Efficient Gradient Computation for Dynamical Models

B. Sengupta[*], K.J. Friston[†] and W.D. Penny[‡]

Wellcome Trust Centre for Neuroimaging
Institute of Neurology, University College London
12 Queen Square, London WC1N 3BG, UK

Abstract

Data assimilation is a fundamental issue that arises across many scales in neuroscience – ranging from the study of single neurons using single electrode recordings to the interaction of thousands of neurons using fMRI. Data assimilation involves inverting a generative model that can not only explain observed data but also generate predictions. Typically, the model is inverted or fitted using conventional tools of (convex) optimization that invariably extremise some functional – norms, minimum descriptive length, variational free energy, etc. Generally, optimisation rests on evaluating the local gradients of the functional to be optimised. In this paper, we compare three different gradient estimation techniques that could be used for extremising any functional in time – (i) finite differences, (ii) forward sensitivities and a method based on (iii) the adjoint of the dynamical system. We demonstrate that the first-order gradients of a dynamical system, linear or non-linear, can be computed most efficiently using the adjoint method. This is particularly true for systems where the number of parameters is greater than the number of states. For such systems, integrating several sensitivity equations – as required with forward sensitivities – proves to be most expensive, while finite-difference approximations have an intermediate efficiency. In the context of neuroimaging, adjoint based inversion of dynamical causal models (DCMs) can, in principle, enable the study of models with large numbers of nodes and parameters.

Introduction

An important goal of systems neuroscience is to integrate empirical data from various neuroimaging modalities with biologically informed models that describe the underlying generative processes. Here, the data to be explained are for example M/EEG or fMRI recordings made while subjects perform various experimental tasks, and the underlying neurodynamic processes are framed in terms

[*]b.sengupta@ucl.ac.uk
[†]k.friston@ucl.ac.uk
[‡]w.penny@ucl.ac.uk

of differential equations describing activity in neural masses, mean fields, or neural fields [1, 2, 3].

Considerable insight can be gained from studying the emergent properties of such neurodynamic processes. These can then be qualitatively compared with empirical data, allowing consilience among multiple levels of description [4, 5, 6]. An alternative approach is to directly fit neurodynamical models to neuroimaging data using standard model fitting procedures from statistics and machine learning [7, 8]. Differences in the generative processes induced by experimental manipulations can then be associated with changes in underlying brain connectivity. One example of such an approach is Dynamic Causal Modelling (DCM) [1] which fits differential equation models to neuroimaging data using a variational Bayesian scheme [9].

More generally, in the statistics and machine learning literature various methods have been employed to fit differential equations to data, from maximum likelihood approaches [10] to Bayesian sampling algorithms [11, 12]. The majority of these convex optimisation approaches involve computing the gradient; the change in the cost function produced by a change in model parameters. This gradient is then combined with information from line searches (e.g., Wolfe's conditions) or methods involving a Newton, quasi-Newton (low-rank) or Fisher information based curvature estimators to update model parameters [7, 13, 14]. The main computational bottleneck in these algorithms is the computation of the gradient (or the curvature) of the parametric cost function. This motivates the search for efficient methods to evaluate gradients.

This paper compares three different methods for computing gradients, and studies the conditions under which each is preferred. The first is the Finite Difference (FD) method, which is simplest and most general method – and is currently used in DCM. The second is the Forward Sensitivity (FS; also known as tangent linear) method, which has previously been proposed in the context of modelling fMRI time series [15]. The third is the Adjoint Method (AM) which has previously been used in the context of dynamical systems theory [16], weather forecasting [17], image registration [18] and single-neuron biophysics [19].

The paper is structured as follows – the methods section describes each approach including a mathemtical derivation of the adjoint method. Examples of the FS and AM updates are then provided for the case of simple Euler intergration. The results section reports numerical simulations that disclose the scaling characteristics of each method. Simulations are provided for a linear dynamical and weakly-coupled oscillator systems. We conclude with a discussion of the relative merits of each method.

Methods

We consider dynamical systems of the form

$$\begin{aligned} \dot{x} &= f(x,p) \\ j(x,p) &= -\frac{1}{2}\left\|y - g(x,p)\right\|^2 \end{aligned} \qquad (1)$$

where x is a state variable, the dot notation denotes a time derivative $\frac{dx}{dt}$, t is time, $f(\cdot)$ is the flow equation (dynamics), and p are model parameters.

The model produces a prediction via an observation function $g(x, p)$ and an instantaneous cost function $j(x, p)$ measures the squared difference from data points y. The total cost is then given by the integral up to time point T

$$J(p) = \int_0^T j(x,p)dt \tag{2}$$

We consider three methods for computing the gradient $\frac{dJ}{dp}$.

Finite Difference Method

The (one-sided) finite difference approximation to the gradient is then

$$\frac{dJ}{dp_i} = \frac{J(p+\delta_i) - J(p)}{\delta_i} \tag{3}$$

where δ_i denotes a small change (generally, $\sqrt{\epsilon}$ where ϵ is the machine epsilon) to the ith parameter. The error in the computation of this gradient is of order δ_i. The computation of $\frac{dJ}{dp}$ requires $P+1$ runs of the integration process, one for each model parameter. It is also possible to use central differences

$$\frac{dJ}{dp_i} = \frac{J(p+\delta_i) - J(p-\delta_i)}{2\delta_i} \tag{4}$$

which has an error of order δ_i^2 but requires $2P+1$ runs of the integration process. Variations on the vanilla FD approach are discussed in [7, 20].

