Nonlinear Least Squares or Trajectory Matching

Zhe Gao

School of Mathematics
Sun Yat-sen University

Sep, 21 2022

Basic model

- The standard ordinary differential equation initial value problem is

$$\mathrm{D}\mathbf{x}(t) = \mathbf{f}(t; \mathbf{x}(t), \boldsymbol{\theta}) \text{ and } \mathbf{x}(t_0) = \mathbf{x}_0$$

  where $\mathbf{x}(t)$ is a vector containing the state or trajectory values $x_i(t)$, $i = 1, \ldots, d$.
- The initial state vector can itself be an additional set of parameters to be estimated.
- We use $\boldsymbol{\vartheta}$ to indicate the subset of $(\boldsymbol{\theta}, \mathbf{x}_0)$ that must be estimated throughout this chapter and $\mathbf{x}(t; \boldsymbol{\vartheta}) = \mathbf{x}(t; \boldsymbol{\theta}, \mathbf{x}_0)$.

Nonlinear least square

The sum of squared errors criterion is defiend as:

$$\mathrm{SSE}(\boldsymbol{\vartheta}) = \sum_{j=1}^{n} \left(y_j - x(t_j, \boldsymbol{\vartheta})\right)^2 .$$

- The minimum value of $\mathrm{SSE}(\boldsymbol{\vartheta})$ occurs when the gradient is zero.
- In a nonlinear system, the derivatives are functions of both the independent variable and the parameters, so in general these gradient equations do not have a closed solution.
- The Gauss-Newton method is the most commonly used technique for optimization in non-linear least squares problems.

Table of Contents

Gauss-Newton Methods

- Taylor's theorem says that we can approximate $\mathrm{SSE}(\boldsymbol{\vartheta})$ by a quadratic function of ϑ given in terms of its derivatives at $\boldsymbol{\vartheta}^0$:

$$\begin{aligned}\mathrm{SSE}(\boldsymbol{\vartheta}) \approx & \mathrm{SSE}\left(\boldsymbol{\vartheta}^0\right)+\partial_{\vartheta} \mathrm{SSE}\left(\boldsymbol{\vartheta}^0\right)\left(\boldsymbol{\vartheta}-\boldsymbol{\vartheta}^0\right) \\ & +\frac{1}{2}\left(\boldsymbol{\vartheta}-\boldsymbol{\vartheta}^0\right)^T \partial_{\vartheta}^2 \mathrm{SSE}\left(\boldsymbol{\vartheta}^0\right)\left(\boldsymbol{\vartheta}-\boldsymbol{\vartheta}^0\right)\end{aligned}$$

  where $\partial_{\boldsymbol{\vartheta}} \mathrm{SSE}\left(\boldsymbol{\vartheta}^0\right)$ is the gradient vector at the initial guess and $\partial_{\boldsymbol{\vartheta}}^2 \mathrm{SSE}\left(\boldsymbol{\vartheta}^0\right)$ is the corresponding Hessian matrix.
- The new guess is obtained by

$$\boldsymbol{\vartheta}^1=\boldsymbol{\vartheta}^0-\left[\partial_{\boldsymbol{\vartheta}}^2 \mathrm{SSE}\left(\boldsymbol{\vartheta}^0\right)\right]^{-1} \partial_{\boldsymbol{\vartheta}} \mathrm{SSE}\left(\boldsymbol{\vartheta}^0\right).$$

Gauss-Newton Methods

- For the least squares criterion the derivatives are:

$$\begin{aligned}\partial_{\boldsymbol{\vartheta}} \operatorname{SSE}\left(\boldsymbol{\vartheta}^0\right) &= -2\sum_{j=1}^{n} \partial_{\boldsymbol{\vartheta}} x(t_j, \boldsymbol{\vartheta})\left(y_j - x(t_j, \boldsymbol{\vartheta})\right) \\ &= -2\partial_{\boldsymbol{\vartheta}} \mathbf{x}(\boldsymbol{\vartheta})^T(\mathbf{y} - \mathbf{x}(\boldsymbol{\vartheta})),\end{aligned}$$

  where $\mathbf{y}$ is the vector of length n containing observations (y_j) and $\mathbf{x}(\boldsymbol{\vartheta})$ is the vector of $(x(t_j, \boldsymbol{\vartheta}))$ values.

- The Hessian matrix is

$$\partial^2_{\boldsymbol{\vartheta}} \operatorname{SSE}\left(\boldsymbol{\vartheta}^0\right) = -2\mathbf{J}(\boldsymbol{\vartheta})^T\mathbf{J}(\boldsymbol{\vartheta}) - 2\partial^2_{\boldsymbol{\vartheta}} x(\boldsymbol{\vartheta})^T(\mathbf{y} - \mathbf{x}(\boldsymbol{\vartheta})),$$

  where $\mathbf{J}(\boldsymbol{\vartheta})$ is the Jacobian matrix

$$\mathbf{J}(\boldsymbol{\vartheta}) = [\mathbf{J}(\boldsymbol{\vartheta})]_{jl} = \frac{dx(t_j, \boldsymbol{\vartheta})}{d\boldsymbol{\vartheta}_l}.$$

Gauss-Newton Methods

To summarise:

- Initialize $\boldsymbol{\vartheta} = \boldsymbol{\vartheta}^0$.
- Do until convergence:

$$\begin{aligned}
\mathbf{H}(\boldsymbol{\vartheta}^k) &= \mathbf{J}(\boldsymbol{\vartheta}^k)^T \mathbf{J}(\boldsymbol{\vartheta}^k) \\
\mathbf{g}(\boldsymbol{\vartheta}^k) &= \mathbf{J}(\boldsymbol{\vartheta}^k)^T \left(\mathbf{y} - x(\boldsymbol{\vartheta}^k) \right) \\
\boldsymbol{\vartheta}^{k+1} &= \boldsymbol{\vartheta}^k - \mathbf{H}(\boldsymbol{\vartheta}^k)^{-1} \mathbf{g}(\boldsymbol{\vartheta}^k).
\end{aligned}$$

Remark

- We may check that each step actually reduces SSE($\boldsymbol{\vartheta}$), and include a search for a minimum along a line in the step direction $-\mathbf{H}(\boldsymbol{\vartheta}^k)^{-1}\mathbf{g}(\boldsymbol{\vartheta}^k)$.
- The Levenberg-Marquardt algorithm replaces $\mathbf{H}\left(\boldsymbol{\vartheta}^k\right)$ with $\mathbf{H}\left(\boldsymbol{\vartheta}^k\right) + \gamma_k \mathbf{I}$ where γ_k is chosen each iteration to ensure both that each step improves the objective function as well as making sure that the matrix can be inverted.

