Fitting ordinary differential equations to short time course data

Daniel Brewer, Martino Barenco, Robin Callard, Michael Hubank and Jaroslav Stark

*Phil. Trans. R. Soc. A* 2008 **366**, 519-544
doi: 10.1098/rsta.2007.2108

**References**

This article cites 31 articles, 2 of which can be accessed free
http://rsta.royalsocietypublishing.org/content/366/1865/519.full.html#ref-list-1

Article cited in:
http://rsta.royalsocietypublishing.org/content/366/1865/519.full.html#related-urls

**Email alerting service**

Receive free email alerts when new articles cite this article - sign up in the box at the top right-hand corner of the article or click [here](http://rsta.royalsocietypublishing.org/subscriptions).
Fitting ordinary differential equations to short
time course data

BY DANIEL BREWER¹,², MARTINO BARENCO¹,², ROBIN CALLARD¹,²,
MICHAEL HUBANK¹,² AND JAROSLAV STARK³,⁴,*

¹Institute of Child Health, University College London, 30 Guilford Street,
London WC1N 1EH, UK
²CoMPLEX, University College London, 4 Stephenson Way,
London NW1 2HE, UK
³Department of Mathematics, and ⁴Centre for Integrative Systems Biology at
Imperial College, Imperial College London, London SW7 2AZ, UK

Ordinary differential equations (ODEs) are widely used to model many systems in physics,
chemistry, engineering and biology. Often one wants to compare such equations with
observed time course data, and use this to estimate parameters. Surprisingly, practical
algorithms for doing this are relatively poorly developed, particularly in comparison with the
sophistication of numerical methods for solving both initial and boundary value problems for
differential equations, and for locating and analysing bifurcations. A lack of good numerical
fitting methods is particularly problematic in the context of systems biology where only a
handful of time points may be available. In this paper, we present a survey of existing
algorithms and describe the main approaches. We also introduce and evaluate a new efficient
technique for estimating ODEs linear in parameters particularly suited to situations where
noise levels are high and the number of data points is low. It employs a spline-based
collocation scheme and alternates linear least squares minimization steps with repeated
estimates of the noise-free values of the variables. This is reminiscent of expectation–
maximization methods widely used for problems with nuisance parameters or missing data.

Keywords: parameter estimation; ordinary differential equation; time series; splines;
collocation; systems biology

1. Introduction

Ordinary differential equations (ODEs) are one of the most popular frameworks for
describing the temporal evolution of a wide variety of systems in physics, chemistry,
engineering and biology (e.g. Gershenfeld 1999). Such models take the form

\[ \frac{dx}{dt} = f(x, t, \alpha), \]  

where \( x \) is a vector of variables evolving with time; \( f \) is a vector field; and \( \alpha \) denotes an
(optional) set of parameters. Once an ODE model has been built, a vast array of

* Author and address for correspondence: Department of Mathematics, Imperial College London,
London SW7 2AZ, UK (j.stark@imperial.ac.uk).

One contribution of 14 to a Theme Issue ‘Experimental chaos II’.
powerful analytical and numerical methods exists for exploring its properties. Any such model will usually involve one or more parameters, \( \alpha \). In some cases, particularly in physics where models are based on well-understood physical mechanisms, such parameters may be derived from first principles, or measured directly. Increasingly, however, ODEs are being applied in disciplines such as cell and molecular biology where many parameters cannot be determined by either of these approaches.

In such situations, one can attempt to estimate the unknown parameters from experimentally measured data. In most cases, these data consist of time series, or time courses, of repeated measurements of one or more experimental variables. In cell and molecular biology, these might, for instance, be mRNA or protein concentrations measured at several time points throughout an experiment. One can fit such data by systematically varying the parameters to determine a set of parameters which minimize the difference between a solution of the differential equation and the data (e.g. Gershenfeld 1999). One can also apply the same methodology to build a ‘black-box’ model by starting with a general set of basis functions (such as polynomials or radial basis functions) and then estimating their coefficients by minimizing the discrepancy between model and data. In principle, such an approach can be used to deduce the network of interactions between the variables (Stark et al. 2003a,b) which in turn can lead to more refined mechanistic models.

A variety of approaches exists to fit data to ODE models. Unfortunately, many of these are poorly documented in the literature, and may only be described in the context of specific applications in specialist publications. We therefore present an overview of the key concepts below. We show that such methods can be classified by whether the ODE is solved using a conventional iterative numerical integrator such as fourth-order Runge–Kutta, or whether a global solution is approximated using splines or related methods. This distinction is similar to that between shooting and collocation methods for boundary value problems (Golub & Ortega 1992). Although global methods are now generally preferred for solving boundary value problems, they are poorly developed in the context of fitting ODEs to data. In such cases, shooting is generally more popular and well known.

Most experiments in cell and molecular biology produce very short time courses, often with very noisy measurements. Shooting-type methods tend to perform particularly poorly in such situations; we show an example of this below. As ODEs become more widely applied in this area, particularly with the current rapid growth in systems biology, there is therefore an urgent need to develop alternative methods more suited to this type of data. The use of collocation-type algorithms appears to be particularly attractive. We explain the main ideas behind this approach below, describe existing algorithms and present a new two-stage algorithm for ODEs that are linear in parameters. To illustrate these ideas, we use a model of the p53 tumour-suppressor network, which is described briefly in appendix A.

2. Estimating parameters in ODEs

(a) General principles of model fitting

Any method for estimating model parameters from data requires two main ingredients. We need to construct an error function \( E_D(\alpha) \) that quantifies the difference between a model with parameters \( \alpha \) and the data, and we need an
optimization method that finds the value of $\alpha$ that minimizes $E_D(\alpha)$. Except for error functions that have particular features (such as those occurring in linear least squares problems), the minimization stage requires an iterative approach. Typically, the minimization method can be chosen independently of the construction of the error function. In some cases, however, an integrated approach can have advantages. This can occur, for instance, if it is sufficient to compute only a rough estimate of $E_D(\alpha)$ at each iteration of the minimization scheme, rather than calculating $E_D(\alpha)$ exactly.

A wide variety of standard minimization algorithms exist (e.g. Gershenfeld 1999). Where $E_D(\alpha)$ has no, or only a few, local minima apart from the global minimum, then methods that iteratively step downhill, such as the Nelder–Mead simplex method (Nelder & Mead 1965 or see Press et al. 2002) or the Levenberg–Marquardt method (Marquardt 1963 or see Press et al. 2002) work well. If, on the other hand, $E_D(\alpha)$ has a more complex landscape, stochastic search algorithms such as simulated annealing (Gershenfeld 1999 or Press et al. 2002), Markov Chain Monte Carlo (e.g. Gilks et al. 1996) or genetic algorithms (e.g. Gershenfeld 1999) are often necessary. These are usually computationally very demanding.

Attempting to estimate the parameters $\alpha$ of the ODE in equation (1.1) from observed data presents additional difficulties. These are mainly centred on the construction and efficient computation of a suitable error function. This is because we cannot directly determine how well a given set of data points

$$D = \{ \hat{x}(t_i) : i = 1, \ldots, n \},$$

fits the ODE in equation (1.1). We shall describe two common strategies for the construction and minimization of appropriate error functions.