Forward Sensitivity Method

The original dynamical model (Eqn. 1) can be implicitly differentiated w.r.t parameters to give

$$\frac{d\dot{x}}{dp} = \frac{\partial f}{\partial x}\frac{dx}{dp} + \frac{\partial f}{\partial p} \tag{5}$$

If the state variables are of dimension D and the parameters of dimension P then the quantity $\frac{d\dot{x}}{dp}$ is a $D \times P$ matrix, which can be vectorised to form a new flow function. This forms a new dynamical system of dimension $D \times P$ that can then be integrated using any numerical method to produce $\frac{dx}{dp}$ as a function of time. The forward sensitivity approach has been known since the publication of Gronwall's theorem [21]. The cost gradient is then given by accumulating the sensitivity derivative $\frac{dx}{dp}$ over time according to:

$$\begin{aligned} \frac{dJ}{dp} &= \int_0^T \frac{dj}{dp}dt \\ \frac{dj}{dp} &= \frac{\partial j}{\partial x}\frac{dx}{dp} + \frac{\partial j}{\partial p} \\ &= \frac{\partial j}{\partial g}\frac{\partial g}{\partial x}\frac{dx}{dp} + \frac{\partial j}{\partial g}\frac{\partial g}{\partial p} \end{aligned} \tag{6}$$

Euler Example

This section illustrates the FS approach first-order Euler integration of the dynamics

$$x_n = x_{n-1} + \tau f(x_{n-1}, p) \tag{7}$$

at discrete times $t(n)$. The FS method is based on differentiating this equation to give

$$\frac{dx_n}{dp} = \frac{dx_{n-1}}{dp} + \tau \left[\frac{\partial f}{\partial x_{n-1}} \frac{dx_{n-1}}{dp} + \frac{\partial f}{\partial p} \right] \tag{8}$$

This method is illustrated in Figure 1 where the solid path indicates a trajectory of points x_n for a dynamical system with parameters p and the dotted path indicates the trajectory $\underline{x}_n$ for the same dynamical system but with parameters $\underline{p} = p + \delta_i$. The dotted path can be obtained from the solid path via the total derivative $\frac{dx_n}{dp_i}$ in the direction of the perturbation, δ_i. The FS method provides a method for computing this derivative. Under a first order Euler approach for integrating the dynamics, this is implemented using the above recursion.

Because the perturbed path (dotted in Figure 1) can be reached from the original trajectory via the total derivative $\frac{dx_n}{dp}$, there is no need to separately integrate the system with parameters $\underline{p}$. Geometrically, the points $\underline{x}_n$ in Figure 1 can be reached via the solid and dashed lines (rather than the dotted lines).

We rewrite the recursion equation as

$$S_x(n) = S_x(n-1) + \tau \left[F_x(n-1) S_x(n-1) + F_p(n-1) \right] \tag{9}$$

where

$$\begin{aligned} F_p(n) &= \left. \frac{\partial f}{\partial p} \right|_{x_n} \\ F_x(n) &= \left. \frac{\partial f}{\partial x} \right|_{x_n} \\ S_x(n) &= \left. \frac{dx}{dp} \right|_{x_n} \end{aligned} \tag{10}$$

S_x is a $[D \times P]$ matrix, F_x is $[D \times D]$ and F_p is $[D \times P]$. We then have

$$\begin{aligned} \frac{dJ}{dp} &= \sum_{n=1}^{N} \frac{\partial j}{\partial x_n} \frac{dx_n}{dp} \\ &= \sum_{n=1}^{N} j_x(n) S_x(n) \end{aligned} \tag{11}$$

and $j_x(n)$ is the derivative of $j(x, p)$ with respect to x, evaluated at x_n. This method requires the derivatives F_x and F_p. These will be specific to the dynamical model in question and, in this paper, are computed analytically. We provide the Euler example here as a simple illustration of the method. The numerical simulations in this paper use a more accurate integration method (see below).

Adjoint Method

Errico [17] and Giles and Pierce [22] provide introductions to the adjoint method. A derivation of the Adjoint method for dynamical models is provided rigorously in [23] and less formally in [24]. Here, we provide an informal derivation, starting with the cost function

$$J(p) = \int_0^T j\,(x,p)dt \tag{12}$$

The constraints implied by the dynamics allow us to write the Lagrangian

$$\mathcal{L}(p) = \int_0^T j(x,p)dt + \int_0^T \lambda^T\left[\dot{x} - f(x,p)\right]dt \tag{13}$$

Once the system has been integrated (solved for x) we have $\dot{x} = f(x,p)$. Hence the second term in the Lagrangian disappears and we have

$$\frac{dJ}{dp} = \frac{d\mathcal{L}}{dp} \tag{14}$$

This is the gradient we wish to compute. So far it may seem that we have made no progress but it turns out that $\frac{d\mathcal{L}}{dp}$ can be computed efficiently.

Before proceeding further, we summarise the main ideas behind the adjoint method. The central concept is that the Lagrange vector λ^T constrains the dynamical system to variations around the forward path x_n. The Lagrange vectors are of the same dimension as x and form a time series. Algebraically, the contribution of the total derivative $\frac{dx}{dp}$ to the gradient $\frac{dJ}{dp}$ is made zero, by setting λ^T appropriately. This means that the sensitivity derivative need never be calculated, resulting in a large computational saving. Instead, the gradient $\frac{dJ}{dp}$ can be expressed as a function of λ^T. We will now go through this in a bit more detail:

The proof proceeds by differentiating Eqn. 13 to give the gradient

$$\frac{dJ}{dp} = \int_0^T \left(\frac{\partial j}{\partial x}\frac{dx}{dp} + \frac{\partial j}{\partial p}\right)dt + \int_0^T \lambda^T\left(\frac{d\dot{x}}{dp} - \frac{\partial f}{\partial x}\frac{dx}{dp} - \frac{\partial f}{\partial p}\right)dt \tag{15}$$

The term involving the change in total derivative, $\frac{d\dot{x}}{dp}$, can be rewritten using integration by parts

$$\int_0^T \lambda^T\frac{d\dot{x}}{dp}dt = \left[\lambda^T\frac{dx}{dp}\right]_0^T - \int_0^T \frac{d\lambda^T}{dt}\frac{dx}{dp}dt \tag{16}$$