Stop criteria

- There has been little movement in parameter estimates:

$$\left\| \boldsymbol{\vartheta}^{k+1} - \boldsymbol{\vartheta}^{k} \right\| < \delta_1,$$

- The objective function has not improved much:

$$\mathrm{SSE}(\boldsymbol{\vartheta}^{k}) - \mathrm{SSE}(\boldsymbol{\vartheta}^{k+1}) < \delta_2.$$

- The gradient of the objective function is small:

$$\| \partial_{\boldsymbol{\vartheta} x} \mathrm{SSE}(\boldsymbol{\vartheta}) \| < \delta_3.$$

How to compute the Jacobian matrix

We may use a finite difference to approximate:

$$\frac{dx(t;\boldsymbol{\vartheta})}{d\boldsymbol{\vartheta}_l} \approx \frac{x(t;\boldsymbol{\vartheta}+\delta\mathbf{e}_l) - x(t;\boldsymbol{\vartheta})}{\delta}.$$

where $\mathbf{e}_l$ is a vector of zeros with a one in the l-th position and δ is a small positive number.

However, these approximations can be unstable, especially if the ODE solver you are using includes adaptive strategies to choose step sizes.

Sensitivity Equations

Instead, we define the sensitivity equations by considering the time-derivative of $\mathbf{J}(t;\vartheta)$.

- Applying the chain rule, we have

$$\begin{aligned}&\mathrm{D}\left[\partial_\theta x(t;\boldsymbol{\theta},x_0)\right]\\ =&\partial_\theta f(t;x(t;\boldsymbol{\theta},x_0),\boldsymbol{\theta})+\partial_x f(t;x(t;\boldsymbol{\theta},x_0),\boldsymbol{\theta})\,\partial_\theta x(t;\boldsymbol{\theta},x_0).\end{aligned}$$

and similarly

$$\mathrm{D}\left[\partial_{x_0} x(t;\boldsymbol{\theta},x_0)\right]=\partial_x f(t;x(t;\boldsymbol{\theta},x_0),\boldsymbol{\theta})\,\partial_{x_0} x(t;\boldsymbol{\theta},x_0).$$

Since

$$\partial_\theta x(t_0;\boldsymbol{\theta},x_0)=\mathbf{0},\quad \partial_{x_0} x(t_0;\boldsymbol{\theta},x_0)=\mathbf{I}.$$

Sensitivity Equations

- We have the expanded set of differential equations

$$\mathrm{D}\begin{pmatrix} x \\ \partial_\theta x \\ \partial_{x_0} x \end{pmatrix} = \begin{pmatrix} f(t; x, \boldsymbol{\theta}) \\ \partial_\theta f(t; x, \boldsymbol{\theta}) + \partial_x f(t; x, \boldsymbol{\theta}) \partial_\theta x \\ \partial_x f(t; x, \boldsymbol{\theta}) \partial_{x_0} x \end{pmatrix}$$

  with corresponding initial conditions

$$\begin{pmatrix} x(t_0) \\ \partial_\theta x(t_0) \\ \partial_{x_0} x(t_0) \end{pmatrix} = \begin{pmatrix} x_0 \\ \mathbf{0} \\ \mathbf{I} \end{pmatrix}.$$