(b) Solution-based approaches

By far the better known, and arguably statistically more valid, approach is to solve equation (1.1) numerically to obtain an approximate solution $u(t)$ such that

$$\frac{du}{dt} \approx f(u, t, \alpha).$$

(2.1)

Since this is meant to provide a model for the data, the points $\hat{x}(t_i)$ should be close to the values $u(t_i)$. It is thus natural to base the error function $E_D(\alpha)$ on the difference between $u(t_i)$ and $\hat{x}(t_i)$. The most common choice is to use the least squares error

$$E_D(\alpha) = \sum_{i=1}^{n} ||u(t_i) - \hat{x}(t_i)||^2,$$

(2.2)

possibly weighted by the reciprocal of the noise level at each data point. The subscript D here highlights that this is an error between the data and the solution $u$, in order to distinguish this quantity from other error functions defined below. When measurement errors are independently normally distributed, $E_D(\alpha)$ will be the logarithm of the likelihood of the data (or more strictly, log (likelihood)), and minimizing equation (2.2) is equivalent to maximum likelihood estimation of the parameters. It is also possible to use this approach even if we cannot measure all of the components of the state vector $\hat{x}(t)$. In such a case, the norm in equation (2.2) is
just taken over the measured components. As long as we have a sufficient number of data points to ensure that \( E_D(\alpha) \) has a non-degenerate minimum, it may still be possible to estimate parameters successfully.

Observe that when \( f \) has linear dependence on its parameters, the error function given by equation (2.2) is a quadratic form. The minimum in such a case can be obtained efficiently in one step using algebraic methods such as QR decomposition (e.g. Lawson & Hanson 1974). This is much faster and more robust than the iterative optimization algorithms mentioned above. For ODEs this possibility seems irrelevant, however, since even if the vector field \( f \) depends linearly on the parameters \( \alpha \), the solution \( u(t) \) will depend nonlinearly on \( \alpha \). Nevertheless, we shall present below a new approach that makes use of linearity in parameters.

The most popular method of solving the differential equation is to use a standard numerical integration scheme such as fourth-order Runge–Kutta, possibly with adaptive step-size control (e.g. Press et al. 2002). Such an approach immediately encounters a major potential obstacle: we very rarely know the correct initial condition \( x(t_0) \) for all of the variables. Even if we are able to measure all of the components of \( x(t_0) \), such measurements will inevitably be subject to some error. Thus, in practice, we only have \( \hat{x}(t_0) = x(t_0) + \epsilon(t_0) \) where \( \epsilon(t_0) \) denotes the experimental error. If the differential equation is globally stable, and \( \epsilon(t_0) \) is small, the difference between \( \hat{x}(t_0) \) and \( x(t_0) \) may not be too significant. However, most models in the real world are nonlinear. In such a case, the dynamics can dramatically amplify small differences in the initial conditions. Attempting to estimate parameters for such systems using \( \hat{x}(t_0) \) as an initial condition leads to poor results.

\((c)\) Shooting

A better approach is to regard the initial condition \( x(t_0) \) as an additional set of unknown parameters which are incorporated in the minimization scheme. We thus regard the error function \( E(\alpha, x(t_0)) \) as depending on both \( \alpha \) and \( x(t_0) \). This type of approach appears to have first been tried by Bellman et al. (1967) and similar ideas appear in Swartz & Bremermann (1975). It is closely related to shooting methods for boundary value problems, including methods used for finding periodic orbits and other special solutions (e.g. Golub & Ortega 1992; Kuznetsov 1995). It can work well if data are plentiful and noise levels are low. However, if we only have a few time points, as is typical of cell and molecular biology datasets, \( E(\alpha, x(t_0)) \) can have a large number of local minima separated by steep peaks and ridges. As we shall see below, it is difficult to find the global minimum in such cases.

One possible extension is to use multiple shooting methods where the solution is broken down into a number of successive segments, with appropriate matching at the joints (e.g. Kuznetsov 1995; Timmer et al. 2000). This can improve results, but our experience with even moderately complex models is that it can suffer from similar problems to simple shooting.

\((d)\) Collocation methods

An alternative to using iterative numerical integration is to represent the solution globally using a set of convenient basis functions \( B_j : j = 0, \ldots, p \). A suitable choice is usually given by piecewise polynomials, usually of low order.
If we match values and derivatives at the joins between adjacent polynomials, globally smooth functions can be obtained, called splines. We can represent these as linear combinations of the form

$$u(t) = \sum_{j=0}^{p} b_j B_j(t).$$  \hfill (2.3)

A variety of different types of spline are available, but often the most convenient are B-splines (De Boor 1978; Gershenfeld 1999). Such splines have minimal support with respect to degree, smoothness and partition, and any other spline function of a given degree, smoothness and domain partition can be represented as a linear combination of B-splines of that same degree and smoothness over the same partition. B-splines have the great advantage that each $B_j$ has compact support so that evaluating $u(t)$ requires only summing over a small number of $B_j$'s. More precisely, if we partition the domain using the knots $\{s_j : j = 0, \ldots, p\}$, we can recursively define basis functions of increasing order by

$$B_j(t) = \begin{cases} 0 & t < s_j \\ \frac{1}{6h^3} (t - s_{j-2})^3 & s_{j-2} \leq t \leq s_{j-1} \\ \frac{1}{6} + \frac{1}{2h} (t - s_{j-1}) + \frac{1}{2h^2} (t - s_{j-1})^2 - \frac{1}{2h^3} (t - s_{j-1})^3 & s_{j-1} \leq t \leq s_j \\ \frac{1}{6} + \frac{1}{2h} (s_{j+1} - t) + \frac{1}{2h^2} (s_{j+1} - t)^2 - \frac{1}{2h^3} (s_{j+1} - t)^3 & s_j \leq t \leq s_{j+1} \\ \frac{1}{6h^3} (s_{j+2} - t)^3 & s_{j+1} \leq t \leq s_{j+2} \\ 0 & t \geq s_{j+2} \end{cases}$$

where $h$ is the spacing between the knots $s_j$. Observe that for $s_j \leq t \leq s_{j+1}$ the sum in equation (2.3) reduces to

$$u(t) = b_{j-1} B_{j-1}(t) + b_j B_j(t) + b_{j+1} B_{j+1}(t) + b_{j+2} B_{j+2}(t).$$
Furthermore, evaluating at the knot point $s_j$, we have

$$u(s_j) = b_{j-1}B_{j-1}(s_j) + b_jB_j(s_j) + b_{j+1}B_{j+1}(s_j) = \frac{1}{6}b_{j-1} + \frac{2}{3}b_j + \frac{1}{6}b_{j+1}.$$ 

This can be written in matrix form as

$$
\begin{pmatrix}
    u(s_{-1}) \\
    u(s_0) \\
    u(s_1) \\
    \vdots \\
    u(s_p) \\
    u(s_{p+1})
\end{pmatrix} = \frac{1}{6}
\begin{pmatrix}
    4 & 1 & 0 & \ldots & 0 \\
    1 & 4 & 1 & \ldots & 0 \\
    0 & 1 & 4 & \ldots & 0 \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    0 & \ldots & 1 & 4 & 1 \\
    0 & \ldots & 0 & 1 & 4
\end{pmatrix}
\begin{pmatrix}
    b_{-1} \\
    b_0 \\
    b_1 \\
    \vdots \\
    b_p \\
    b_{p+1}
\end{pmatrix},
$$

(2.4)

where $u(s_{-1})$ and $u(s_{p+1})$ are dummy values required to give a well-determined system. These determine the behaviour of $u$ at the boundaries and can, for instance, be chosen to set the second derivatives of $u$ at the boundaries to 0 (a so-called natural cubic $B$-spline). The matrix in (2.4) is explicitly invertible (though in practice one would solve equation (2.4) using a specialized LR decomposition). This shows that we can either parametrize the spline $u(t)$ using the coefficients $b_j$, or using its value at the knot points $u(s_0), u(s_1), \ldots, u(s_p)$. This makes it straightforward to obtain a spline that interpolates any particular set of points. It is also possible to generalize this derivation to irregularly spaced knot points, which can have significant advantages in some applications.

