obtained that $E_{VE}, E_{CSD} \in (10^{-15}, 10^{-14})$ and $E_{AD} \in (10^{-14}, 10^{-13})$. Fig. 4.1 (left) demonstrates the computed relative error of the three approaches versus the value of the relative tolerance (the absolute tolerance used is two orders less) for a particular vector of the model parameters.

Table 4.3 presents the CPU time required to evaluate derivatives $\partial \mathbf{y}(\mathbf{p})/\partial \mathbf{p}$ by the VEs, the AD and CSD methods with ($v+$) and without ($v-$) the vectorization facility. Recall that

Table 4.3: The CPU time (seconds) required to solve numerically the indicated problems. The time measurements were obtained by 1000 runs of each problem and then dividing the total time by 1000. Computations were carried out on PC Intel Core 2 CPU 2.16 GHz.

Problem	VEs-approximation of first derivatives		AD-approximation of first derivatives		CSD-approximation of first derivatives		IVP for original model	
Code used	`BV78`		`ode78_ad1`		`ode78_csd1`		`BV78`	
Vectorization	$v+$	$v-$	$v+$	$v-$	$v+$	$v-$	$v+$	$v-$
CPU time								
model (4.1)	0.018	0.081	0.021	0.049	0.024	0.088	0.0051	0.018
model (4.3)	0.024	0.051	0.026	0.032	0.066	0.21	0.0047	0.014

we evaluate $n \times n_p$ derivatives, i.e., 27 and 14 derivatives for models (4.1), respectively, (4.3). For comparison, we give the CPU time required to solve the IVP for the considered models. Although this table presents the CPU time estimates for two particular models, the results are representative enough to make some general conclusions, presented below.

- If the number of parameters n_p is small (e.g. $n_p = 3$ for model (4.1)), the three methods with the vectorization facility are equally expensive. If n_p is moderate or large (e.g. $n_p = 7$ for model (4.3)) the CSD method is several times slower. The reason is that an ODE system is simulated n_p times to compute n_p derivatives of the solution vector by the CSD method, while with the VEs and AD methods it is solved only once. However, in the implemented VEs and AD the explicit coding of the derivatives $\partial \mathbf{f}(\mathbf{p})/\partial \mathbf{p}$ by the user is needed. The trade-offs between the CSD and AD methods are discussed, e.g. in [1,17,22].
- If the vectorization facility is not used, the AD method is significantly faster for any number of parameters.
- The vectorization facility of the implemented AD method does not decrease the computational time significantly in case of nonlinear models (e.g. model (4.3)). This is due to manipulations with 3-D arrays appearing in coding the right hand side of the model and its derivatives. For convenience in programming, the input and output arguments of this function, corresponding to derivatives, are not vectors, but the Jacobian matrices. When one wishes to use the vectorization facility, these matrices have to be replaced by 3-D arrays.

Second derivatives. As with the first derivatives, we evaluated the maximal absolute values of the relative errors (4.4) using matrices $D, D_{DVE}, D_{AD}, D_{CSD} \in \mathbb{R}^{(n\times n_p)\times n_p}$ for the second derivatives of the solution to model (4.1) with respect to $\mathbf{p}$. The DVEs-approach is the most accurate one since in the AD-approximation the step size is adjusted to the error in the computed solution only and the CSD-approximation of the second derivatives suffers from cancelation error. For this reason, we compare the errors E_{AD} and E_{CSD} versus E_{DVE}. For a number of vectors of the model parameters values, we obtained that $E_{AD}/E_{DVE} \in (1, 10^2)$, while $E_{CSD}/E_{DVE} \in (1, 10^{11})$. For example, using the relative and absolute tolerances 10^{-12}, respectively, 10^{-14}, we obtained that $E_{DVE} \in (10^{-15}, 10^{-12})$, $E_{AD} \in (10^{-14}, 10^{-12})$, while $E_{CSD} \in (10^{-12}, 10^{-2})$. Fig. 4.1 (right) depicts the computed E_{DVE}, E_{CSD} and E_{AD} versus the value of the relative tolerance for a particular vector of the model parameters. The above results suggest that the implemented CSD-approach does not provide in general a reliable approximation to the second derivatives.

Table 4.4: The CPU time (seconds) required to solve the indicated problems. The time measurements were obtained by 1000 runs of each problem and then dividing the total time by 1000. Computations were carried out on PC Intel Core 2 CPU 2.16 GHz.