Example: Filling a Container

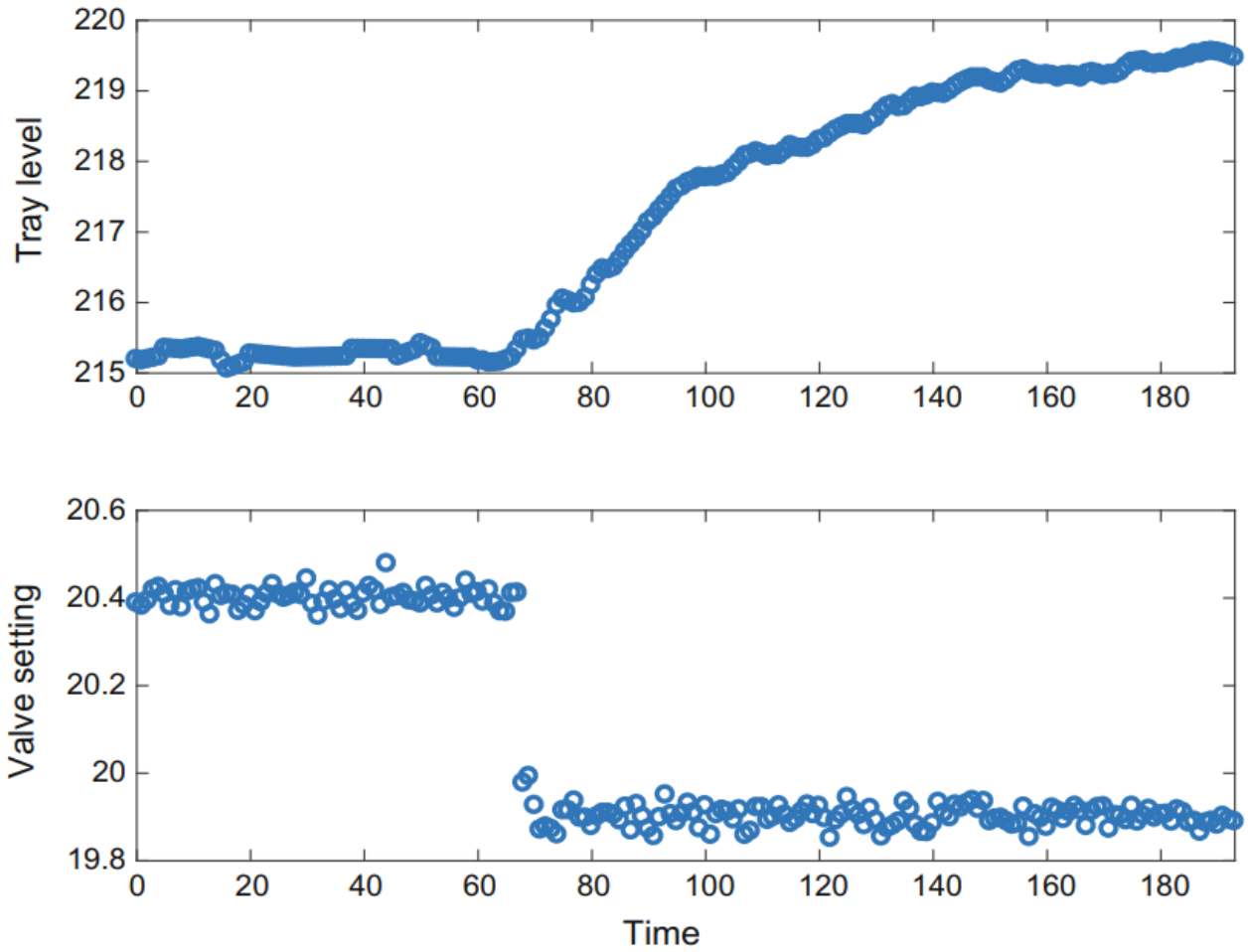


Fig. 1.2 The *upper panel* shows the fluid level in a tray that receives input from another container when a valve is opened. The *lower level* shows the setting of a valve before and after being opened

Example: Filling a Container

To describe this behavior, we write that

$$\mathrm{D}x = \beta_0 + \beta_1 x + \beta_2 u$$

in which u is represented by the step change in the flow into the tank. We expect β_1 to be negative, so the fixed point for this equation is $(\beta_0 + \beta_2 u)/\beta_1$.

In this system, the solutions are given by

$$x(t) = x_0 e^{\beta_1 t} - \frac{\beta_0}{\beta_1}\left(1 - e^{\beta_1}\right) - u * \frac{\beta_2}{\beta_1} e^{\beta_1 t}\left(e^{-\beta_1 (t)\,\min(t,t_0)} - 1\right).$$

Example: Filling a Container

The sensitivity equations is defined as

$$\mathrm{D}\begin{pmatrix} x \\ \partial_{\beta_0} x \\ \partial_{\beta_1} x \\ \partial_{\beta_2} x \\ \partial_{x_0} x \end{pmatrix} = \begin{pmatrix} \beta_0 + \beta_1 x + \beta_2 u \\ 1 + \beta_1 \partial_{\beta_0} x \\ x + \beta_1 \partial_{\beta_1} x \\ u + \beta_1 \partial_{\beta_3} x \\ \beta_1 \partial_{x_0} x \end{pmatrix}$$

starting from initial conditions

$$\mathrm{D}\begin{pmatrix} x \\ \partial_{\beta_0} x \\ \partial_{\beta_1} x \\ \partial_{\beta_2} x \\ \partial_{x_0} x \end{pmatrix} = \begin{pmatrix} x_0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Limits of sensitivity equations

- They come at the cost of increasing the dimension of the system that must be estimated.
- They don't account for the numerical approximation of ODEs.

Automatic differentiation

- The idea of Automatic differentiation is that, rather than using an analytical expression for the derivative, the function is evaluated in terms of the numerical operations performed by the computer.
- AD requires additional functionality that tracks numerical operation in code. Packages to perform this exist in R and Matlab as well as many other programming environments.

Table of Contents

Inference

If we imagine conducting the experiment and obtaining new data many times, how far, on average, would our parameters be from the truth?

In a hypothetical repeat data set we would have observations y_j^* that satisfy

$$y_j^* = x(t_j, \boldsymbol{\theta}, x_0) + \varepsilon_j^*$$

where the ε_j^* are different random values, but nothing else has changed. We may think of these estimates as functions of the data $\hat{\vartheta} = \vartheta(y_1, \ldots, y_n)$.

Asymptotic distribution

Under appropriate conditions, if the true set of parameters are $\boldsymbol{\vartheta}$ and we estimate $\hat{\boldsymbol{\vartheta}}$ via nonlinear least squares, the parameter estimates are asymptotically normal, that is

$$\hat{\boldsymbol{\vartheta}} \sim N\left(\boldsymbol{\vartheta}, \sigma^2 \left[\mathbf{J}(\boldsymbol{\vartheta})^T \mathbf{J}(\hat{\boldsymbol{\vartheta}})\right]^{-1}\right)$$

where σ^2 is the population variance of the errors ε.

This is not always the case for dynamic systems even if the asymptotic assumptions are correct.

Estimate of parameters

- An estimate of σ^2 is obtained from the empirical residuals at the estimated parameters:

$$\hat{\sigma}^2 = \frac{1}{n-p} \sum_{j=1}^{n} \left(y_j - x(t_j, \hat{\boldsymbol{\vartheta}}) \right)^2 .$$

- The covariance matrix can be estimated by

$$\hat{\Sigma}_{\hat{\boldsymbol{\vartheta}}} = \hat{\sigma}^2 \left[\mathbf{J}(\hat{\boldsymbol{\vartheta}})^T \mathbf{J}(\hat{\boldsymbol{\vartheta}}) \right]^{-1} .$$

Confidence interval

For a particular parameter $\boldsymbol{\vartheta}_k$, the 95 % confidence interval is

$$\left[\hat{\boldsymbol{\vartheta}}_k - 1.96\sigma_{\hat{\boldsymbol{\vartheta}}_k}, \hat{\boldsymbol{\vartheta}}_k + 1.96\sigma_{\hat{\boldsymbol{\vartheta}}_k}\right].$$

where $\sigma_{\hat{\boldsymbol{\vartheta}}_k}$ is the square root of the (k, k) th entry in $\Sigma_{\hat{\boldsymbol{\vartheta}}}$.

For the whole vector $\boldsymbol{\vartheta}$, we can replace 1.96 with any quantile z^{α} of the standard normal to control the probability of a confidence interval missing the true value at α.