The $B_j$ are polynomials and hence using equation (2.3), we can easily differentiate $u$ and substitute the derivative into equation (1.1). In general, since we only take a finite number of basis functions, we cannot expect this to be satisfied exactly everywhere, i.e. for every $t$. Instead, we choose a finite number of so-called collocation points $r_k : k = 1, \ldots, q$, and require (1.1) to hold at these points, so that

$$
\sum_{j=-1}^{p+1} b_j \frac{dB_j}{dt}(r_k) = f \left( \sum_{j=-1}^{p+1} b_j B_j(r_k), r_k, \alpha \right),
$$

(2.5)

for all $k = 1, \ldots, q$. Note that the derivatives $dB_j/dt$ can be easily pre-computed for any given set of basis functions and collocation points, so that equation (2.5) represents a system of algebraic equations for $b = (b_{-1}, b_0, \ldots, b_{p+1})$, or for the values $u(s_0), u(s_1), \ldots, u(s_p)$ via equation (2.4). Note that equation (2.5) is independent of the particular choice of spline, or indeed other basis functions.

Given an appropriate choice of splines and collocation points, the system (2.5) is well determined and can be solved to yield the coefficients of the approximate solution of the differential equation. It turns out that this is equivalent to an appropriate implicit Runge–Kutta integration scheme. This approach is particularly useful for boundary value problems (Villadsen & Stewart 1967) and today forms the basis of standard packages such as AUTO for finding periodic orbits and bifurcation points. The basic principle is to use the Newton method or a similar root finder to solve equation (2.5), together with appropriate side
conditions. These ensure that \( u \) is periodic (or homo-/heteroclinic for certain global bifurcations) and in the case of finding local bifurcations has particular eigenvalues. In the case of bifurcations, it is necessary for the root finder to simultaneously vary both the coefficients \( b_j \) and the parameters \( \alpha \). This bears close resemblance to parameter estimation from data and suggests that collocation-based methods may also be useful in the latter problem.

However, there is an important difference between the two problems. When finding bifurcations, we simply want to solve a set of (nonlinear) equations made up of equation (2.5) and whatever side conditions we need to specify the bifurcation. In the case of parameter fitting, the side conditions are replaced by the minimization of \( E_D(\alpha) \) which has to be carried out simultaneously with the solution of equation (2.5). There are a number of possible approaches to this. Observe that if \( u \) is given by equation (2.3) then \( E_D \) has no direct dependence on \( \alpha \) but rather is a function of \( b \). From now on, we shall thus denote it by

\[
E_D(b) = \sum_{j=1}^{n} \left| \sum_{j=-1}^{p+1} b_j B_j(t_i) - \hat{x}(t_i) \right|^2.
\]

(i) \textit{Nested minimization and collocation}

This is closely related to shooting, except that instead of integrating the differential equation using a method such as fourth-order Runge–Kutta, we use the Newton method to solve equation (2.5) at each candidate value of the parameters \( \alpha \). This Newton solver is then nested inside an optimization method that iteratively minimizes \( E_D(b, \alpha) \). This is potentially very slow, and we are not aware of this method appearing in the published literature.

(ii) \textit{Simultaneous minimization and collocation}

It is possible to construct combinations of gradient-based minimization and Newton root solving which essentially simultaneously linearize both equation (2.5) and \( E_D(b) \). Given a reasonable initial guess, such an algorithm will rapidly converge to a minimum of \( E_D(b) \) that satisfies equation (2.5). The first such method appears to have been published by Baden & Villadsen (1982). Biegler (1984) independently initiated the application of collocation-based methods to dynamic optimization, which includes parameter estimation as a special case. His method uses sequential quadratic programming, a common algorithm for minimizing an objective function subject to equality constraints without requiring satisfaction of the constraints at each iteration. This scheme solves the exact constraint such as equation (2.5) once, and then at subsequent iterations, it uses only the linearization of the constraint. This is combined with a quadratic approximation to the objective function \( E_D(b) \) which is easily minimized subject to the linearized constraint. It can be shown that such an iteration converges quadratically to the desired minimum. Biegler’s algorithm has stimulated a range of variations and extensions (e.g. Tjoa & Biegler 1991; Esposito & Floudas 2000; Wang 2000; Li et al. 2005) and is widely used in the chemical engineering community.
(iii) **Dual minimization**

Observe that instead of using a root finder to solve equation (2.5), we could minimize the difference between the r.h.s. and l.h.s.

\[
E_M(b, \alpha) = \sum_{i=1}^{q} \left\| \sum_{j=1}^{p+1} b_j \frac{dB_j}{dt}(r_k) - f \left( \sum_{j=1}^{p+1} b_j B_j(r_k), r_k, \alpha \right) \right\|^2
\]

with respect to \( b = (b_{-1}, b_0, \ldots, b_{p+1}) \). We could also give a different weight to each component of the ODE (e.g. Ramsay et al. 2007). We can think of \( E_M(b, \alpha) \) as a measure of how well the spline \( u(t) \), defined by \( b \), satisfies the differential equation (1.1). This approach has the advantage that we can take a larger number \( q \) of collocation points. In that case, equation (2.5) will be over-determined, and no longer have a solution, but it is still possible to find a minimum of \( E_M(b, \alpha) \). Indeed, in the limit \( q \to \infty \), we have

\[
E_M(b, \alpha) = \int \left\| \sum_{j=1}^{p+1} b_j \frac{dB_j}{dt}(r) - f \left( \sum_{j=1}^{p+1} b_j B_j(r), r, \alpha \right) \right\|^2 dr.
\]

The minimization of such a function is often used in the derivation of various collocation schemes (e.g. Golub & Ortega 1992). Given either version of \( E_M(b, \alpha) \), now our problem is to simultaneously minimize both \( E_D(b) \) and \( E_M(b, \alpha) \) with respect to both \( b \) and \( \alpha \). This is most easily achieved by minimizing

\[
\tilde{E}(b, \alpha) = E_D(b) + \lambda E_M(b, \alpha).
\]

Here \( \lambda \) is a weighing factor that determines how much emphasis we place on the data and the model, respectively. This is a particular attraction of this approach, since if we have high confidence in the model but the data are very noisy, we can take \( \lambda \) large, and conversely if the measurement error for the data is low but the model is suspect, we can use a low \( \lambda \).

Brewer (2006) compares the Nelder–Mead simplex algorithm and simulated annealing in the minimization \( \tilde{E}(b, \alpha) \), with \( \lambda = 1 \) and \( E_M(b, \alpha) \) defined by equation (2.6). A more sophisticated optimization scheme has recently been presented by Ramsay et al. (2007) for the case of equation (2.7). Ramsay et al. (2007) also analyse the behaviour of their method in the limit \( \lambda \to \infty \).

An alternative algorithm, applicable when \( f \) is linear in parameters, was introduced in Brewer (2006) and is presented in §2c. This is motivated by the observation that if the data are observed without error, so that \( \hat{x}(t_i) = x(t_i) \), then we know that the rate of change of a solution at \( t_i \) is precisely \( f(x(t_i), t_i, \alpha) \). In such a case, it would be reasonable to replace \( E_M(b, \alpha) \) by

\[
\tilde{E}_M(b, \alpha) = \sum_{i=1}^{n} \left\| \sum_{j=1}^{p+1} b_j \frac{dB_j}{dt}(t_i) - f(\hat{x}(t_i), t_i, \alpha) \right\|^2.
\]

The advantage of this is that \( \tilde{E}_M(b, \alpha) \) is a quadratic form (i.e. a linear least squares problem) in \((b, \alpha)\). Since \( E_D(b) \) is also a quadratic form, the overall error \( E_D(b) + \lambda \tilde{E}_M(b, \alpha) \) is a linear least squares function that can be minimized using a single QR decomposition.