Problem	DVEs-approximation of second derivatives		AD-approximation of second derivatives	CSD-approximation of second derivatives	
Code used	`BV78`		`ode78_ad2`	`ode78_csd2`	
Vectorization	$v+$	$v-$	$v-$	$v+$	$v-$
CPU time					
model (4.1)	0.056	0.41	0.19	0.22	0.71
model (4.3)	0.40	0.85	0.12	1.1	3.1

The CPU time required to compute the second derivatives of $\mathbf{y}(\mathbf{p})$ by the implemented methods is presented in Table 4.4 for both illustrative models. Note that we did not implement the vectorization facility for the AD method since it is a rather tedious task in the case of second derivatives. We observe that the implemented AD is almost as fast as the CSD method with the vectorization if the number of parameters n_p is small (model (4.1)), otherwise the AD is much faster (model (4.3)). The reason is the same as in the case of the first derivatives: an ODE system is simulated $(n_p + 2n_p^2)$ times to compute $(n_p + n_p^2)$ first and second derivatives of the solution by the CSD method, while with the AD method the system is solved only once. As in the first derivatives case, the AD method is significantly faster than the CSD and DVEs methods when the vectorization facility is not used. The DVEs approach with vectorization can be cheaper or more expensive than the AD depending on the complexity of the approximated derivatives.

4.2. Constrained optimization

In this section we illustrate the performance of the code `GN_NLS` for both mathematical models and compare it with the code `lsqnonlin` from the Matlab Optimization Toolbox. The latter code solves (2.2) by the trust-region-reflective algorithm. Our experience suggests that the efficiency of `GN_NLS` depends on the problem under study, for some problems the code `lsqnonlin` gives better results, for others `GN_NLS` is preferable. We note that the constraints $p_j \geq 0$ should be replaced by the constraints $p_j \geq \tau$ with $0 < \tau \ll 1$ to avoid the zero divisor in (3.7) if the CSD method is used to compute derivatives. We set $\tau = 10^{-15}$ in the examples below, i.e., the constraints used for the minimization problem (2.2) are $\mathbf{c} = \mathbf{p} - 10^{-15} \geq 0$.

Model (4.1) *Case 1.* We scale the model solution and the data (Table 4.1) by the factor 10^{-5} to avoid dealing with very large numbers. Starting with the initial guess

$$\alpha = 0.1, \quad \beta = 0.1, \quad \delta = 0.1,$$

we obtain the following maximum likelihood estimates

$$\begin{aligned} &\alpha^* \approx 2.13 \times 10^{-2}, \quad \beta^* \approx 3.35 \times 10^{-3}, \quad \delta^* \approx 10^{-15}, \quad \Phi(\mathbf{p}^*) \approx 6.15, \\ &\nabla\Phi(\mathbf{p}^*) \approx [8 \times 10^{-4}, \ -4 \times 10^{-2}, \ 30]^T. \end{aligned} \tag{4.5}$$

We observe that the best-fit value δ^* is close to the lower bound. A relatively large value of $\partial\Phi(\mathbf{p}^*)/\partial\delta$ indicates that the zero value of this derivative is attained for $\delta < 0$.

The CPU time required to solve the minimization problem for the above model (as well as for model (4.3)) is given in Table 4.5. The table values indicate that the computational

Table 4.5: The CPU time (seconds) required to solve the indicated minimization problems by the code `GN_NLS` with the approximation of the first derivatives of the objective by the indicated methods.

Method	Model (4.1)				Model (4.3)			
	Case 1		Case 2		Case 1		Case 2	
	$v+$	$v-$	$v+$	$v-$	$v+$	$v-$	$v+$	$v-$
VEs	0.72	3.4	2.3	11	0.61	1.2	2.5	4.8
AD	0.94	3.1	3.3	7.6	0.66	0.82	2.8	3.4
CSD	1.0	3.7	3.4	13	1.8	5.2	7.3	21

cost of solving the minimization problem depends on the method used to approximate the first derivatives of the objective function with respect to the model parameters. The `GN_NLS` with the VEs-approximation and the vectorization is the cheapest one, but it requires maximum user effort - to code the derivatives $\partial\mathbf{f}(\mathbf{y}(\mathbf{p}),\mathbf{p})/\partial\mathbf{p}$ in a matrix form. If we avoid the use of the vectorization (and hence reduce the user effort), then the `GN_NLS` with the AD-approximation is preferable. The use of the CSD-approximation leads to the most expensive computations, however its implementation requires much less effort from the user. For the code `lsqnonlin` with the user provided Jacobian of residuals (approximated by the three methods used) a similar CPU time is required to find the best-fit values (4.5).