Prediction intervals

We can also translate these into prediction intervals for $x(t)$ via a Taylor expansion of $x(t;\boldsymbol{\vartheta})$

$$x(t;\hat{\boldsymbol{\vartheta}}) = x\left(t;\boldsymbol{\vartheta}^0\right) + \mathbf{J}(t;\hat{\boldsymbol{\vartheta}})\left(\hat{\boldsymbol{\vartheta}} - \boldsymbol{\vartheta}^0\right)$$

where $\mathbf{J}(t;\hat{\boldsymbol{\vartheta}})$ is the single-row Jacobian evaluated at only t. Then

$$\text{var}(x(t;\hat{\boldsymbol{\vartheta}})) \approx \mathbf{J}(t;\hat{\boldsymbol{\vartheta}})^T \Sigma_{\hat{\boldsymbol{\vartheta}}} \mathbf{J}(t;\hat{\boldsymbol{\vartheta}}) = \Sigma_x.$$

For a new point in any one dimension $y_j = x\left(t_j,\boldsymbol{\vartheta}\right) + \sigma\varepsilon$, we can employ the interval

$$\left[x(t;\hat{\boldsymbol{\vartheta}}) - 1.96\left(\sigma_x + \sigma\right), x(t;\hat{\boldsymbol{\vartheta}}) + 1.96\left(\sigma_x + \sigma\right)\right].$$

Note that these estimates only require solutions to the sensitivity equations at the estimates $\hat{\boldsymbol{\vartheta}}$.

Table of Contents

Multiple Variables

We now extend the discussion to $d > 1$ variables:

$$y_{ji} = x_i(t_j, \boldsymbol{\vartheta}) + \varepsilon_{ji}, i \in \mathscr{M}, j = 1, \ldots, n_i,$$

where n_i denotes the number of observations for variable i, ε_{ji} has variance σ_i^2 but the errors are otherwise independent.

Multivariate Gauss-Newton Method

- Account for different scales and different measurement precision, we weight each state variable differently.

$$\mathrm{SSE}(\boldsymbol{\vartheta}) = \sum_{i \in \mathscr{M}} \sum_{j=1}^{n_i} \omega_i \left(y_{ji} - x_i\left(t_j, \boldsymbol{\vartheta}\right)\right)^2 .$$

- Choosing $\omega_i = 1/\sigma_i^2$ is a particularly sensible choice, but we will have to estimate the measurement error variance σ_i^2 from the data before this option can be employed.
- Using a priori weights ω_i is also reasonable.

Multivariate Gauss-Newton Method

The derivatives can be computed as:

$$\begin{aligned}\partial_{\boldsymbol{\vartheta}} \operatorname{SSE}\left(\boldsymbol{\vartheta}^0\right) &= -2 \sum_{i=1}^{d} \frac{1}{\sigma_i^2} \partial_{\boldsymbol{\vartheta}} \mathbf{x}_i(\boldsymbol{\vartheta})^T \left(\mathbf{y}_i - \mathbf{x}_i(\boldsymbol{\vartheta})\right) \\ &= -2 \sum_{i=1}^{d} \frac{1}{\sigma_i^2} \mathbf{J}_i(\boldsymbol{\vartheta})^T \left(\mathbf{y}_i - \mathbf{x}_i(\boldsymbol{\vartheta})\right)\end{aligned}$$

where $\mathbf{J}_i(\boldsymbol{\vartheta})$ is the $n_i \times p$ Jacobian matrix for variable i. The Hessian matrix is:

$$\partial_{\boldsymbol{\vartheta}}^2 \operatorname{SSE}\left(\boldsymbol{\vartheta}^0\right) = -2 \sum_{i=1}^{d} \omega_i \mathbf{J}_i(\boldsymbol{\vartheta})^T \mathbf{J}_i(\boldsymbol{\vartheta}) - 2 \sum_{i=1}^{d} \omega_i \partial_{\boldsymbol{\vartheta}}^2 \mathbf{x}_i(\boldsymbol{\vartheta})^T \left(\mathbf{y}_i - \mathbf{x}_i(\boldsymbol{\vartheta})\right).$$

Multivariate Gauss-Newton Method

To summarise:

- Initialize $\boldsymbol{\vartheta} = \boldsymbol{\vartheta}^0$.
- Do until convergence:

$$\mathbf{H}\left(\boldsymbol{\vartheta}^k\right) = \sum_{i=1}^{d} \omega_i \mathbf{J}_i\left(\boldsymbol{\vartheta}^k\right)^T \mathbf{J}_i\left(\boldsymbol{\vartheta}^k\right)$$

$$\mathbf{g}\left(\boldsymbol{\vartheta}^k\right) = \sum_{i=1}^{d} \omega_i \mathbf{J}_i\left(\boldsymbol{\vartheta}^k\right)^T \left(\mathbf{y}_i - x_i\left(\boldsymbol{\vartheta}^k\right)\right)$$

$$\boldsymbol{\vartheta}^{k+1} = \boldsymbol{\vartheta}^k - \mathbf{H}\left(\boldsymbol{\vartheta}^k\right)^{-1} \mathbf{g}\left(\boldsymbol{\vartheta}^k\right)$$

Remarks

- We can use sensitivity equations to obtain the derivatives in the Jacobian matrices.
- Unmeasured components are needed to solve the ODE system at each parameter value, and they may also appear in the sensitivity equations for the measured components.

Variable Weighting Using Error Variance

- Assume

$$\hat{\boldsymbol{\vartheta}} \sim N\left(\boldsymbol{\vartheta}, \Sigma_{\hat{\boldsymbol{\vartheta}}}\right)$$

where

$$\Sigma_{\hat{\boldsymbol{\vartheta}}} = \mathbf{H}(\hat{\boldsymbol{\vartheta}})^{-1} \left[\sum_{i=1}^{d} \omega_i^2 \sigma_i^2 \mathbf{J}_i(\hat{\boldsymbol{\vartheta}})^T \mathbf{J}_i(\hat{\boldsymbol{\vartheta}})\right] \mathbf{H}(\hat{\boldsymbol{\vartheta}})^{-1}.$$

- We can produce the same confidence and prediction intervals as above.
- If we choose $\omega_i = 1/\sigma_i^2$, we can simplify this expression to the inverse of the information matrix:

$$\mathbf{I}_{\hat{\boldsymbol{\vartheta}}}^{-1} = \left[\sum_{i=1}^{d} \frac{1}{\sigma_i^2} \mathbf{J}_i(\hat{\boldsymbol{\vartheta}})^T \mathbf{J}_i(\hat{\boldsymbol{\vartheta}})\right]^{-1}.$$

Estimate σ_i^2

- We can estimate σ_i^2 by

$$\hat{\sigma_i}^2 = \frac{1}{n-p} \sum_{j=1}^{n} \left(y_{ji} - x(t_j, \hat{\boldsymbol{\vartheta}}) \right)^2 .$$

  but we must have estimated $\boldsymbol{\vartheta}$ in order to obtain estimates for σ_i^2.
- A solution to this is to alternate estimating $\boldsymbol{\vartheta}$ and σ_i^2. This is a co-ordinate descent strategy for maximizing the log likelihood in a normal error model and can be generalized if the $\mathbf{y}_j \sim N(0, \Sigma)$ do not have independent errors.