Substituting this into the previous expression and rearranging to group together terms involving the sensitivity derivative $\frac{dx}{dp}$ gives

$$\begin{aligned}\frac{dJ}{dp} &= \int_0^T \frac{dx}{dp}\left(\frac{\partial j}{\partial x} - \lambda^T \frac{\partial f}{\partial x} - \frac{d\lambda^T}{dt}\right) dt \\ &+ \int_0^T \left(\frac{\partial j}{\partial p} - \lambda^T \frac{\partial f}{\partial p}\right) dt + \left[\lambda^T \frac{dx}{dp}\right]_0^T\end{aligned} \tag{17}$$

As the adjoint vector λ^T has no effect on the Lagrangian (given $\dot{x} = f(x,p)$) it can be used to eliminate the first term involving the sensitivity derivative. This term is zero when:

$$\frac{d\lambda^T}{dt} = \frac{\partial j}{\partial x} - \lambda^T \frac{\partial f}{\partial x} \tag{18}$$

This is known as the adjoint equation and is used to compute λ^T. The gradient is then given by

$$\frac{dJ}{dp} = \int_0^T \left(\frac{\partial j}{\partial p} - \lambda^T \frac{\partial f}{\partial p}\right) dt + \left[\lambda^T \frac{dx}{dp}\right]_0^T \tag{19}$$

As our goal has been to avoid computation of the sensitivity derivative, $\frac{dx}{dp}$, we can eliminate the last term above by integrating the adjoint equations backward in time, starting with $\lambda_T = 0$. The starting value for the adjoint equation is arbitrary and it can be proven that if $\lambda|_{t=t_{f,a}}$ and $\lambda|_{t=t_{f,b}}$ are two different starting values for the adjoint equation with solutions λ_a and λ_b respectively, then $\left.\frac{dJ}{dp}\right|_{\lambda_a} = \left.\frac{dJ}{dp}\right|_{\lambda_b}$. Therefore, there exist infinitely many starting conditions for the adjoint equation that yields the same parametric gradient.

If the initial conditions do not depend on the parameters, as we assume for our numerical examples, then we have $\frac{dx}{dp} = 0$ at $t = 0$ and the gradient reduces to

$$\frac{dJ}{dp} = \int_0^T \left(\frac{\partial j}{\partial p} - \lambda^T \frac{\partial f}{\partial p}\right) dt \tag{20}$$

This equality can now be used to compute the parametric gradients, given the backwards solution of the adjoint equation.

There are no restrictions on the functional form to make the adjoint method viable – if one can pose the optimization problem via a Lagrangian, then the adjoint method could be used for any dynamical system (ordinary-, delay-, random- and partial- differential equation). The one and only constraint is the stability of the adjoint equation for the underlying dynamical system. Thus, static or dynamical systems that are routinely used in neuroimaging are amenable to an adjoint formulation under some loss-function including stochastic DCMs that have an analytical model for the noise (of class C^ω). Table 1 highlights the key differences between all of the methods and the crucial steps required in each of them.

Euler Example

The specification of the adjoint method starts from the specification of the Lagrangian. For us, this has the particular form

$$\mathcal{L} = -\frac{1}{2}\sum_n ||y_n - g(x_n)||^2 + \sum_n \lambda_n \left[x_n - x_{n-1} - \tau f(x_{n-1}, p)\right] \tag{21}$$

where the first term is the original cost function, the second term enforces the constraint embodied in the Euler integration of the state dynamics, and λ_n is a $[1 \times D]$ vector of Lagrange multipliers. Because $\mathcal{L}$ is a scalar, and the state x_n is a column vector, the Lagrange multipliers must be a row vector. It is in this sense that they are adjoint (or transposed) to the states. The derivative of $\mathcal{L}$ with respect to the states is then given by

$$\frac{d\mathcal{L}}{dx_n} = g_x(n)[y_n - g(x_n)] + \lambda_n - \lambda_{n+1} - \tau\lambda_{n+1}F_x(n) \tag{22}$$

where $g_x(n)$ is the derivative of $g(x,p)$ with respect to x, evaluated at x_n. Setting Eqn. 22 to zero (i.e., solving for the states) gives

$$\lambda_n = \lambda_{n+1}\left[I + \tau F_x(n)\right] - g_x(n)[y_n - g(x_n)] \tag{23}$$

This is a backward recursion, known as the adjoint equation, that starts with $\lambda_N = 0$. After solving the adjoint equations we can enter λ_n into Eqn. 20, giving

$$\frac{dJ}{dp} = \tau \sum_{n=1}^{N} \left[j_p(n) - \lambda_n F_p(n)\right] \tag{24}$$

where $j_p(n)$ is the derivative of $j(x,p)$ with respect to p, evaluated at x_n. If the observation function does not depend on model parameters then the first term disappears. A first order Euler Adjoint method has been used previously in the context of image registration [18]. However, we provide the Euler example here as an illustration of the method. The numerical simulations in this paper use a more accurate integration method (see below).

Stability

It is known that if flows are prescribed as ODEs, then their adjoint solutions are also stable [23]. Under these conditions, the numerical stability of the adjoint system is guaranteed when the adjoint equation is integrated backwards in time, in the sense that the flow is reversed. Consider a linear autonomous system, $f = Ax + B$ where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^n$ and both are invariant in time. Being linear in states with pre-defined initial conditions, such a system can be analytically integrated to yield a solution as a sum of its n matrix exponentials with unique eigenvectors and their respective eigenvalues. Such a system is asymptotically stable when the eigenvalues have a negative real part i.e., $\mathrm{Re}\left(\Lambda\left(A\right)\right) < 0$. For such a linear autonomous system the eigenvalues of the adjoint equation, $\dot{\lambda} = -\left(\frac{df}{dx}\right)^T \cdot \lambda = -A^T\lambda$ have a positive real part, proving to be asymptotically unstable. If one were now to reverse the flow i.e.,$\dot{\lambda}_* = \left(\frac{df}{dx}\right)^T \cdot \lambda_* = A^T\lambda_*$, the eigen-values then have a negative real part and the dynamics is asymptotically stable. One can derive similar results for non-linear equations using a perturbation expansion, suggesting the condition of asymptotic and uniform stability is guaranteed when the adjoint equations are integrated backwards.