Case 2. If the following parameter values are used for the starting point

$$\alpha = 0.3, \quad \beta = 0.4, \quad \delta = 0.3,$$

then again the values corresponding to the minimum point (4.5) are obtained. On the contrary, the code `lsqnonlin` turned out to be inefficient so that the solution was not found in 15 min. Note that the starting point in the above case is located rather far from the minimum point.

Case 3. If the starting point for the minimization procedure is

$$\alpha = 0.1, \quad \beta = 0.3, \quad \delta = 0.1,$$

then the code `GN_NLS` gets stack as soon as the model parameters at the computed minimum (and the objective function) reach the values $[6.06 \times 10^{-3},\ 3.88,\ 6.06 \times 10^{-3}]$ (and 23.3). The underlying reason is that the time integration of the model with these parameter values becomes too expensive (the step size is less than 10^{-9}) due to the stiffness of the ODE system. In such cases the code should be interrupted by the user and started with another initial guess. The code `lsqnonlin` found the the best-fit solution (4.5) in a few seconds.

Model (4.3) *Case 1.* If the starting point is located not far from a minimum of $\Phi(\mathbf{p})$,

$$\beta = 0.2, \quad \gamma = 6 \times 10^{-4}, \quad K = 10^{10}, \quad b = 0.1, \quad \theta = 10^{7}, \quad \alpha = 0.03, \quad C = 0.2,$$

then the following best-fit parameters are computed by the code `GN_NLS`

$$\begin{aligned} &\beta^* \approx 0.273, \quad \gamma^* \approx 6.18 \times 10^{-4}, \quad K^* \approx 1.48 \times 10^{10}, \quad b^* \approx 0.151, \\ &\theta^* \approx 1.24 \times 10^{7}, \quad \alpha^* \approx 3.49 \times 10^{-2}, \quad C^* \approx 0.228, \quad \Phi(\mathbf{p}^*) \approx 0.790. \end{aligned} \tag{4.6}$$

The code `lsqnonlin` computes another solution to the minimization problem:

$$\begin{aligned} &\beta^* \approx 0.272, \quad \gamma^* \approx 5.64 \times 10^{-4}, \quad K^* \approx 1.00 \times 10^{10}, \quad b^* \approx 0.160, \\ &\theta^* \approx 1.00 \times 10^{7}, \quad \alpha^* \approx 3.88 \times 10^{-2}, \quad C^* \approx 0.188, \quad \Phi(\mathbf{p}^*) \approx 0.871. \end{aligned}$$

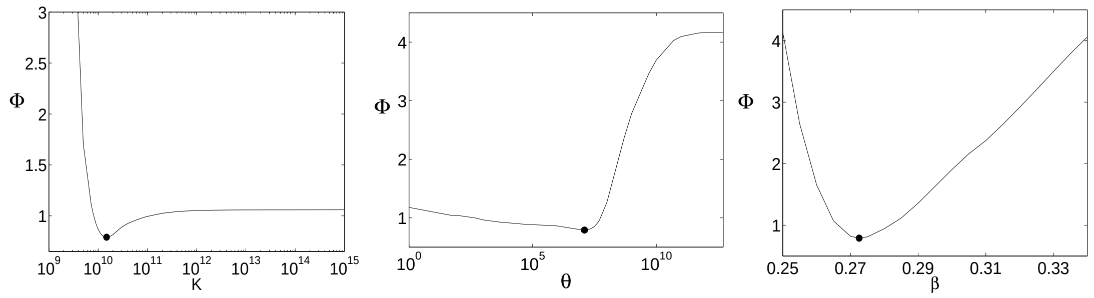

Fig. 4.2. The dependence of the minimized function $\Phi(\tilde{\mathbf{p}})$ on the estimated parameters K (left), θ (middle) and β (right). Here $\tilde{\mathbf{p}} = \arg\min \Phi(\tilde{\mathbf{p}}) \in \mathbb{R}^{n_p - 1}$ for fixed K, respectively, θ or β. The bold circles indicate the location of the computed minimum.

The objective function at this minimum has a larger value than the one obtained by GN_NLS.