Iterative procedure

- Initial starting points for σ_i^2.
- Minimizing the unweighted sum of squares-setting the σ_i^2 to be all equal-and obtain an initial ϑ.
- Alternatively, when observations are frequent, relative to the speed of the dynamics, successive differences between observations can be used as an estimate:

$$\tilde{\sigma}_i^2 = \frac{1}{n-1} \sum_{j=2}^{n} \left(\frac{y_{ji} - y_{(j-1)i}}{2} \right)^2 .$$

FitzHugh-Nagumo Models

The FitzHugh-Nagumo Models has a voltage V and a recovery variable R with the following dynamics:

$$\mathrm{D}V = c\left(V - V^3/3 + R\right)$$
$$\mathrm{D}R = -(V - a - bR)/c.$$

We add measurement errors:

$$y_{jv} = V(t_j) + 0.2\varepsilon_{jv}, \quad y_{jr} = R(t_j) + 0.5\varepsilon_{jr}.$$

FitzHugh-Nagumo Models

To fit these data, we start from the true values of the parameters: $(a, b, c) = (0.2, 0.2, 3)$ and initial conditions $(V0, R0) = c(-1, 1)$. The first five iterations of the Gauss-Newton algorithm now yield the following values:

Iteration	a	b	c	V_0	R_0
1	0.2000	0.2000	3.0000	−1.0000	1.0000
2	0.1932	0.0942	3.0066	−1.0893	1.0734
3	0.1925	0.0892	2.9964	−1.0763	1.0793
4	0.1928	0.0834	2.9947	−1.0761	1.0820
5	0.1928	0.0828	2.9946	−1.0759	1.0828
6	0.1928	0.0828	2.9946	−1.0759	1.0830

All these parameters have very quickly stabilized to at least three decimal places; though parameter b has moved a long way from its true value.

FitzHugh-Nagumo Models

The confidence intervals which all cover the true parameters:

	Lower	Upper
a	0.154	0.232
b	−0.232	0.398
c	2.918	3.071
V_0	−1.231	−0.921
R_0	0.890	1.276

FitzHugh-Nagumo Models

- We reweight the data according to our estimated variance, we get new parameter estimates, and then new estimated variances.
- Iterating this a few times, however stabilizes our estimated measurement standard deviations to 0.18 and 0.50. When using these variances we obtain parameter estimates

$$(a, b, c) = (0.179, 0.258, 2.977)$$

and initial conditions $(V_0, R_0) = (-1.076, 1.115)$.

FitzHugh-Nagumo Models

The confidence intervals:

	Lower	Upper
a	0.147	0.212
b	0.048	0.469
c	2.899	3.056
v_0	−1.204	−0.947
w_0	0.968	1.262

Local Minima

- One of the practical difficulties in fitting ODEs to data is that the shapes in the trajectories $x_i(t, \theta)$ can change with $\boldsymbol{\theta}$ in ways that make the squared error surface SSE($\boldsymbol{\vartheta}$) quite complex.
- While we seek an over-all minimum, the surface can have many local minima in which a Gauss-Newton method can get trapped.
- When solving the FitzHugh-Nagumo Model, we may obtain local minima.

Local Minima

- We can try many different starting values, but this improves the chances of finding a global optimum, it is no guarantee.
- There are also a collection of alternativemethods for searching for the minimum of a function over parameters. Many of these methods are computationally challenging; such as simulated annealing, which is designed to overcome complex objective surfaces, can require solving the ODE thousands of times.

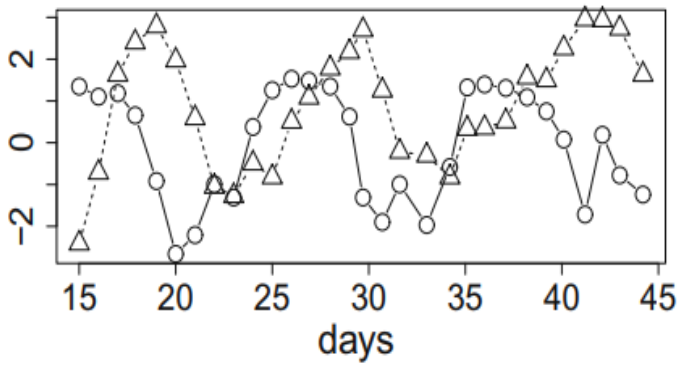
Initial Parameter Values

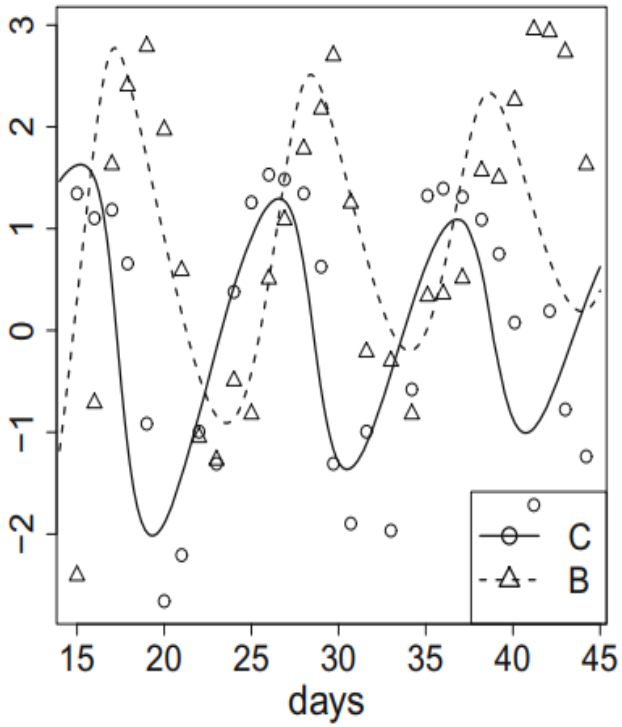
- We illustrate this with some data from an ecological experiment. These data come from a chemostat experiment in which algae are grown in a nitrogen-limited medium and serve as a prey species to rotifers, which are near-microscopic animals.
- We focus on the initial periods where classical predator-prey dynamics appears to hold and fit these with a Rosenzweig-MacArthur ODE model:

$$\mathrm{D}C = \rho C(\kappa - C) - \frac{\gamma\beta CB}{\chi + C}$$

$$\mathrm{D}B = \frac{\beta CB}{\chi + C} - \delta B.$$

Rosenzweig-MacArthur ODE model





Rosenzweig-MacArthur ODE model

- We fit the model on the logarithmic scale because it is clear that both the amplitude and the period of the oscillations do not match.
- If we simply start from these parameters and attempt to fit the data, we find very poor fits. Many optimization methods also return errors as parameters are forced into regions where numerical solvers break down.

Rosenzweig-MacArthur ODE model

- We see that the period at our initial parameters is around 50 days, which we would like to bring down to a little over 10 days as observed in the data.

$$\mathrm{D}x(at) = a[\mathrm{D}x](at)$$

- Multiplying each equation by a will "speed up" the period of the system by a units. Here, we chose a to be 4.54 and observe that this is equivalent to multiplying each of ρ, β and δ by a.

Rosenzweig-MacArthur ODE model

- We can re-scale variables; if we set $C^* = a_1 C, B^* = a_2 B$, then we can again apply the chain rule to discover that C^* and B^* satisfy the system with new parameters $\rho^* = \rho/a_1, \kappa^* = a_1\kappa, \chi^* = a_1\chi$ and $\gamma^* = (a_1/a_2)\gamma$.
- We minimized squared error on the log scale; representing our data as (C_j, B_j) at time t_j, we transformed the parameters to have approximately the correct timing as above and then minimized

$$\sum_{j=1}^{30} (\log C_j - \log(a_1 C(t_j, \boldsymbol{\theta})))^2 + (\log B_j - \log(a_2 B(t_j, \boldsymbol{\theta})))^2$$

for all of a_1, a_2 and $\boldsymbol{\theta}$.

Rosenzweig-MacArthur ODE model

- We updated the parameters as above to get rid of a1 and a2 and simply minimized squared error (on the log scale).
- Both minimizations were carried out using a Nelder-Meade method we then began from these solutions and used a Newton-Raphson-like method to find a convergent solution.

Remarks

- Using a Newton-type method at any earlier stages quickly resulted in errors, for example, that motivated trying alternative optimizers.
- A different model, or different data, might require quite different techniques; good fits may be found by optimizing early, on the other hand, it may be necessary to refine initial parameter guesses even further.

Remarks

There are some simple tricks that can be employed:

- Shift the inter-time intervals of equations by multiplying equations by a constant. Often this translates to changes in parameter values.
- Find a way to change state variables that can be mimicked by changes in parameters. (Re-scaling state variables, shifting the level of the state variable)
- In some cases, good initial conditions can be found from data directly. In others, especially where there is stable cyclic behavior, solving the system for a length of time and then choosing initial conditions from an appropriate point in the cycle can be helpful.

Remarks

- Optimizing over only some parameters to begin with can help get to better values.
- Alternatively, allowing some flexibility in the observation process-as in our estimation a_1 and a_2 above, can also be useful, even if we will ultimately re-absorb that into our parameter estimates.

Identifiability

- Parameter identifiability is an important issue in fitting ODE models to data.
- The re-scaling state variables in the Rosenzweig-MacArthur model as described above; it's reasonable to assume that we sample some percentage α of an ecosystem, so that our observations are of the form $(\alpha C(t_j) + \varepsilon_C, \alpha B(t_j) + \varepsilon_B)$. However, we cannot estimate α and all the other parameters jointly, even though they all have interpretable meanings.

Identifiability

In some cases, this type of confounding is straightforward to detect. Let's consider the Susceptible-Infected-Recovered ODE model for epidemics:

$$\begin{aligned} \mathrm{D}S &= -\alpha SI \\ \mathrm{D}I &= \alpha SI - \beta I \\ \mathrm{D}R &= \beta I. \end{aligned}$$

Susceptible-Infected-Recovered ODE model

However, in most epidemic models, we only measure the number of infected individuals and usually we only measure some percentage p of them; S and R are unmeasured states. Let's assume that we observe $I^* = pI$ perfectly and at all time points, then we have(dropping R)

$$\mathrm{D}S = -\frac{\alpha}{p} S I^*$$

$$\mathrm{D}I^* = \alpha S I^* - \frac{\beta}{p} I^*.$$

Had we observed both S and I, we should still be able to identify all three of α, β and p.

Susceptible-Infected-Recovered ODE model

However, we generally don't get to measure the Susceptible population and so would need to estimate at least its initial values, S_0 and we cannot distinguish this effect from others.
Let's write $S = S_0 S^*$, so that S^* starts from the value 1 , then we can write out an equation for (S^*, I^*):

$$\mathrm{D}S^* = -\frac{\alpha}{p} S^* I^*$$

$$\mathrm{D}I^* = \frac{\alpha}{S_0} \alpha S^* I^* - \frac{\beta}{p} I^*$$

where it should be clear that we cannot distinguish all of α, β, p and S_0 since they jointly define only three rates.
In this case, we often have an estimate of the over-all at-risk population size to use as S_0.

Summary

For a general ODE model, it is not necessarily easy to work out whether all the parameters and initial conditions can be determined by the data. Some simple measures can be used to identify whether parameters can be determined from a finite set of noisy data in practice:

- Perform a simulation experiment to identify the precision of the parameter estimates.
- Solve the sensitivity equations (at a plausible set of parameters) and test how linearly dependent they are.
- Perform an eigen decomposition of the Hessian $\partial^2_{\boldsymbol{\vartheta}} l(\hat{\boldsymbol{\vartheta}})$; very small or zero eigenvalues may indicate that some set of parameters are poorly identified.