Integration

For the numerical results in this paper, we used Matlab's `ode15s` function which implements a variable order method for integrating stiff differential equations [25]. Two important parameters governing the operation of this algorithm are the absolute, a, and relative, r, error tolerances. The estimated error in each integration step is constrained to be less than $\max(r|x_n|, a)$.

The absolute and relative tolerances were set to 10^{-7} for each of the gradient computation methods although results were also obtained with different sets of tolerances, taking on values a_{FD}, a_{FS}, a_{AM} and r_{FD}, r_{FS}, r_{AM} for the Finite Difference, Forward Sensitivity and Adjoint Methods respectively. When tolerances were set differently, these values were tuned for each problem (linear/nonlinear) so as to achieve good agreement among the methods.

Table 1: **Comparison of the different gradient computation methods** *The flow eqn. is either linear or non-linear, with P parameters and N state variables.*

	Finite Differences	Forward Sensitivities	Adjoint
Suitability	arbitrary	$N >> P$	$P >> N$
Cost	$(1 + P)$ flow eqns.	P non-linear sensitivity eqns. + 1 flow eqn.	1 linear adjoint eqn. + 1 flow eqn.
Key steps	1. Integrate flow eqn. 2. Parametrically perturb flow P times	1. Integrate the coupled flow and sensitivity eqns.	1. Integrate flow eqn. 2. Integrate adjoint eqn.

Results

Custom MATLAB scripts were written to implement each of the gradient computation methods.

Linear Models

First we consider the linear models

$$\dot{x} = Ax \tag{25}$$

where x is a D-dimensional state vector with initial value $x_0 = 1$, and A is a $D \times D$ connectivity matrix. Readers familiar with DCM for fMRI will recognise A as the endogenous or average connectivity matrix. A model with D states therefore has $P = D^2$ parameters (Figure 2(A)). The system is integrated from time 0 to T. To ensure stability, we constructed A using the linear expansion

$$A = \sum_{d=1}^{D} q_d v_d v_d^T \tag{26}$$

where $v_d \sim \mathcal{N}(v_d; 0, 1)$ are standard D-dimensional Gaussian random vectors, which are serially orthogonalized. The scalars q_d are negative real numbers so that the corresponding eigenstates are exponentially decaying modes. The values of q_d were set so that the corresponding time constants were between $T/5$ and T. Figure 2(B) show the time series for five such eigenstates.

For each model dimension considered (see below) we generated a state trajectory using known model parameters generated as described above. We then created an observable data time series $y_n = g(x_n)$ with the observation function $g(x_n) = x_n$, that is, all of the dynamical states are observed.

We then created 'perturbed' parameters by adding Gaussian noise with a standard deviation of 10% of the original parameters. The cost function was defined as

$$J \quad = \quad -\frac{1}{2}\sum_n [y_n - g(x_n)]^2 \tag{27}$$

To summarize, the 'data points' y_n were created using the original parameters and the 'model predictions', $g(x_n)$ used the perturbed parameters. Gradients were then estimated at this perturbed point.

The systems were integrated using the tolerances of FD and FS fixed at 10^{-7}. Although, the tolerance of AM was adjusted so as to achieve best fit to the FD based gradient estimate, for the efficiency-scaling simulations we fixed it at a lower value of 10^{-3}. This is illustrated in Figure 3(A) that shows the estimated gradients for a $D = 5$ dimensional linear system. Setting the tolerance of the AM method to 10^{-3} did not affect the mean-squared deviation of the gradients obtained between the FD and the AM methods (data not shown).

We then compared the three gradient computation methods. Figure 3(B) plots the computation time as a function of state dimension. For a 28 node system with 784 model parameters the computation time for the adjoint method is 77 times less than for the finite difference method.

Nonlinear Models

Next, we consider weakly coupled oscillators of the form

$$\dot{x}_i(t) = f_i + \sum_{j=1, j\neq i}^{D} (\alpha_{ij} \sin[x_i(t) - x_j(t)] + \beta_{ij}\cos[x_i(t) - x_j(t)]) \tag{28}$$

where the model parameters comprise the parameters f, α and β. A model with D states therefore has $P = 2D^2 - D$ parameters. We used a cost function equal to the mean square deviation between observed and predicted state trajectories i.e., the norm of the prediction error (again, all states were observed).

The tolerance parameters of the integration process were set identical to those used for the linear models. Again, the adjoint equation being a linear first order ODE enables the use of lower tolerances (10^{-3}). This process was implemented for a $D = 5$ dimensional problem and Figure 4(B) shows the estimated gradients.

We then compared the three gradient computation methods. Figure 4(C) plots the computation time as a function of state dimension. For a 24 node system with 1128 model parameters the computation time for the adjoint method is 50 times less than for the finite difference method.

The efficiency of the AM formulation is due to two reasons – first, the adjoint equation is linear and second it is integrated only once to compute the gradient. Given that the AM equation is linear, the condition number is low, enabling any ODE integrator to integrate the adjoint equation with ease. Indeed, if the ODE integrator is subjected to unnecessary high tolerances it spends more time integrating the adjoint equation. Thus, the advantage of the adjoint scheme reveals both the parsimonious integration scheme as well as the linearity of this equation that requires less-conservative tolerances.