Case 2. If we start rather far from the minimum (4.6) using the following parameter values:

$$\beta = 0.3, \quad \gamma = 0.01, \quad K = 10^5, \quad b = 0.2, \quad \theta = 10^4, \quad \alpha = 0.01, \quad C = 0.1,$$

then the code GN_NLS gives a few warnings in the beginning of the minimization procedure,

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.772813e-017.
> ... In quadprog at 271
```

Nevertheless, the optimal solution (4.6) is computed. On the contrary, the code `lsqnonlin` failed to generate results in 15 min. The above warning is due to a badly scaled or nearly singular matrix which is involved in the QP algorithm (`quadprog` code). This matrix depends on the Jacobian of the residuals via the matrix $B = J^T J$ in (2.6). The elements of B indicate sensitivity of $\Phi(\mathbf{p})$ to the parameters. The reason for the difficulties in computing the minima of the objective function for model (4.3) is illustrated in Fig. 4.2. Here $\Phi(\tilde{\mathbf{p}})$ with $\tilde{\mathbf{p}} = \arg\min \Phi(\tilde{\mathbf{p}}) \in \mathbb{R}^{n_p-1}$ for fixed K, respectively, θ or β is shown. As we see, $\Phi(\mathbf{p})$ is parabolic only in a very small neighborhood of the computed minimum and the sensitivity of $\Phi(\mathbf{p})$ to the variation in the parameters K and θ is rather low within broad ranges of these parameters.

4.3. Computing confidence intervals for the estimated parameters

Model (4.1) Using the variance-covariance method with the second derivatives evaluated by either the AD or the CSD method, we obtained the following estimates to the 95% confidence intervals (similar up to 3 significant digits) for the best-fit model parameters

$$\mathrm{CI}_\alpha = [1.59,\ 2.66] \times 10^{-2}, \quad \mathrm{CI}_\beta = [0,\ 8.49 \times 10^{-3}], \quad \mathrm{CI}_\delta = [0,\ 3.58 \times 10^{-2}]. \tag{4.7}$$

The profile-likelihood-based method gives

$$\mathrm{CI}_\alpha = [1.81,\ 2.49] \times 10^{-2}, \quad \mathrm{CI}_\beta = [1.38,\ 6.55] \times 10^{-3}, \quad \mathrm{CI}_\delta = [0,\ 1.87 \times 10^{-2}]. \tag{4.8}$$

We observe that the ranges for CIs (4.8) are more narrow than the ones computed by the first approach. Contrary to the variance-covariance method, the profile-likelihood-based method computes nonsymmetric approximation to the CIs with respect to the best-fit parameter values.

Model (4.3) Computing the estimates to the 95% confidence intervals for the best-fit model parameters by the variance-covariance method with the Hessian matrix of the objective $\Phi(\mathbf{p}^*)$ evaluated by the AD method, we get the following warning concerning the Hessian

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 6.338546e-030.
```

In this example the objective function is not a parabolic surface in the vicinity of the best-fit shown in Fig. 4.2. Hence, the variance-covariance approximation of the CIs,

$$\begin{aligned} &\mathrm{CI}_\beta = [2.53,\ 2.92]\times 10^{-1}, \quad \mathrm{CI}_\gamma = [0,\ 1.42\times 10^{-3}], \quad \mathrm{CI}_K = [0,\ 4.51\times 10^{10}],\\ &\mathrm{CI}_b = [4.26\times 10^{-2},\ 2.60\times 10^{-1}], \quad \mathrm{CI}_\theta = [0,\ 8.41\times 10^{7}], \quad \mathrm{CI}_\alpha = [0,\ 8.28\times 10^{-2}],\\ &\mathrm{CI}_C = [0,\ 7.23\times 10^{-1}], \end{aligned}$$

can be subject to a large bias. If we estimate the CIs using the CSD method to evaluate the Hessian of the objective function, then we get additional warning that the Hessian is not a positive-definite matrix since two of its eigenvalues are negative $(-6\times 10^{-12}, -2\times 10^{-13})$. In fact, two eigenvalues of the Hessian are close to zero $(10^{-13},\ 6\times 10^{-16})$ and therefore their computed approximate values can be negative. The computed ranges to the CIs with the CSD method are similar to the ones presented above up to 3 significant digits.

Using the profile-likelihood-based method, we computed the following estimates to the CIs

$$\begin{aligned} &\mathrm{CI}_\beta = [2.64,\ 2.85]\times 10^{-1}, \quad \mathrm{CI}_\gamma = [0.347,\ 1.32]\times 10^{-3}, \quad \mathrm{CI}_K = [7.15\times 10^{9},\ \infty],\\ &\mathrm{CI}_b = [1.01,\ 2.02]\times 10^{-1}, \quad \mathrm{CI}_\theta = [4.75,\ 7.55\times 10^{7}], \quad \mathrm{CI}_\alpha = [1.25,\ 5.73]\times 10^{-2}, \\ &\mathrm{CI}_C = [0.0382,\ 5.37]\times 10^{-1}. \end{aligned} \tag{4.9}$$

Interestingly, all the CIs, except for CI_K, computed by the profile-likelihood-based method are more narrow than the CIs computed by the variance-covariance method.