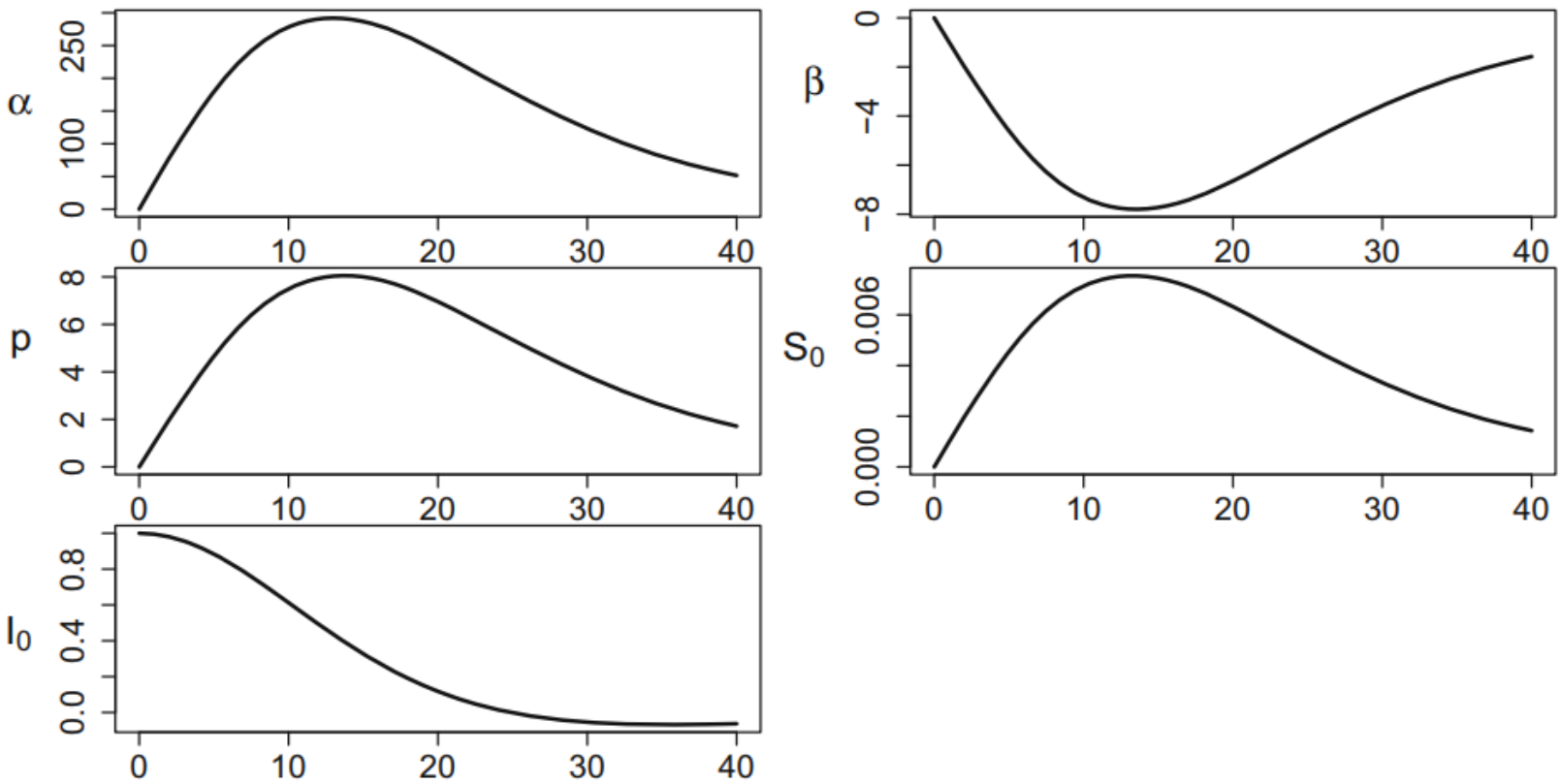
SIR model

We can take the SIR model as an example, where we assume we only measure I^* but had not recognized the problem of estimating S_0.

$$\mathrm{D}S^* = -\frac{\alpha}{p} S^* I^*$$

$$\mathrm{D}I^* = \frac{\alpha}{S_0} \alpha S^* I^* - \frac{\beta}{p} I^*.$$

SIR model



SIR model

We consider the sensitivity of the I^* equation with respect to initial conditions and parameters, using $\alpha = 0.005, \beta = 0.02$ and $p = 0.2$.

- Here we see strongly similar shapes in the sensitivity to α, p and S_0.
- If we perform a principal components analysis on these sensitivities, the final component has zero eigenvalue with a vector that contrasts the equations for α, p and S_0.
- If we remove the sensitivity to S_0- assuming it known and fixed when we estimate parameters—the smallest component contrasts the sensitivities to β and p but is still greater than zero.

Table of Contents

Review of Bayesian framework

- The Bayesian framework starts with prior distribution $\pi(\vartheta)$ over the space of possible parameter values. This prior is the key to Bayesian analysis.
- Our model specification can now be extended to

$$\boldsymbol{\vartheta} \sim \pi(\boldsymbol{\vartheta}), y_{ji} \mid \boldsymbol{\vartheta} \sim N\left(\mathbf{x}\left(t_j, \boldsymbol{\vartheta}\right), \sigma_i^2\right)$$

value of ϑ, we obtain a specific distribution for y_{ij}. The conditional density of the observation matrix Y given $\boldsymbol{\vartheta}$ can now be seen to be numerically equal to the likelihood

$$\begin{aligned}\log P(Y \mid \boldsymbol{\vartheta}) &= l(\boldsymbol{\vartheta}) \\ &= \sum_{i=1}^{d} \sum_{j=1}^{2} \frac{1}{\sqrt{2\pi}\sigma_i} \left(y_{ji} - x(t_j, \boldsymbol{\vartheta})\right)^2 - \frac{1}{2} \log 2\pi\sigma_i^2.\end{aligned}$$

Review of Bayesian framework

- The central concept in Bayesian statistics is that given $\pi(\boldsymbol{\vartheta})$ and $P(Y \mid \boldsymbol{\vartheta})$, we can apply Bayes theorem to recover the a posteriori density of ϑ conditional on Y :

$$P(\boldsymbol{\vartheta} \mid Y) = K^{-1} P(Y \mid \boldsymbol{\vartheta}) \pi(\boldsymbol{\vartheta}) = K^{-1} e^{l(\boldsymbol{\vartheta})} \pi(\boldsymbol{\vartheta})$$

  where K is a normalizing constant.