Discussion

Optimization theory attaches mathematical well-posedness to the issue of biological data-assimilation by formalizing the relationship between empirically measured data and a model generating those responses. In this paper, we introduced three different methods for numerical gradient estimation that forms an integral part of any convex optimization framework. Our comparison establishes that the adjoint method is computationally more efficient for numerical estimation of parametric gradients for state-space models – both linear and non-linear, as in the case of a dynamical causal model (DCM). As is apparent from the gradient equations, the adjoint method is efficient when the numbers of parameters are much greater than the number of states determining the cost function. The contrary is true for the forward sensitivity approach albeit for large state-space models, finite-difference based gradients prove to be beneficial. There are two remarks that can be made about the adjoint formulation. First, regardless of whether the flow is linear or non-linear the adjoint method requires the integration of a single linear equation – the computational efficiency is inherent in the structure of the equation. Second, the appearance of a transpose on the adjoint vector implies that the flow of information in the system of equations is reversed, it is in this sense that the adjoint equations are integrated backwards in time.

Although, adaptive error correction is invariably used in the integration of differential equations, the numerical simulations suggest that the tolerance used for integrating the flow and adjoint differential equations are vital in determining the accuracy of the parametric gradients, due to the presence of discretization error. In theory, plugging in the solution field to the flow equation should yield zero, but due to the existence of discretization error the residual is generally non-zero. The same is true for the adjoint equation. In fact, a theorem by Becker and Rannacher [26] shows how discretization of the gradient depends on the average of the errors and the residuals accumulated in the integration of flow and the adjoint equations [27]. This is also the case for obtaining gradients via finite-differencing, where we find that the fidelity of error-free discretisation of the flow equations are a prerequisite for guaranteeing parametric gradients that are a reliable estimate of the true gradient.

It is known that if the flows are prescribed as ODEs the numerical stability of the adjoint system is guaranteed when the adjoint equation is integrated backwards in time, in the sense that the flow is reversed. Our derivation of the adjoint method is mathematically informal so as to illustrate the basic working principle; rigorous mathematical proofs that accommodate higher order differential algebraic equations, time-dependent parameters or objective functionals that depend on initial conditions are available elsewhere [23].

For DCM inversions that allow problem specification in a pre-defined form it may be generally time-efficient to derive the gradient functions analytically rather than using automatic differentiation [28]. Automatic differentiation is particularly important for partial differential equations (PDEs) that have 3-dimensional representations, requiring automatization and therefore proving to be error resilient [29]. For a PDE-constrained optimization problem the solution is governed by a fully coupled Karush-Kuhn-Tucker (KKT) system of equations. These can be computationally expensive for parabolic and hyperbolic PDEs, as well as displaying slow convergence of the defined objective functional (ill-conditioning). The adjoint formulation remedies this by decoupling the coupled PDEs and replacing them by iterative solves of a linear adjoint PDE equation. Additional success of adjoint-based gradient methods for PDE-constrained optimization relies on the fact that mesh independent convergence can be attained. Further speedup could also be obtained by using compiled implementation of forward and adjoint sensitivity methods available in the SUNDIALS time integration package [24]. This code is written in C and may offer substantial speed advantages over MATLAB implementations.

For data assimilation, it is only rarely that we have precise information on the states or the parameters [30]. Is the adjoint method equally efficient when there is noise on the states and the parameters? One way to represent uncertainty in a mathematical model, whether static or dynamic is to formulate it as a polynomial chaos expansion [31], one for each noisy state or parameter. This then enables the characteristic statistical quantities to be evaluated as some function of the expansion coefficients – the uncertainty now becomes parameterised. Estimation of the numerical gradient can then proceed akin to a deterministic dynamical model where the computational burden does not depend on the number of parameters [32]. Alternatively, adjoint methods can be gracefully combined with Markov Chain Monte Carlo (MCMC) sampling-based evaluation of the posterior densities [33]. In a forthcoming paper we address how second-order adjoined gradient estimates could be obtained in the context of Bayesian inversion of neural masses, mean fields, and neural field equations.

Constrained optimization problems arise in many scientific fields, from neuroscience to financial mathematics, therefore a fundamental need for efficient computational methodologies arises. Our work promotes such an endeavor especially for data-sets arising in neuroscience, for example the inversion of large-scale DCMs that have been routinely used to test hypotheses about different functional brain architectures.

Acknowledgements

This work is supported by a Wellcome Trust/DBT Early Career fellowship to BS. KJF and WDP are supported by the Wellcome Trust. Initial code-base used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] K.J. Friston, L. Harrison, and W. Penny. Dynamic Causal Modelling. *NeuroImage*, 19(4):1273–1302, 2003.

[2] O David, S. Kiebel, L. Harrison, J Mattout, J. Kilner, and K. Friston. Dynamic causal modeling of evoked responses in EEG and MEG. *Neuroimage*, 30(4):1255–1272, May 2006.

[3] Gustavo Deco, Viktor K. Jirsa, Peter A. Robinson, Michael Breakspear, and Karl Friston. The dynamic brain: from spiking neurons to neural masses and cortical fields. *PLoS Comput Biol*, 4(8):e1000092, 2008.

[4] H. Wilson. *Spikes, Decisions and Actions: The Dynamical Foundations of Neuroscience.* Oxford University Press, New York, 1999.

[5] J. Hopfield and C. Brody. What is a moment ? Transient synchrony as a collective mechanism for spatiotemporal integration. *Proceedings of the National Academy of Sciences*, 98(3):1282–1287, 2001.

[6] M. Gazzaniga. Neuroscience and the correct level of explanation for understaning mind. *Trends in Cognitive Sciences*, 14:291–292, 2010.

[7] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, New York, 1992.

[8] C.M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2006.

[9] K Friston, J Mattout, N Trujillo-Barreto, J Ashburner, and W Penny. Variational free energy and the Laplace approximation. *Neuroimage*, 34 (1):220–234, Jan 2007.

[10] J. Ramsay, H. Hooker, D. Campbell, and J. Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society Sereis B*, 69(5):741–796, 2007.

[11] V. Vyshemirsky and M. Girolami. Bayesian ranking of biochemical system models. *Bioinformatics*, 24(6):833–9, 2008.

[12] B Calderhead and M Girolami. Estimating Bayes factors via thermodynamic integration and population MCMC. *Computational Statistics & Data Analysis*, 53(12):4028–4045, 2009.