4.4. Sensitivity of the parameter estimates to the observation data

To evaluate the sensitivity of the estimated parameters to the data, the following relation between the data variations $\Delta_d \in \mathbb{R}^{n_d}$ and the corresponding parameter deviations $\Delta_p \in \mathbb{R}^{n_p}$ can be used,

$$S\Delta_p = \Delta_d, \quad \text{where} \quad S_{i,j}(\mathbf{p}^*) = \frac{\partial Y_i(\mathbf{p}^*)}{\partial p_j}, \quad S(\mathbf{p}^*) \in \mathbb{R}^{n_d\times n_p},$$

is the sensitivity coefficient matrix [7] and Y_i is the i-th element of the vector $\mathbf{Y}\in\mathbb{R}^{n_d}$,

$$\mathbf{Y} := [y_1(t_{1,1}), y_1(t_{1,2}), \cdots, y_1(t_{1,k_1}), y_2(t_{2,1}), \cdots, y_2(t_{2,k_2}), \cdots, y_n(t_{n,k_n})],$$

which consists of the elements of the model solution $\mathbf{y}(\mathbf{p}^*)$ computed at the time points of the corresponding data measurements. Here we assume that the data corresponding to different elements of $\mathbf{y}$ can be measured at different times and that $\sum_{i=1}^{n} k_i = n_d$. Hence, each row of S corresponds to n_p derivatives of a certain component of the vector of the model solution computed at a certain sampling time.

The spectral condition number

$$\tau := \max_{i,j=1,\dots,n_p} \frac{\sigma_i}{\sigma_j}, \tag{4.10}$$

with $\{\sigma_i\}_{i=1}^{n_p}$ standing for the square roots of the eigenvalues of the matrix $\mathbf{S}^T\mathbf{S}$ indicates the influence of measurement errors on the identified parameters: the parameter estimation procedure via minimization of the objective function $\Phi(\mathbf{p})$ is more stable for smaller values of τ. The values of τ computed for models (4.1) and (4.3) are $\tau \approx 350$ and $\tau \approx 10^{21}$, respectively. One can conclude that small variations in the data will not cause large changes in the identified parameters of model (4.1) and, therefore, the corresponding minimization problem is well-posed. On the contrary, the minimization problem for model (4.3) is unstable and hence ill-posed.

4.5. Fisher Information Matrix

It often happens in parameter estimation problems in immunology that some of the parameters cannot be identified robustly using the existing experimental data set(s). One of the methods to evaluate the information content of the data and the number of estimable parameters is to compute the eigenvalues of the Fisher Information Matrix (FIM) [13]

$$F(\mathbf{p}) = S(\mathbf{p})^T \Sigma S(\mathbf{p}) \in \mathbb{R}^{n_p \times n_p}. \tag{4.11}$$

Here $S \in \mathbb{R}^{n_d \times n_p}$ is the sensitivity coefficient matrix and $\Sigma \in \mathbb{R}^{n_d \times n_d}$ is the variance-covariance matrix of the data set used for parameter estimation. Since we deal with only one set of the experimental data for each illustrative model and assumed that the variance of the observation errors (σ^2) is the same for all the state variables and observation times, see section 2.1, the matrix Σ is the diagonal matrix with all diagonal elements equal to ${\sigma^*}^2 = \Phi(\mathbf{p}^*)/n_d$.

The number of large eigenvalues of F corresponds to the number of reliably estimable parameters, see, e.g., [26]. For model (4.1) the eigenvalues of $F(\mathbf{p}^*)$ range in between $10^2 - 10^4$, while for model (4.3) 5 eigenvalues are of the order $10^0 - 10^7$ and 2 eigenvalues are of the order 10^{-16}, 10^{-13}. The latter implies that two parameters of this model cannot be identified by the data set available, i.e. the model structure is not consistent with the information content of the data. In this case the model structure needs to be refined using information-theoretic criteria as presented in the next section.