*Phil. Trans. R. Soc. A* (2008)
Of course, in general, the data are subject to measurement error. We can, however, still use a modified error function of the above type, with our best estimate \( \hat{u}(t_i) \) of the real measurement replacing \( \hat{x}(t_i) \) in the above formula. Motivated by the expectation–maximization methods used in the case of nuisance parameters or missing data (Dempster et al. 1977; Moon 1996), we alternatively minimize \( E_D(b) + \lambda E_M(b, \alpha) \) and generate a new estimate of the data. Full details are given below.

\((e)\) Derivative approximation methods

All of the methods presented so far use the least squares error function in equation (2.2) which determines the discrepancy between the observed data and a numerically computed solution of the differential equation. An alternative is to minimize the discrepancy between the r.h.s. and l.h.s. of the differential equation at a selected number of data points. This gives an error function like

\[
E_D(a) = \sum_{k=1}^{q} \left\| \frac{du}{dt}(r_k) - f(u(r_k), r_k, \alpha) \right\|^2.
\]

(2.8)

This relationship between minimizing \( E_D(a) \) and \( E_M(b, \alpha) \) under suitable conditions is discussed by Baden & Villadsen (1982). Observe the close similarity between \( E_D(a) \) and \( E_M(b, \alpha) \) in equation (2.6). In particular, if we chose an approximate solution \( u \) given by equation (2.3), then \( E_D(a) = E_M(b, \alpha) \). In such circumstances, minimizing \( E_D(a) \) will be equivalent to minimizing \( \tilde{E}(b, \alpha) \) in the limit \( \lambda \to \infty \).

Note, however, that \( E_D(a) \) has no direct dependence on the observed data \( \hat{x}(t_i) \). In the case of \( \tilde{E}(b, \alpha) \), for a finite \( \lambda \), such dependence is obtained through \( E_D(b) \). Alternatively, one can incorporate the data directly into \( E_D(a) \) by choosing the points \( u(r_k) \) in equation (2.8) to be exactly or approximately the data points \( \hat{x}(t_i) \). The advantage of this is that it can be done without the need to solve the differential equation (1.1). Instead, we simply need to estimate the derivative \( du/dt \) at the points of interest. This can be done by smoothing the data \( \hat{x}(t_i) \) using an appropriate local polynomial (or globally using a spline), and then differentiating the result.

This approach appears to have first been employed by van den Bosch & Hellinckx (1974), possibly inspired by collocation-based methods for parameter estimation in partial differential equations (Seinfeld 1969). Baden & Villadsen (1982) suggested an improvement and compared both schemes with the simultaneous minimization of \( E_D(b) \) and solution of equation (2.5), as described above. Swartz & Bremermann (1975) independently published an algorithm of this type and compared it with a shooting method. In contrast to van den Bosch & Hellinckx (1974) and Baden & Villadsen (1982) who used a global spline to smooth the data, Swartz & Bremermann (1975) employed local polynomials to either approximate or interpolate several successive data points, without any attempt to match such local fits at their intersections. Varah (1982) presented and evaluated an algorithm based on their ideas but again using a global spline smoothing method.

Finally, in a modern systems biology context, where observations are only available for a very restricted number of time points, Barenco et al. (2006) used Lagrangian interpolation between nearby data points to estimate the required
derivative. This allowed them to estimate both parameters in a simple model of gene expression and to estimate the activity of an unobserved transcription factor (p53) driving the system. This, in turn, allowed the prediction of previously unknown targets of p53.

3. A new efficient method for ODEs linear in parameters

(a) Linearity in parameters

Our experience is that direct minimization of $\hat{E}(b, \alpha)$ can be time-consuming (Brewer 2006). One would thus like to use specific features of a particular class of models to develop a more efficient algorithm. In particular, many models in physics, chemistry, engineering and biology are linear in their parameters. This is true for both models derived from underlying principles such as mass–action models in chemistry and cell biology, and for data-driven ‘black-box’ models built with general set of basis functions (e.g. radial basis functions). Recall that if a model is linear in its parameters, then the objective function is a quadratic form and a least squares estimate can be obtained in one step (i.e. non-iteratively) using standard linear algebra techniques such as QR decomposition (e.g. Lawson & Hanson 1974). This is much faster than the iterative minimization routines required for models that are nonlinear in parameters.

It seems difficult, however, to make use of this for ODE models, since even if the vector field $f(x, t, \alpha)$ depends linearly on the parameters $\alpha$, the solution $u(t)$ will typically exhibit nonlinear dependence. This makes it difficult to benefit from the linearity of $f$ for solution-based approaches employing $E_M$ or $\tilde{E}$.

On the other hand, if $f(x, t, \alpha)$ is linear in $\alpha$ then for fixed $u(r_k)$ in equation (2.8) the error function $E_\Delta(\alpha)$ is a quadratic form that can be minimized in the usual way using QR decomposition. In other words, equation (2.8) is a linear least squares problem in $\alpha$. This makes methods based on $E_\Delta(\alpha)$ particularly attractive for ODEs that are linear in parameters. However, as Baden & Villadsen (1982) point out, $E_\Delta(\alpha)$ is the wrong objective function from a likelihood point of view. Our aim here, therefore, is to use the similarity between $E_\Delta(\alpha)$ and $E_M(b, \alpha)$ highlighted above to derive an algorithm that approximately minimizes $\hat{E}(b, \alpha)$ making full use of the linearity of $f(x, t, \alpha)$ with respect to $\alpha$.

(b) Overview of the new algorithm

As already indicated above, if the data are known precisely, we can replace $E_M(b, \alpha)$ by $\hat{E}_M(b, \alpha)$ which is a linear least squares objective function. We can employ the same principle even with noisy data if we replace $\hat{x}(t_i)$ in $\hat{E}_M(b, \alpha)$ by our best available estimate of the noise-free data values. More generally, we do not need to restrict to just the observed time points $t_i$. In particular, if we have estimates $\hat{u}(r_k)$ of the variables at the collocation points $r_k$, we obtain the modified model error function

$$\hat{E}_M(b, \alpha, \hat{u}) = \sum_{k=1}^q \left\| \sum_{j=1}^{p+1} b_j \frac{dB_j}{dt}(r_k) - f(\hat{u}(r_k), r_k, \alpha) \right\|^2,$$

(3.1)
which yields the overall
\[ \hat{E}(b,\alpha,\hat{u}) = E_D(b) + \lambda \hat{E}_M(b,\alpha,\hat{u}). \]  
(3.2)

Motivated by expectation–maximization methods used in the case of nuisance parameters or missing data (Dempster et al. 1977; Moon 1996), we can attempt to fit the model by generating a sequence of better and better estimates \( \hat{u}^{(m)} \). At each stage, this estimate is substituted into \( \hat{E} \) which is minimized with respect to \( (b,\alpha) \). The resulting \( \hat{b} \) is used to generate a new estimate \( \hat{u}^{(m+1)} \) using equation (2.3). We thus alternate estimating the expected value of the nuisance parameter with a minimization of the parameter of interest. Note, however, that our algorithm differs from a conventional expectation–maximization method where we would only minimize \( \alpha \), whereas here we simultaneously minimize both \( \alpha \) and \( b \).

(c) Description of the algorithm
We iteratively generate a sequence of estimates \( (b^{(m)},\alpha^{(m)}) \) with \( \hat{u}^{(m)} \) given by
\[ \hat{u}^{(m)}(t) = \sum_{j=-1}^{n+1} b_j^{(m)} B_j(t). \]  
(3.3)

To do this, we proceed as follows.

(i) The iteration is initialized with a spline \( \hat{u}^{(0)} \) obtained by smoothing the data. More precisely, \( \hat{u}^{(0)} \) is given by equation (3.3) with \( b^{(0)} \) the minimum of \( E_D(b) \).