[13] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, 2nd edition, 2006.

[14] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[15] T Deneux and O Faugeras. Using nonlinear models in fMRI data analysis: model selection and activation detection. *Neuroimage*, 32:1669–1689, 2006.

[16] Q. Wang. Forward and adjoint sensitivity computation of chaotic dynamical systems. *Journal of Computational Physics*, 235:1–13, 2013.

[17] R Errico. What is an Adjoint Model ? *Bulletin of the American Metereological Society*, 78:2577–2591, 1997.

[18] A Clark. *Geodesic shooting for anatomical curve registration on the plane.* PhD thesis, Department of Aeronautics, Imperial College, London, 2011.

[19] M. Stemmler, B. Sengupta, S. B. Laughlin, and J. E. Niven. Energetically optimal action potentials. In *Advances in Neural Information Processing Systems*, pages 1566–1574, 2012.

[20] D Richtmeyer and K Morton. *Difference methods for initial value problems.* Wiley, New York, 1967.

[21] T Gronwall. Note on the Derivatives with Respect to a Parameter of the Solutions of a System of Differential Equations. *Annals of Mathematics*, 20:292–296, 1919.

[22] M Giles and N Pierce. An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion*, 65:393–415, 2000.

[23] Y Cao, S Li, L Petzold, and R Serban. Adjoint sensitivity analysis for differential-algebraic equations: the adjoint DAE system and its numerical solution. *SIAM journal on Scientific Computing*, 24:1076–1089, 2003.

[24] A Hindmarsh and R Serban. User Documentation for CVODES, and ODE Solver with Sensitivity Analysis Capabilities. Technical report, Centre for Applied Scientific Computing, Lawrence Livermore National Laboratory, 2002.

[25] L Shampine and M Reichelt. The MATLAB ODE Suite. *SIAM Journal on Scientific Computing*, 18:1–22, 1997.

[26] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1102, 2001.

[27] W. Bangerth and R. Rannacher. *ETH Zürich Lectures in Mathematics*, chapter Adaptive finite element methods for differential equations. Birkhäuser, 2003.

[28] C Bischof, H. Martin Bücker, A Vehreschild, and J Willkomm. Automatic differentiation for matlab (ADiMat). In *MATLAB-Day*, Aachen, Germany, October 2012.

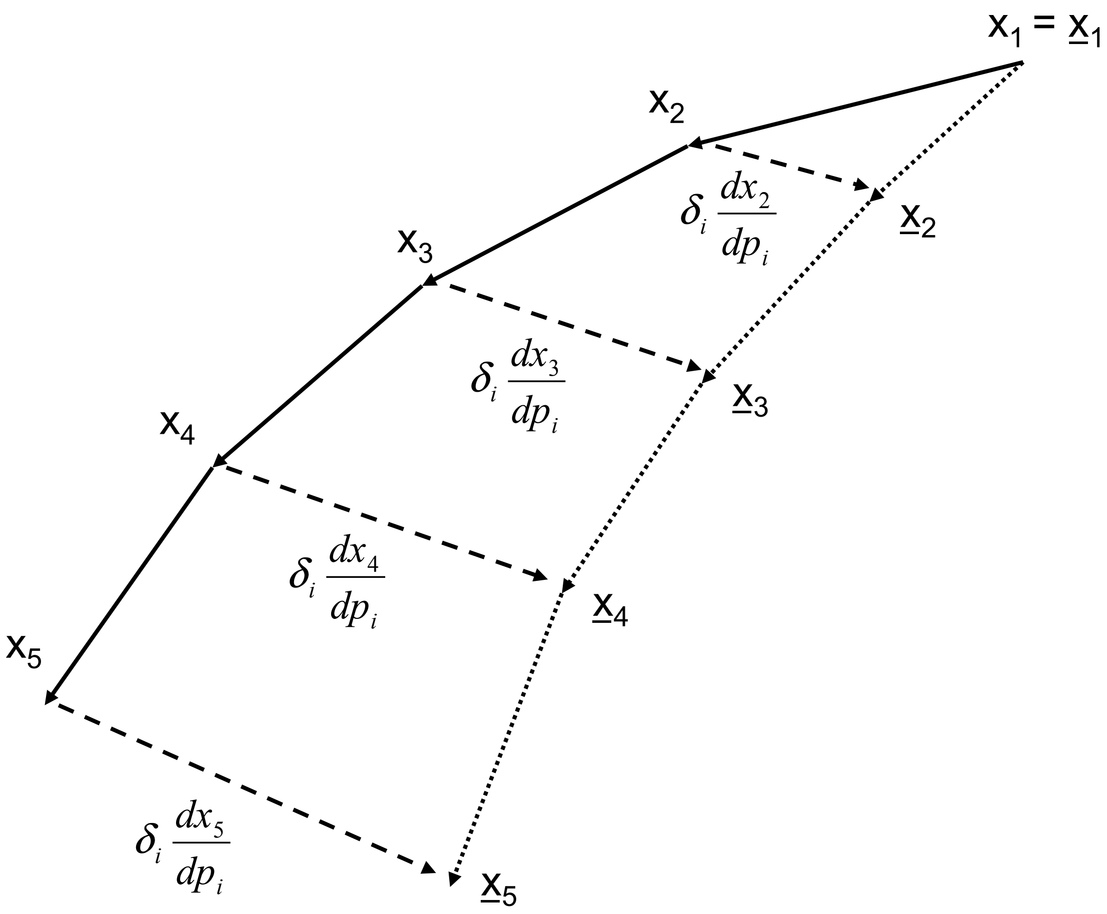
[29] Marzio Sala, Michael A. Heroux, and David M. Day. Trilinos Tutorial. Technical Report SAND2004-2189, Sandia National Laboratories, 2004.

[30] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series.* The MIT Press, 1964.

[31] Norbert Wiener. The Homogeneous Chaos. *American Journal of Mathematics*, 60(4):897–936, 1938. ISSN 00029327.