4.6. Model refinement

The comprehensive analysis of the best-fit parameters presented above allows us to propose an iterative model refinement procedure. For example, the estimated 95% CI$_K$ and Fig. 4.2 (left) indicate that the objective function Φ for model (4.3) practically does not depend on K for $K > 10^{11}$. This suggests to investigate whether the term $-\beta \exp(V(t))/K$ in the first model equation is significant for the given data set. To this end, we set $K = \infty$, so that the first equation in (4.3) becomes

$$\frac{\mathrm{d}V}{\mathrm{d}t}(t) = \beta - \gamma \exp(E(t)). \tag{4.12}$$

Estimating the remaining 6 parameters of the simplified model, we found that, although the value of the objective at the computed minimum is 34% larger ($\Phi(\mathbf{p}^*) \approx 1.06$), the values of the best-fit parameters,

$$\beta^* \approx 0.279, \quad \gamma^* \approx 1.08 \times 10^{-3}, \quad b^* \approx 0.115, \quad \theta^* \approx 4.54 \times 10^4, \quad \alpha^* \approx 0.0310, \quad C^* \approx 0.0695,$$

are located inside the 95% CIs for the parameters of the original model computed by the profile-likelihood-based method (4.9).

To decide which model is preferable, one can apply the information-theoretic criteria for model assessment and ranking. We evaluate the Akaike indicator, characterizing the information complexity of the models and measuring the information loss for a particular model [11]. The Akaike criterion is based upon the Kullback-Leibler notion of the directed distance between the given model and an ‘ideal model’ of the data. It makes use of the maximum likelihood estimation to evaluate the information loss associated with the specific model for the given data set. Because our interest is in the relative size of the indicators, we considered the revised Akaike indicators obtained by discarding extraneous terms (see [5] for technical details),

$$\mu = n_d \ln(\Phi(\mathbf{p}^*)) + 2(n_p + 1) + \frac{2(n_p+1)(n_p+2)}{n_d - n_p - 2}. \tag{4.13}$$

The model with a larger value of μ is less consistent with the unknown true model of the data than the model characterized by a smaller μ. The value of Akaike index equals approximately 85 and 52 for the original and simplified models, respectively. Therefore, the simplified model is closer to the true model of the data as indicated by the smaller value of the information loss.

The variance-covariance method applied to the simplified model to compute the 95% CIs still leads to the warning of the computational procedure as in the case of the original model. However, the reciprocal condition number estimate of the computed Hessian has a larger value, rcond(H) $\approx 10^{-20}$. The computed, by the profile-likelihood-based method, estimates to the 95% CIs for the best-fit parameters are

$$\begin{array}{lll} \mathrm{CI}_\beta = [2.68,\ 2.92] \times 10^{-1}, & \mathrm{CI}_\gamma = [0.564,\ 1.65] \times 10^{-3}, & \mathrm{CI}_b = [0.836,\ 1.53] \times 10^{-1}, \\ \mathrm{CI}_\theta = [0.001,\ 7.67 \times 10^7], & \mathrm{CI}_\alpha = [0.40,\ 7.36] \times 10^{-2}, & \mathrm{CI}_C = [0.0336,\ 6.80] \times 10^{-1}. \end{array} \tag{4.14}$$

We observe that the value of the estimated parameter θ is still rather uncertain. The spectral condition number for the simplified model is $\tau \approx 10^{19}$ and there exists one very small (10^{-14}) eigenvalue of the FIM, both indicating that the parameter estimation problem is still ill-posed.

If we fix θ at the value of the low or upper boundary of CI_θ in (4.14), say $\theta = 0.001$, and estimate the remaining 5 parameters, we obtain

$$\beta^* \approx 0.281, \quad \gamma^* \approx 1.31 \times 10^{-3}, \quad b^* \approx 0.143, \quad \alpha^* \approx 6.16 \times 10^{-2}, \quad C^* \approx 8.25 \times 10^{-3},$$

and $\Phi(\mathbf{p}^*) \approx 1.51$. We observe that all the estimated values are inside the CIs (4.14) and, except the parameter α, belong to the CIs (4.9). The estimate for α is located slightly outside of the CI_b in (4.9). The value of the Akaike index is $\mu \approx 38$. Hence, the model with the fixed $\theta = 0.001$ is closer to the true model of the data. This conclusion holds true for any value of θ from the interval CI_θ in (4.14) since the value of the objective function is maximal at the ends of this interval.

The variance-covariance method applied to the model with $\theta = 0.001$ to compute the 95% CIs suggests that

$$\begin{array}{lll} \mathrm{CI}_\beta = [2.58,\ 3.04] \times 10^{-1}, & \mathrm{CI}_\gamma = [0.412,\ 2.21] \times 10^{-3}, & \mathrm{CI}_b = [0.777,\ 2.08] \times 10^{-1}, \\ \mathrm{CI}_\alpha = [0.0479,\ 1.18] \times 10^{-1}, & \mathrm{CI}_C = [0,\ 2.37] \times 10^{-2}. & \end{array}$$

Note that the Hessian of the objective is not singular in the case of 5 estimated parameters and hence the results of the variance-covariance method can be considered as accurate. The spectral condition number for the simplified model with $\theta = 0.001$ is $\tau \approx 10^6$ and the eigenvalues of the FIM range in between $10^1 - 10^7$.