(ii) For each \( m = 1, 2, \ldots \), we obtain \( (b^{(m)},\alpha^{(m)}) \) from \( \hat{u}^{(m-1)} \) by minimizing \( \hat{E}(b,\alpha,\hat{u}^{(m-1)}) \) with respect to \( (b,\alpha) \). This is a linear least squares problem which is carried out in one step using QR decomposition.

(iii) We define \( \hat{u}^{(m)} \) from \( b^{(m)} \) using equation (3.3).

(iv) If for some preset tolerance \( \delta \) we have \( ||b^{(m)} - b^{(m-1)}|| < \delta \) then terminate, otherwise return to step (ii).

(d) Properties of the solution
When the iteration terminates, we have \( b^{(m)} = b^{(m-1)} \) to some pre-specified numerical tolerance. Thus
\[ \hat{E}_M(b^{(m)},\alpha,\hat{u}^{(m-1)}) = \hat{E}_M(b^{(m)},\alpha,\hat{u}^{(m)}) = E_M(b^{(m)},\alpha) \]
and hence
\[ \hat{E}(b^{(m)},\alpha,\hat{u}^{(m-1)}) = \hat{E}(b^{(m)},\alpha). \]
Since \( (b^{(m)},\alpha^{(m)}) \) minimizes \( \hat{E}(b,\alpha,\hat{u}^{(m-1)}) \) with respect to \( (b,\alpha) \), we see that \( \alpha^{(m)} \) is a minimum of \( \hat{E}(b^{(m)},\alpha) \) with respect to \( \alpha \). In general, it does not appear to be possible to ensure that \( \hat{E}(b,\alpha) \) is also minimized with respect to \( b \) but in practice the distinction between \( \hat{E}_M \) and \( E_M \) appears to have negligible effect.

(e) Implementation
The algorithm was implemented in C++, using the TNT library (Pozo 2004). A weight of \( \lambda = 1 \) and a stopping tolerance of \( \delta = 10^{-8} \) were used, unless otherwise stated. In evaluating the accuracy of the final spline to the model, we used the
integral form of $E_J = E_M(b, \alpha)$, as defined in equation (2.7). This was evaluated using Simpson’s rule with 10,000 steps (Press et al. 2002).

4. Numerical results

(a) A test model

As a specific example, we consider a simple four-component model of the core of p53 gene regulatory network; see appendix A and Brewer (2006). This ODE describes the behaviour of active ATM ($a$), p53 ($z$), active p53 ($x$) and MDM2 ($y$) after a cell experiences DNA damage

$$
\frac{da}{dt} = -D_{ATM}a,
$$

$$
\frac{dz}{dt} = p_{p53} - D_{p53}z - k_1yz - k_2az,
$$

$$
\frac{dx}{dt} = -D_{p53}x - k_1yx + k_2az \quad \text{and}
$$

$$
\frac{dy}{dt} = p_{MDM2} + k_3x - D_{MDM2}y - k_4ay.
$$

In total, the model depends linearly on its nine parameters $D_{ATM}$, $D_{MDM2}$, $D_{p53}$, $p_{MDM2}$, $p_{p53}$, $k_1$, $k_2$, $k_3$ and $k_4$. We generated a simulated dataset for the parameter values given in Table 4a using a fourth-order Runge–Kutta scheme with step size of $10^{-8}$ implemented in C++ (Press et al. 2002). Different sized datasets were created by sampling from this set at fixed intervals.

To simulate measurement noise, we added an independent Gaussian error to each data point. This had mean 0 and variance $\sigma^2$, with $\sigma$ a constant value for all data points. A number of different noise levels were used, ranging from $\sigma = 0$ to $\sigma = 0.06$. For each noise level, we report results averaged over 1000 independent realizations, each of which was fitted to the model.

(b) Results: shooting method using Nelder–Mead optimization

We implemented a shooting algorithm approach in C++ using a Runge–Kutta algorithm with adaptive step size control as the integrator (Fehlberg 1968; Cash & Karp 1990) and a Nelder–Mead simplex method as the optimization routine (Nelder & Mead 1965; Press et al. 2002). A starting simplex was constructed around an initial parameter estimate $P_0$ by

$$
P_i = P_0 + \zeta \hat{e}_i,
$$

where $\hat{e}_i$ is the unit vector in the $i$th coordinate direction. Four different choices of initial guess were tried, as shown in Table 1.

We used a stopping criterion based on the fractional range of the simplex

$$
\frac{|l_h - l_l|}{|l_h| + |l_l|} < \frac{1}{2} \eta,
$$

Phil. Trans. R. Soc. A (2008)
where $\eta$ is the required accuracy; $l_h$ is the highest value of the objective function among the vertices of the simplex; and $l_l$ is the lowest value.

Table 2 shows the results of this method for the four different initial starting points from Table 1. This demonstrates that the parameter space contains many local minima, and unless the algorithm is started close to the true value, it is difficult to recover the correct parameter estimates. Using the Powell minimization method (Acton 1990), instead of Nelder–Mead, produced similar results (data not shown). Multiple local minima are common when applying parameter estimation to ODE models (Esposito & Floudas 2000), especially when the model is nonlinear and there are a large number of parameters. In this case, using any kind of local method, including other approaches such as gradient- or Hessian-based methods, will not be sufficient unless the starting parameter estimates are close to their true values.

**(c) Results: shooting method using simulated annealing**

An alternative is to use a global minimization method such as simulated annealing (Metropolis et al. 1953; Kirkpatrick et al. 1983; Kirkpatrick 1984; Gershenfeld 1999 or Press et al. 2002), Markov Chain Monte Carlo (e.g. Gilks et al. 1996) or genetic algorithms (e.g. Gershenfeld 1999). Such methods all have some possibility of moving to a worse solution during a systematic search of the parameter space and hence are able to escape out of local minima. Simulated annealing is probably the oldest and best established of these methods. This global minimization method slowly ‘cools’ the system, where the
‘temperature’ determines the probability that a step in parameter space that worsens the error is accepted. Generally, the step is determined by taking a random point from a Gaussian distribution with the mean set at the current point but here a refinement is implemented that uses the Nelder–Mead algorithm to propose the step (Cardoso et al. 1996; Kvasnicka & Pospichal 1997; Torres et al. 1997; Press et al. 2002). The cooling scheme is an important factor in the effectiveness of the algorithm. Here we employ the one recommended by Press et al. (2002), taking \( T = T_0(1 - k/K)^4 \) where \( T_0 \) is the initial temperature, \( k \) is the total number of moves so far and \( K \) is the estimated number of moves required.

The above scheme was implemented in C++. A number of different choices of cooling parameters (starting temperature and rate of cooling) and initial conditions were applied to the 1052 time point dataset with no measurement error (\( \sigma = 0 \)) using a number of different G5-based computers with processor speeds between 1.6 and 2.0 GHz (tables 3 and 5). Convergence to the global minimum only occurred in approximately one-quarter of the runs and these successful runs all started with the same initial parameters. The duration of the

Table 3. Parameter estimation using shooting with simplex simulated annealing. (Initial temperatures of 10, 100 and 1000 correspond to initial acceptance percentages of 52, 68 and 71% (based on 1000 proposed steps after an initial transient of 100 steps). The total number of steps was chosen for its computational feasibility: \( 10^6 \) steps can require several days of computer time.)