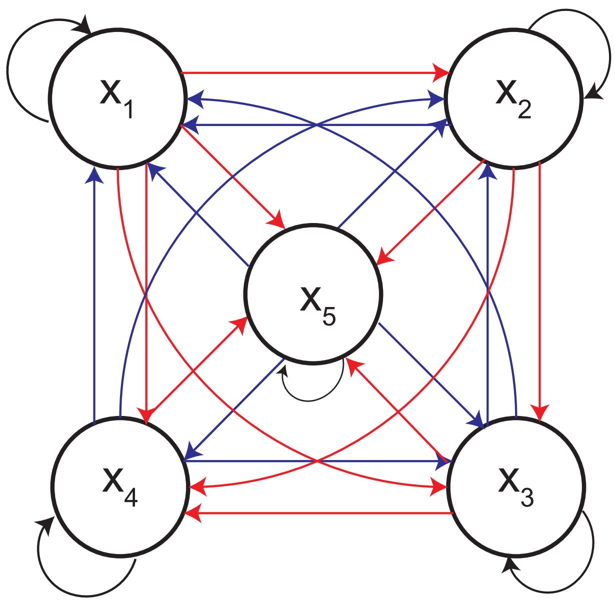
[32] A. K. Alekseev, I. M. Navon, and M. E. Zelentsov. The estimation of functional uncertainty using polynomial chaos and adjoint equations. *Int. J. Numer. Meth. Fluids*, 67:328341, 2010.

[33] D. Ghate and M.B. Giles. Inexpensive monte carlo uncertainty analysis. In *Symposium on Applied Aerodynamics and Design of Aerospace Vehicles*, 2005.

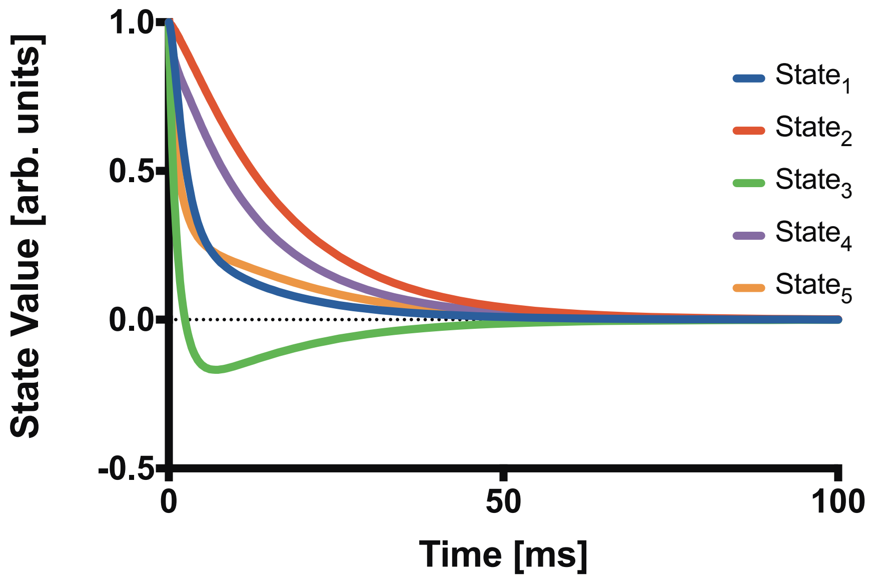


$x_1 = \underline{x}_1$
x_2
$\underline{x}_2$
$\delta_i \frac{dx_2}{dp_i}$
x_3
$\underline{x}_3$
$\delta_i \frac{dx_3}{dp_i}$
x_4
$\underline{x}_4$
$\delta_i \frac{dx_4}{dp_i}$
x_5
$\underline{x}_5$
$\delta_i \frac{dx_5}{dp_i}$

Figure 1: **Forward Sensitivity** *The solid path indicates a trajectory of points* x_n*, with* $n = 1...5$*, for a dynamical system with parameters* p*. The dotted path indicates the trajectory* $\underline{x}_n$ *for the same dynamical system but with parameters* $\underline{p} = p + \delta_i$*. The dotted path can be reached from the solid path via the total derivative* $\frac{dx_n}{dp}$*. The Forward Sensitivity approach provides a method for computing this derivative.*