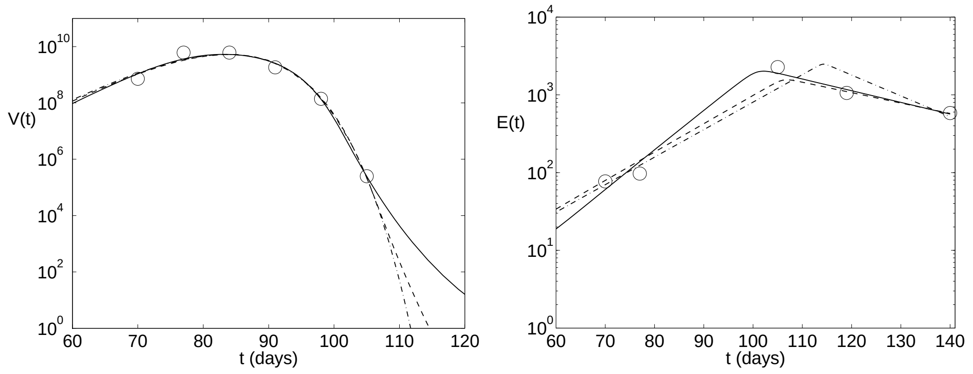

Fig. 4.3. Comparison of the clinical data (circles) and the solutions of the three considered variants of model (4.3) corresponding to the best-fit values of their parameters. Solid, dashed and dot-dashed curves correspond, respectively, to the solution of the original model (4.3), the simplified model and the simplified model with $\theta = 0.001$.

Overall, one conclude that the considered data set does not allow one to estimate reliably the parameters K and θ of model (4.3). If we set $K = \infty$ and take θ from CI_θ in (4.14), then the minimization problem becomes well-posed and the remaining model parameters can be robustly identified. Fig. 4.3 shows the consistency of the data set and the best-fit solution of the three considered variants of model (4.3). Although the best-fit solution of the reduced version of the model does not fit the data as good as the original model, the fitting quality seems still to be satisfactory.

5. Conclusions

We developed a computational methodology for analysis of the best-fit parameter estimates and the information-theoretic assessment of the mathematical models formulated with ODEs. The core element of the methodology is a robust evaluation of the first and second derivatives of the model solution with respect to the model parameter values. The critical issue of the reliable estimation of the derivatives was addressed in the context of inverse problems arising in mathematical immunology. To evaluate the first and second derivatives of the ODE solution with respect to parameters, we implemented the variational equations-, automatic differentiation and complex-step derivative approximation methods. A comprehensive analysis of these approaches to the derivative approximations is presented to understand their advantages and limitations.

We showed that depending on the specific aspects of the model (e.g. the functional forms used to parameterize the reaction terms in the equations, the number of state variables, number of parameters, completeness of the data set) the researcher in mathematical immunology has to be equipped with a broad range of tools to select the most efficient one for a given inverse modeling problem. A robust and reliable parameter estimation is a critical issue for modeling in immunology. Indeed, it has been proposed that small variations in multiple parameters account for wide variations in HIV infection outcome [18]. Understanding the feedback regulation of immune responses and relating multiple sets of experimental data, using nested models [9], requires the availability of the computational technologies to perform a comprehensive model assessment including parameter estimation, uncertainty analysis, model ranking. In this study we described in detail a thorough application of the modeling and analysis tools to representative inverse problems of the dynamics of T cell proliferation and antiviral immune response in HBV infection. Further research is urgently needed to address similar computational problems for distributed parameter systems in immunology, including spatio-temporal ones - an area of challenge and complexity.

Acknowledgments. This work was supported by the Russian Academy of Sciences Program "Basic Research for Medicine" and the Russian Foundation for Basic Research (Grant 11-01-00117-a).

References

[1] A. Abokhodair, Numerical tools for geoscience computations: semiautomatic differentiation - SD, *Comput. Geosci.*, **11** (2007), 283-296.

[2] A List of Selected AD Tools, Available from `http://www.autodiff.org/?module=Tools`.

[3] S.M. Andrew, C.T.H. Baker and G.A. Bocharov, Rival approaches to mathematical modelling in immunology, *J. Comput. Appl. Math.*, **205** (2007), 669-686.

[4] C.T.H. Baker, G. Bocharov, J. Ford, P. Lumb, S. Norton, C. Paul, T. Junt and B.L.P. Krebs, Computational approaches to parameter estimation and model selection in immunology, *J. Comput. Appl. Math.*, **184** (2005), 50-76.