<table>
<thead>
<tr>
<th>initial point</th>
<th>starting temperature ((T_0))</th>
<th>estimated count ((K))</th>
<th>no. of iterations</th>
<th>least squares score</th>
<th>approx. duration (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>10</td>
<td>10^6</td>
<td>1 000 000</td>
<td>6.36 \times 10^{-4}</td>
<td>22</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10^6</td>
<td>1 000 000</td>
<td>6.36 \times 10^{-4}</td>
<td>49</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10^7</td>
<td>10 000 000</td>
<td>6.36 \times 10^{-4}</td>
<td>25</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>10^7</td>
<td>10 000 000</td>
<td>0.237</td>
<td>276</td>
</tr>
<tr>
<td>B</td>
<td>100</td>
<td>10^6</td>
<td>1 000 000</td>
<td>6.36 \times 10^{-4}</td>
<td>52</td>
</tr>
<tr>
<td>B</td>
<td>100</td>
<td>10^6</td>
<td>1 000 000</td>
<td>6.11</td>
<td>276</td>
</tr>
<tr>
<td>B</td>
<td>100</td>
<td>10^6</td>
<td>1 000 002</td>
<td>16.4</td>
<td>369</td>
</tr>
<tr>
<td>B</td>
<td>100</td>
<td>10^7</td>
<td>10 000 000</td>
<td>11.4</td>
<td>573</td>
</tr>
<tr>
<td>B</td>
<td>1000</td>
<td>10^6</td>
<td>1 000 000</td>
<td>110</td>
<td>315</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>10^6</td>
<td>1 000 001</td>
<td>5.22</td>
<td>—</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>10^6</td>
<td>1 000 001</td>
<td>5.39</td>
<td>104</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>10^6</td>
<td>1 000 000</td>
<td>5.48</td>
<td>60</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>10^6</td>
<td>1 000 000</td>
<td>5.77</td>
<td>124</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>10^7</td>
<td>10 000 000</td>
<td>5.98</td>
<td>62</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>10^7</td>
<td>10 000 000</td>
<td>5.37</td>
<td>444</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>10^6</td>
<td>1 000 000</td>
<td>4.28</td>
<td>56</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>10^6</td>
<td>1 000 000</td>
<td>5.50</td>
<td>253</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>10^6</td>
<td>1 000 001</td>
<td>5.67</td>
<td>218</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>10^6</td>
<td>1 000 002</td>
<td>5.71</td>
<td>217</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
<td>10^6</td>
<td>1 000 000</td>
<td>12.4</td>
<td>428</td>
</tr>
</tbody>
</table>
runs was consistently high, almost always taking more than a day to converge. In order to provide the best chance of success, the test was performed with a large dataset; a reduction in the dataset would render the parameter space more complex and make convergence to the optimal solution harder.

We conclude that simulated annealing can give good parameter estimates, but requires a lot of time tuning the minimization algorithm for success. Even when it does work, it is slow and is at the limit of practicality on current popular hardware. These conclusions will apply to any global minimization method that relies on single shooting to determine the error function. Multiple shooting (e.g. Kuznetsov 1995; Timmer et al. 2000) can help to regularize the parameter space and generally provides a more robust algorithm, but in our experience still retains many of the same problems as ordinary shooting.

(d) Results: novel collocation scheme for ODEs linear in parameters

Finally, we applied the algorithm presented in §3 for a variety of combinations of noise levels and sizes of dataset. When the dataset is relatively large (1052 time points), we obtain accurate parameter estimates using 22 splines \( p=19 \). In this case, except for \( D_{p53} \), all estimated parameters are within 0.1% of their true values (table 4b). The solution splines are virtually indistinguishable from the true time course (data not shown) and satisfy the model to a high degree of accuracy \( E_f = 2.38 \times 10^{-5} \). The algorithm took approximately 2 min on a 2.0 GHz G5 Power Macintosh when \( p=19 \), \( n=1052 \), \( q=n \), \( \lambda = 1 \) and \( \sigma = 0.06 \). This is significantly faster than the shooting methods above. Furthermore, the speed improves considerably as the amount of data is reduced (12 s required when \( n=106 \)).

As the size of the dataset is decreased, the accuracy of the estimates also declines (figure 1). This occurs because the larger the dataset, the more constrained the spline is and hence the closer it will be to the ‘true’ solution and the more accurate the estimates will be. However, the loss of accuracy is minimal down to approximately \( n=150 \). Even with very small amounts of data, the estimates are still reasonably accurate; when \( n=14 \) the error is less than 7% which is perfectly usable in many systems biology contexts. This behaviour was consistent for each of the parameter estimates, but there were orders of magnitude differences in the error, ranging from less than \( 10^{-5} \) to 15% when \( n=14 \) (see figure 7 in appendix B). This may reflect the relative contributions a

Table 4. Evaluation of the algorithm presented in §3. (a) True parameter values. (b) Percentage error for parameter estimates from a 1052 time point dataset with \( p=19 \), \( q=1052 \), \( \lambda = 1 \) and \( \omega=0.464 \). (c) Parameter estimates from a six time point dataset when \( p=19 \), \( q=29 \) and \( \omega=0.464 \), where \( \omega = \lambda q/n \).

<table>
<thead>
<tr>
<th></th>
<th>( D_{ATM} )</th>
<th>( D_{MDM2} )</th>
<th>( D_{p53} )</th>
<th>( p_{MDM2} )</th>
<th>( p_{p53} )</th>
<th>( k_1 )</th>
<th>( k_2 )</th>
<th>( k_3 )</th>
<th>( k_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>0.05</td>
<td>0.18</td>
<td>0.041</td>
<td>0.01</td>
<td>0.52</td>
<td>1.42</td>
<td>0.39</td>
<td>2.50</td>
<td>0.75</td>
</tr>
<tr>
<td>(b)</td>
<td>0%</td>
<td>0.0335%</td>
<td>0.4221%</td>
<td>0.1684%</td>
<td>0.0562%</td>
<td>0.0835%</td>
<td>0.0604%</td>
<td>0.0172%</td>
<td>0.00160%</td>
</tr>
<tr>
<td>(c)</td>
<td>0.0500</td>
<td>0.176</td>
<td>0.0391</td>
<td>0.0954</td>
<td>0.525</td>
<td>1.44</td>
<td>0.394</td>
<td>2.47</td>
<td>0.741</td>
</tr>
</tbody>
</table>
parameter has on the model solution. We used $q = n$ throughout as preliminary experiments showed that when there was no error in the data, this consistently produced good results in the minimum amount of processing time.

We next added varying levels of noise to the data, and found that the algorithm continues to perform well (figure 2). The mean estimate generally occurs close to the true value and is always within one standard error. As $\sigma$ increases, the estimates shift away from the true parameter value and this shift is significantly greater in some parameters than others. As the error in the data increases, the positional constraints of the spline (i.e. the data points) less resemble the true solution and so the optimal spline is likely to deviate, which in turn produces less accurate estimates.

The performance of the algorithm depends on a number of adjustable factors, including $p$, $q$ and $\lambda$. It is beyond the scope of this paper to look at these factors in detail (Brewer 2006), but here we will briefly examine two situations where their optimization is beneficial. When the amount of error in the data is large, it is no longer appropriate to give equal weight to the spline being close to the data and the spline satisfying the model. This is because the solution spline is unable to represent the model solution accurately and so poor parameter estimates will result. More weight can be placed on the spline satisfying the model by increasing the number of collocation points and/or increasing $\lambda$. This gives an improvement in the spline quality (figure 3) and hence the parameter estimates. Increasing the number of collocation points has the additional benefit of spreading the positions where the model needs to be satisfied, which is important where the amount of data is small. There is a limit to how much additional weight can be used: if it is too large, the procedure fails to converge. In this case, the solution spline moves away from the data points, becoming a worse estimate of the model solution at each iteration. This occurs because the data no longer have a strong enough

Figure 1. The relationship between the amount of data and the accuracy of the parameter estimates. This plot shows the accuracy when applied to datasets with between 14 (the minimum possible in this situation) and 1052 time points, with $p = 19$, $q = n$, $\lambda = 1$. Plots for individual parameters are given in figure 7 in appendix B.
constraining influence on the spline. When there is a small amount of data, it is difficult to get reasonable parameter estimates, but with the appropriate optimization of the factors, it is possible to get good results (table 4c and figure 4). At such low amount of data, convergence is extremely sensitive to the factor values.