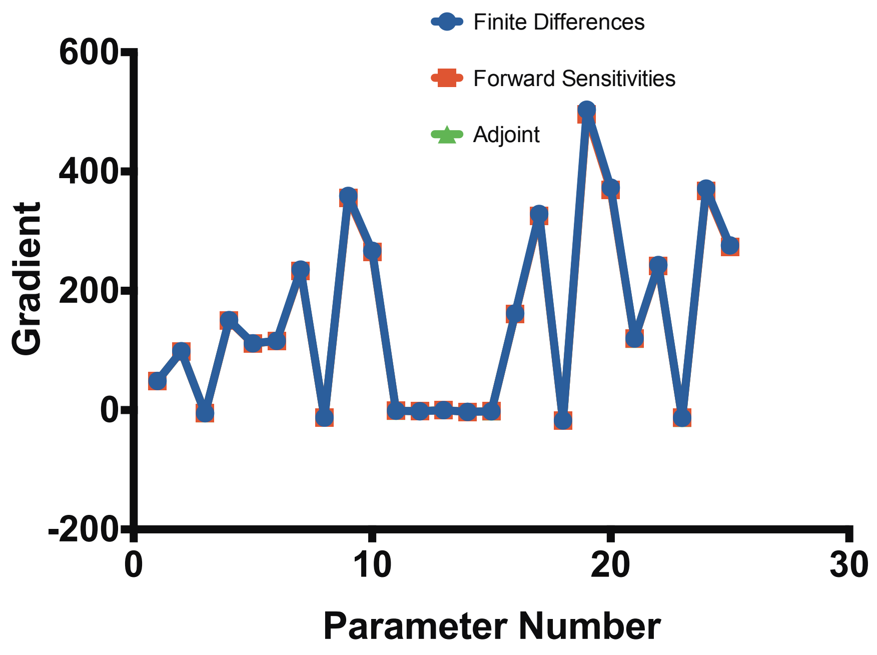


(A) **Connectivity of the linear state-space model**

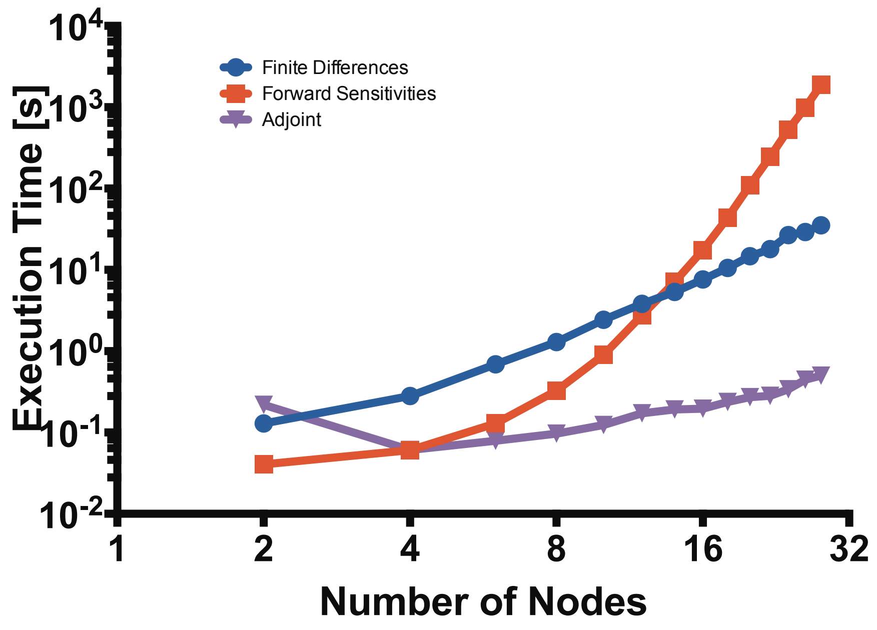


(B) **Time-evolution of the linear states**

Figure 2: **Linear System** *(A) The 5-dimensional state-space model and (B) the linear evolution of its eigenstates.*

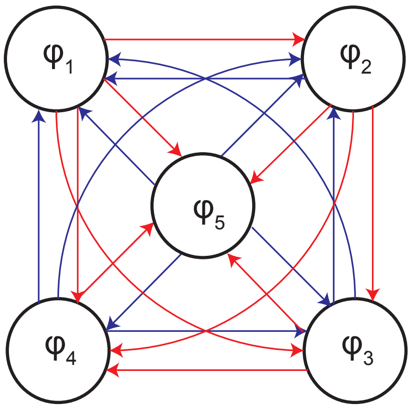


(A) **Gradients for 5-dimensional state-space (25 parameters)**

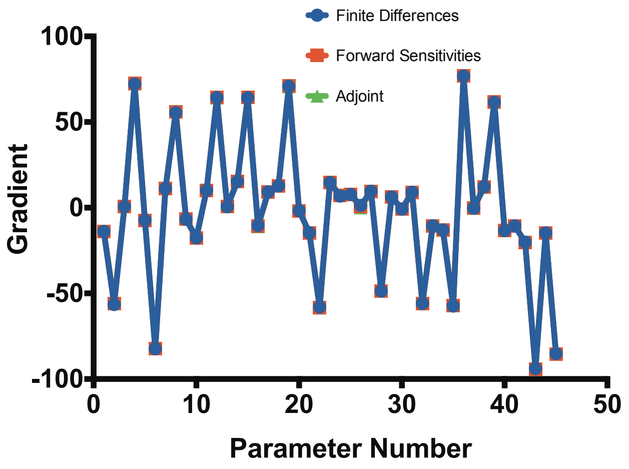


(B) **Computation time as a function of state dimension**

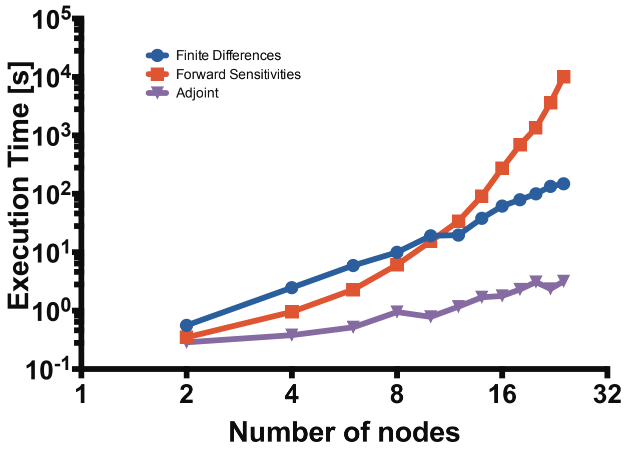
Figure 3: **Computational efficiency for linear systems** *(A) Comparison of the parametric gradient obtained by the three methods. (B) Scaling of run-time as a function of the number of nodes. The absolute and relative tolerances of FD and FS methods were set to* 10^{-7} *while the tolerances for the AM method was fixed to* 10^{-3}*. Simulation time was fixed at 400 ms.*



(A) **Connectivity of the non-linear state-space model**



(B) **Gradients for 5-dimensional state-space (45 parameters)**



(C) **Computation time as a function of state dimension**

Figure 4: **Computational efficiency for non-linear systems** *(A) Comparison of the gradient obtained by the three methods. Here, the last five parameters quantify the intrinsic oscillator frequencies, and the first 40 parameters the sine and cosine interaction terms. (B) Scaling of run-time as a function of the number of nodes. The absolute and relative tolerances of FD and FS methods were set to* 10^{-7} *while the tolerances for the AM method was fixed to* 10^{-3}*. Simulation time was fixed at 100 ms.*