[5] C.T.H. Baker, G. Bocharov, C. Paul and F. Rihan, Computational modelling with functional differential equations: identification, selection and sensitivity, *Appl. Numer. Math.*, **53** (2005), 107-129.

[6] Y. Bard, Nonlinear Parameter Estimation, New-York, 1974.

[7] S. Bitterlich and P. Knabner, An efficient method for solving an inverse problem for the Richards equation, *J. Comput. Appl. Math.*, **147** (2002), 153-173.

[8] G. Bocharov, B. Ludewig, A. Bertoletti, P. Klenerman, T. Junt, P. Krebs, T. Luzyanina, C. Fraser, and R.M. Anderson, Underwhelming the immune response: effect of slow virus growth rates on CD8+ T lymphocyte responses, *J. Virol.*, **78** (2004), 2247-2254.

[9] G. Bocharov, J. Quiel, T. Luzyanina, H. Alon, E. Chiglintsev, V. Chereshnev, M. Meier-Schellersheim, W. Paul and Z. Grossman, Feedback regulation of proliferation versus differentiation explains the dependence of antigen-stimulated CD4 T-cell expansion on precursor number, *Proc. Natl. Acad. Sci. U.S.A.*, **108**:8 (2011), 3318-3323.

[10] G. Bocharov, R. Zust, L. Cervantes-Barragan, T. Luzyanina, E. Chiglintcev, V.A. Chereshnev, V. Thiel and B. Ludewig, A systems immunology approach to plasmacytoid dendritic cell function in cytopathic virus infections, *PLoS Pathog.*, **6**:7 (2010):e1001017.

[11] K.P. Burnham and D.R. Anderson, Model Selection and Multimodel Inference - a Practical Information-Theoretic Approach, Springer, 2002.

[12] Y. Cao, Complex step Hessian, The Matlab code HESSIANCSD, available from `http://www.mathworks.com/matlabcentral/fileexchange/18177-complex-step-hessian`.

[13] N. Gershenfeld, The Nature of Mathematical Modelling, Cambridge University Press, 1999.

[14] T. Luzyanina, S. Mrusek, J.T. Edwards, D. Roose, S. Ehl and G. Bocharov, Computational analysis of CFSE proliferation assay, *J. Math. Biol.*, **54** (2007), 57-89.

[15] T. Luzyanina, D. Roose and G. Bocharov, Distributed parameter identification for a label-structured cell population dynamics model using CFSE histogram time series data, *J. Math. Biol.*, **59** (2009), 581-603.

[16] T. Luzyanina, D. Roose, T. Schenkel, M. Sester, S. Ehl, A. Meyerhans and G. Bocharov, Numerical modelling of label-structured cell population growth using CFSE distribution data, *Theor. Biol. and Math. Model.*, **4** (2007), 26-39.

[17] J. Martins, P. Sturdza and J. Alonso, The complex-step derivative approximation, *ACM T. Math. Software.*, **29** (2003), 245-262.

[18] V. Muller, A.F. Maree and R.J. De Boer, Small variations in multiple parameters account for wide variations in HIV-1 set-points: a novel modelling approach, *Proc. Biol. Sci.*, **268** (2001), 235-242.

[19] J. Nocedal and S. Wright, Numerical Optimization, Springer, 2006.

[20] M. Ridout, Statistical applications of the complex-step method of numerical differentiation, *The Am. Stat.*, **63** (2007), 66-74.

[21] A. Sandu, Sensitivity Analysis of ODE via Automatic Differentiation, *MSc. Thesis*, Graduate College of the University of Iowa, 1997.

[22] L. Shampine, Accurate numerical derivatives in Matlab, *ACM T. Math. Software.*, **33** (2007), 26-42.

[23] L. Shampine, Vectorized solution of ODEs in Matlab, *Scalable Comput.: Pract. and Exper.*, **10** (2010), 337-345.

[24] W. Squire and G. Trapp, Using complex variables to estimate derivatives of real functions, *SIAM Rev.*, **40** (1998), 110-112.

[25] D. Venzon and S. Moolgavkar, A method for computing profile-likelihood-based confidence intervals, *Appl. Stat.*, **37** (1988), 87-94.

[26] K.Z. Yao, B.M. Shaw, B. Kou, K.B. McAuley and D.W. Bacon, Modeling ethylene/butene copolymerization with multi-site catalysis: Parameter estimability and experimental design, *Polym. React. Eng.*, **11** (2003), 563-588.