5. Discussion

Estimation of parameters in ODE models is of considerable importance in many modelling fields. Increasingly in systems biology, this needs to be done for very short time courses with high levels of noise on the data. Traditional algorithms are poorly suited to such problems.
We have presented a summary of the various approaches to parameter fitting in ODEs. We have then introduced a new algorithm for the case of models linear in parameters. It employs a spline-based collocation scheme and alternates linear least squares minimization steps with repeated estimates of the noise-free values of the variables. This has the advantage that fast, established linear algebra solvers can be used to provide optimal parameter values. The proposed procedure also avoids the problems of the model becoming stiff which can hamper shooting-based methods, in particular, when applied to models that are linear in their parameters. Additionally, the proposed procedure is effective at dealing with large-scale models and does not require the estimation of initial conditions.

Figure 3. The relationship between model error \( E_j \) and (a) the number of collocation points \( q \); and (b) the ratio \( \omega = \lambda q / n \). Results are based on 1000 independent noise realizations with \( p = 19 \), \( \sigma = 0.06 \) and \( n = 106 \). In (a) we have \( \lambda = 1 \) and in (b) \( q = 500 \). Error bars indicate standard deviations.

We have then introduced a new algorithm for the case of models linear in parameters. It employs a spline-based collocation scheme and alternates linear least squares minimization steps with repeated estimates of the noise-free values of the variables. This has the advantage that fast, established linear algebra solvers can be used to provide optimal parameter values. The proposed procedure also avoids the problems of the model becoming stiff which can hamper shooting-based methods, in particular, when applied to models that are linear in their parameters. Additionally, the proposed procedure is effective at dealing with large-scale models and does not require the estimation of initial conditions.

*Phil. Trans. R. Soc. A* (2008)
We have shown that the proposed procedure produces reasonable estimates, even at low amounts of data and quite high amounts of error. The accuracy of the proposed procedure depends on how close the intermediary spline is to the true model solution. At low amounts of data and high error, there is less valid information to constrain the spline and so the spline accuracy and hence the estimates suffer. By optimizing the algorithm factors, in particular the weight placed on the spline satisfying the model, it is possible to significantly improve the accuracy of the spline and hence the estimates.

In comparison with single shooting methods, there are many advantages to our new algorithm. Firstly, it is more accurate in converging to reasonable parameter estimates across a wide range of dataset sizes and amounts of error. Many single shooting global optimization methods rely on probabilistic steps but this can often result in inconsistent results. Secondly, our procedure is considerably faster. This arises because the problem is linearized so that efficient linear algebra solvers can be used and by using splines the model has been effectively discretized, negating the need for costly integration. Also, our procedure is simple, with only three key factors that can be varied. In comparison, simulated annealing requires numerous algorithm parameters to be optimized. Finally, this technique neither requires the estimation of parameter values to seed the optimization nor the initial conditions of the dynamic system. This simplifies the requirements to get reasonable parameter estimates, which are of particular concern at low amounts of data or when not much is known about the system in advance.

Despite our procedure being limited to models that are linear in their parameters, it is still a fast and useful tool that is applicable in many areas of modelling. Furthermore, there is potential for this method to be applied to simplifications of more complex problems to provide good initial estimates that can then be refined using a more complex optimization methods applied to the full model.

Figure 4. The solution spline for active p53 produced by the algorithm on a small amount of data with $n=6$ and either $p=19$, $q=29$ and $\omega = 0.464$ or $p=11$, $q=14$ and $\omega = 1.09$. 
Appendix A. p53 model

p53 is a tumour-suppressor and has been described as the ‘guardian of the genome’ (Lane 1992). It is part of a complex and extensive gene regulatory network that integrates a variety of stress signals to produce a range of effects including apoptosis, growth arrest and DNA damage repair. Of particular importance is p53’s role in the decision to commence apoptosis, which is not well understood. p53 is known to play a vital role in preventing cancer; p53 is dysfunctional in the majority of cancer types (Soussi et al. 2000) and more than 18,000 different p53 mutations have been found in cancers (Bode & Dong 2004).

A simple model is proposed that includes the main protein regulatory interactions of the p53 network (figure 5). The following interactions are modelled: through phosphorylation, active ATM enables the stabilization and hence activation of p53 ($k_2$); ATM phosphorylates MDM2 compromising MDM2’s ability to ubiquitinate and bind p53, hence ATM increases the rate at which MDM2 is inactivated/degraded ($k_4$); active p53 transcribes MDM2 ($k_3$); and MDM2 encourages the degradation of both forms of p53 through ubiquitination and also prevents p53 acting as a transcription factor by binding ($k_1$). Active ATM drives the system and it is assumed that at time $t=0$ the active ATM level has been ‘kicked’ to a value away from equilibrium and this decays exponentially according to the rate constant, $D_{ATM}$. The model ODEs are
The following simplifications have been made.

— The path containing CHK2 was removed. This is because ATM→CHK2→
p53 duplicates the behaviour of the more direct ATM→p53 and so the effect
of the CHK2 pathway can be included in the interaction between active
ATM and p53.
— Once MDM2 has bound to ARF or become inactivated through phosphoryl-
ation, it is removed from the system (in effect degraded). This means that
only one component of MDM2 (the active form) is required.
— Active ATM is the only protein that can convert p53 into its active state and
there is no ‘basal’ rate of activation.
— If MDM2 interacts with both inactive p53 and active p53, then it degrades
them at an equal rate.
— p53 forms a tetramer when activated before it can perform its function as a
transcription factor. The details of this mechanism are ignored.

Appendix B. Additional plots and tables
Table 5. The results from simulated annealing with downhill simplex parameter estimation. A range of initial temperatures, estimated counts and initial points were used.

<table>
<thead>
<tr>
<th>initial point</th>
<th>initial temp</th>
<th>estimated count ((K))</th>
<th>no. of iterations</th>
<th>least squares value</th>
<th>(D_{\text{ATM}})</th>
<th>(D_{\text{MDM}})</th>
<th>(p_{\text{MDM2}})</th>
<th>(p_{\text{p53}})</th>
<th>(k_1)</th>
<th>(k_2)</th>
<th>(k_3)</th>
<th>(k_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true parameter values</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td>0.18</td>
<td>0.041</td>
<td>0.01</td>
<td>0.52</td>
<td>1.42</td>
<td>0.39</td>
<td>2.50</td>
</tr>
<tr>
<td>(b) 10 10⁶</td>
<td>1 000 000</td>
<td>6.36 \times 10^{-4}</td>
<td>0.0500</td>
<td>0.200</td>
<td>0.0445</td>
<td>0.0187</td>
<td>0.521</td>
<td>1.42</td>
<td>0.391</td>
<td>2.53</td>
<td>0.755</td>
<td></td>
</tr>
<tr>
<td>(b) 10 10⁷</td>
<td>1 000 000</td>
<td>6.36 \times 10^{-4}</td>
<td>0.0500</td>
<td>0.200</td>
<td>0.0445</td>
<td>0.0187</td>
<td>0.521</td>
<td>1.42</td>
<td>0.391</td>
<td>2.53</td>
<td>0.755</td>
<td></td>
</tr>
<tr>
<td>(b) 10 10⁸</td>
<td>1 000 000</td>
<td>0.288</td>
<td>0.0446</td>
<td>0.215</td>
<td>0.0617</td>
<td>0.0119</td>
<td>0.522</td>
<td>1.39</td>
<td>0.376</td>
<td>2.66</td>
<td>0.698</td>
<td></td>
</tr>
<tr>
<td>(b) 10 10⁷</td>
<td>1 000 000</td>
<td>6.36 \times 10^{-4}</td>
<td>0.0500</td>
<td>0.200</td>
<td>0.0445</td>
<td>0.0187</td>
<td>0.521</td>
<td>1.42</td>
<td>0.391</td>
<td>2.53</td>
<td>0.755</td>
<td></td>
</tr>
<tr>
<td>(b) 10 10⁸</td>
<td>1 000 000</td>
<td>0.237</td>
<td>0.0553</td>
<td>0.191</td>
<td>0.0260</td>
<td>0.0257</td>
<td>0.519</td>
<td>1.45</td>
<td>0.406</td>
<td>2.42</td>
<td>0.800</td>
<td></td>
</tr>
<tr>
<td>(b) 100 10⁶</td>
<td>1 000 000</td>
<td>6.36 \times 10^{-4}</td>
<td>0.0500</td>
<td>0.200</td>
<td>0.0445</td>
<td>0.0187</td>
<td>0.521</td>
<td>1.42</td>
<td>0.391</td>
<td>2.53</td>
<td>0.755</td>
<td></td>
</tr>
<tr>
<td>(b) 100 10⁷</td>
<td>1 000 000</td>
<td>6.11</td>
<td>0.0505</td>
<td>39.6</td>
<td>835</td>
<td>-289</td>
<td>589</td>
<td>107</td>
<td>417</td>
<td>10 600</td>
<td>2780</td>
<td></td>
</tr>
<tr>
<td>(b) 100 10⁸</td>
<td>1 000 002</td>
<td>16.4</td>
<td>1810</td>
<td>1110</td>
<td>1100</td>
<td>469</td>
<td>359</td>
<td>-1300</td>
<td>-208</td>
<td>-340</td>
<td>-561</td>
<td></td>
</tr>
<tr>
<td>(b) 100 10⁷</td>
<td>1 000 000</td>
<td>11.4</td>
<td>1870</td>
<td>652</td>
<td>1390</td>
<td>157</td>
<td>758</td>
<td>232</td>
<td>-141</td>
<td>1520</td>
<td>1870</td>
<td></td>
</tr>
<tr>
<td>(b) 1000 10⁶</td>
<td>1 000 000</td>
<td>11.0</td>
<td>1640</td>
<td>517</td>
<td>1520</td>
<td>115</td>
<td>565</td>
<td>-852</td>
<td>436</td>
<td>-8010</td>
<td>34 800</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁶</td>
<td>1 000 001</td>
<td>5.22</td>
<td>1810</td>
<td>1110</td>
<td>1100</td>
<td>469</td>
<td>359</td>
<td>-1300</td>
<td>-208</td>
<td>-340</td>
<td>-561</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁷</td>
<td>1 000 001</td>
<td>5.39</td>
<td>0.0534</td>
<td>626</td>
<td>1160</td>
<td>313</td>
<td>307</td>
<td>-1300</td>
<td>235</td>
<td>1820</td>
<td>545</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁸</td>
<td>1 000 000</td>
<td>5.48</td>
<td>0.0534</td>
<td>626</td>
<td>1160</td>
<td>313</td>
<td>307</td>
<td>-1300</td>
<td>235</td>
<td>1820</td>
<td>545</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁷</td>
<td>1 000 000</td>
<td>5.77</td>
<td>0.0534</td>
<td>626</td>
<td>1160</td>
<td>313</td>
<td>307</td>
<td>-1300</td>
<td>235</td>
<td>1820</td>
<td>545</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁸</td>
<td>1 000 000</td>
<td>5.48</td>
<td>0.0455</td>
<td>1540</td>
<td>1590</td>
<td>666</td>
<td>806</td>
<td>-690</td>
<td>559</td>
<td>0.0208</td>
<td>13.7</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁷</td>
<td>1 000 000</td>
<td>5.48</td>
<td>0.0505</td>
<td>945</td>
<td>2500</td>
<td>597</td>
<td>604</td>
<td>-2970</td>
<td>418</td>
<td>3.28</td>
<td>696</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁸</td>
<td>1 000 000</td>
<td>5.77</td>
<td>0.0505</td>
<td>945</td>
<td>2500</td>
<td>597</td>
<td>604</td>
<td>-2970</td>
<td>418</td>
<td>3.28</td>
<td>696</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁷</td>
<td>1 000 000</td>
<td>5.98</td>
<td>0.0505</td>
<td>1490</td>
<td>1240</td>
<td>619</td>
<td>567</td>
<td>-666</td>
<td>526</td>
<td>-1340</td>
<td>-197</td>
<td></td>
</tr>
<tr>
<td>(c) 10 10⁸</td>
<td>1 000 000</td>
<td>5.37</td>
<td>0.0501</td>
<td>2160</td>
<td>112</td>
<td>3500</td>
<td>56.1</td>
<td>-28.7</td>
<td>43.8</td>
<td>-29 200</td>
<td>-2160</td>
<td></td>
</tr>
<tr>
<td>(d) 10 10⁶</td>
<td>1 000 000</td>
<td>4.28</td>
<td>0.0519</td>
<td>1870</td>
<td>-333</td>
<td>726</td>
<td>143</td>
<td>1650</td>
<td>192</td>
<td>-2100</td>
<td>-1100</td>
<td></td>
</tr>
<tr>
<td>(d) 10 10⁷</td>
<td>1 000 000</td>
<td>5.50</td>
<td>0.0497</td>
<td>1650</td>
<td>1490</td>
<td>360</td>
<td>960</td>
<td>-177</td>
<td>579</td>
<td>191</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td>(d) 10 10⁸</td>
<td>1 000 001</td>
<td>5.67</td>
<td>0.0505</td>
<td>970</td>
<td>1830</td>
<td>595</td>
<td>526</td>
<td>-2000</td>
<td>403</td>
<td>-413</td>
<td>547</td>
<td></td>
</tr>
<tr>
<td>(d) 10 10⁶</td>
<td>1 000 002</td>
<td>5.71</td>
<td>0.0484</td>
<td>1720</td>
<td>1750</td>
<td>1200</td>
<td>296</td>
<td>-2311</td>
<td>247</td>
<td>-4660</td>
<td>-360</td>
<td></td>
</tr>
<tr>
<td>(d) 10 10⁸</td>
<td>1 000 000</td>
<td>12.4</td>
<td>1840</td>
<td>-44</td>
<td>-1973</td>
<td>-23.9</td>
<td>136</td>
<td>4660</td>
<td>-848</td>
<td>-143</td>
<td>314</td>
<td></td>
</tr>
</tbody>
</table>
Figure 6. The effect of increasing measurement error on the accuracy of parameter estimates. Results are based on 1000 independent noise realizations with $p=19$, $q = n = 106$, $\lambda = 1$. The error bars indicate the standard error of the parameter estimates. A $\sigma$ of 0.06 corresponds to approximately 75% relative error for active p53 and 12% for the other components. (a) $D_{\text{ATM}}$, (b) $D_{\text{MDM2}}$, (c) $D_{\text{p53}}$, (d) $p_{\text{MDM2}}$, (e) $p_{\text{p53}}$, (f) $k_1$, (g) $k_2$, (h) $k_3$ and (i) $k_4$. 

*Phil. Trans. R. Soc. A* (2008)
Figure 7. The relationship between the parameter estimates produced by the method in §3 and the amount of data \((p = 19, q = n, \text{ and } \lambda = 1)\). (a) \(D_{\text{ATM}}\), (b) \(D_{\text{MDM2}}\), (c) \(D_{p53}\), (d) \(p_{\text{MDM2}}\), (e) \(p_{p53}\), (f) \(k_1\), (g) \(k_2\), (h) \(k_3\) and (i) \(k_4\).
References