- This represents the update to our knowledge of $\boldsymbol{\vartheta}$ in the light of evidence from Y. Interpretation uses a probability distribution-first $\pi(\vartheta)$ and then $P(\vartheta \mid Y)$-to encapsulate our state of understanding before and after the experimental evidence.

Review of Bayesian framework

- A single best estimate for parameters is usually given by the expected value of $\boldsymbol{\vartheta}$(EAP):

$$\tilde{\boldsymbol{\vartheta}} = \int \boldsymbol{\vartheta} P(\boldsymbol{\vartheta}|Y) d\boldsymbol{\vartheta}.$$

- We can use the central $(1-\alpha)\%$ region of its distribution to create a credible regions.(Both the EAP and credible intervals perform equivalently to estimates and confidence intervals when n is large.)
- We can also use Markov Chain Monte Carlo methods to produce a sample of points $\vartheta_1, \ldots, \vartheta_M$ from the posterior distribution $P(\boldsymbol{\vartheta}|Y)$. We can then use these samples to obtain credible intervals and EAPs.

MCMC

A simple version of MCMC takes a two-step form. First, we propose a move from $\boldsymbol{\vartheta}^k$ to $\boldsymbol{\vartheta}^{k+1}$, we then accept that step and make a move, or reject it and stay at ϑ^k. This algorithm can be given as:

- Choose $\boldsymbol{\vartheta}^*$ with density $q\left(\boldsymbol{\vartheta}^* \mid \boldsymbol{\vartheta}^k\right)$ that depends on the current $\boldsymbol{\vartheta}^k$.
- Calculate

$$\alpha = \frac{P\left(Y \mid \boldsymbol{\vartheta}^*\right)}{P\left(Y \mid \boldsymbol{\vartheta}^k\right)} \frac{\pi\left(\boldsymbol{\vartheta}^*\right)}{\pi\left(\boldsymbol{\vartheta}^k\right)} \frac{q\left(\boldsymbol{\vartheta}^k \mid \boldsymbol{\vartheta}^*\right)}{q\left(\boldsymbol{\vartheta}^* \mid \boldsymbol{\vartheta}^k\right)}$$

- Set $\vartheta^{k+1} = \vartheta^*$ with probability α, otherwise $\vartheta^{k+1} = \vartheta^k$.

How to choose q

- The most common default choice is a symmetric random walk with normally distributed steps:

$$\boldsymbol{\vartheta}^* = \boldsymbol{\vartheta}^k + \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim N\left(0, \tau^2 \mathbf{I}\right).$$

- The correlation τ^2 in the resulting values not be too large; usually we target having an average value of α to be around 0.3.

FitzHugh-Nagumo data

- The parameters are a, b, c, V_0, R_0 as well as the measurement variances σ_V^2 and σ_R^2 and we use their logarithms which we write as s_V and s_R resulting in a parameter vector $\vartheta = (a, b, c, V_0, R_0, s_V, s_R)$.
- We give each element a $N(0, 10)$ prior, chosen to be uninformative. This gives us a model for the joint probability of our observations and parameters as

$$P(y_{1V}, \ldots, y_{41V}, y_{1R}, \ldots, y_{41R}, \vartheta)$$
$$= \prod_{j=1}^{41} \left[\phi\left(y_{jV}, V(t_j; \vartheta), e^{s_V}\right) \phi\left(y_{jR}, R(t_j; \vartheta), e^{s_R}\right)\right] \prod_{k=1}^{7} \phi(\vartheta_k; 0, 10)$$

  where $\phi\left(x; \mu, \sigma^2\right)$ is the normal density with mean μ and variance σ^2.
- We selected our proposal distribution for $\boldsymbol{\vartheta}$ to be Gaussian with independent dimensions.

FitzHugh-Nagumo data

- We determined the size of the steps by first maximizing

$$\log \ell(\vartheta) = \log P\left(y_{V1}, \ldots, y_{V41}, y_{R1}, \ldots, y_{R41}, \vartheta\right)$$

over ϑ and obtaining the Hessian H at the optimum.

- We then used proposed moves with standard deviations of $0.5\sqrt{\text{diag}(H)}$-about $1/2$ the sampling standard deviation from a Gaussian approximation. When making a proposed move from ϑ to ϑ' we calculated the acceptance probability as

$$\begin{aligned} P\left(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}'\right) = \exp\big[&\log P\left(y_{1V}, \ldots, y_{41V}, y_{1R}, \ldots, y_{41R}, \boldsymbol{\vartheta}'\right) \\ &- \log P\left(y_{1V}, \ldots, y_{41V}, y_{1R}, \ldots, y_{41R}, \boldsymbol{\vartheta}\right)\big] \end{aligned}$$

where calculating the probabilities on the log scale helps to stabilize the numerical calculation of their values.

FitzHugh-Nagumo data

- We started from the true parameter values and ran MCMC for 5000 steps (this includes the 79% of times that we did not accept the proposed move)
- We removed the first 1,000 values that we generated, and we took every 5th value of what remains(800).

FitzHugh-Nagumo data

Table 7.1 Bayesian expected a posteriori estimates and 95% credible intervals from MCMC results for the example in Sect. 7.4.4. These were obtained after running a Markov chain for 5000, steps, discarding the first 1000 and then taking every fifth value in the chain

	a	b	c	V_0	R_0	σ_V^2	σ_R^2
Truth	0.200	0.200	3.000	−1.000	1.000	0.200	0.500
EAP	0.178	0.212	2.965	−1.099	1.133	0.197	0.510
Lower	0.146	−0.022	2.871	−1.244	0.983	0.161	0.405
Upper	0.210	0.463	3.044	−0.966	1.275	0.244	0.655

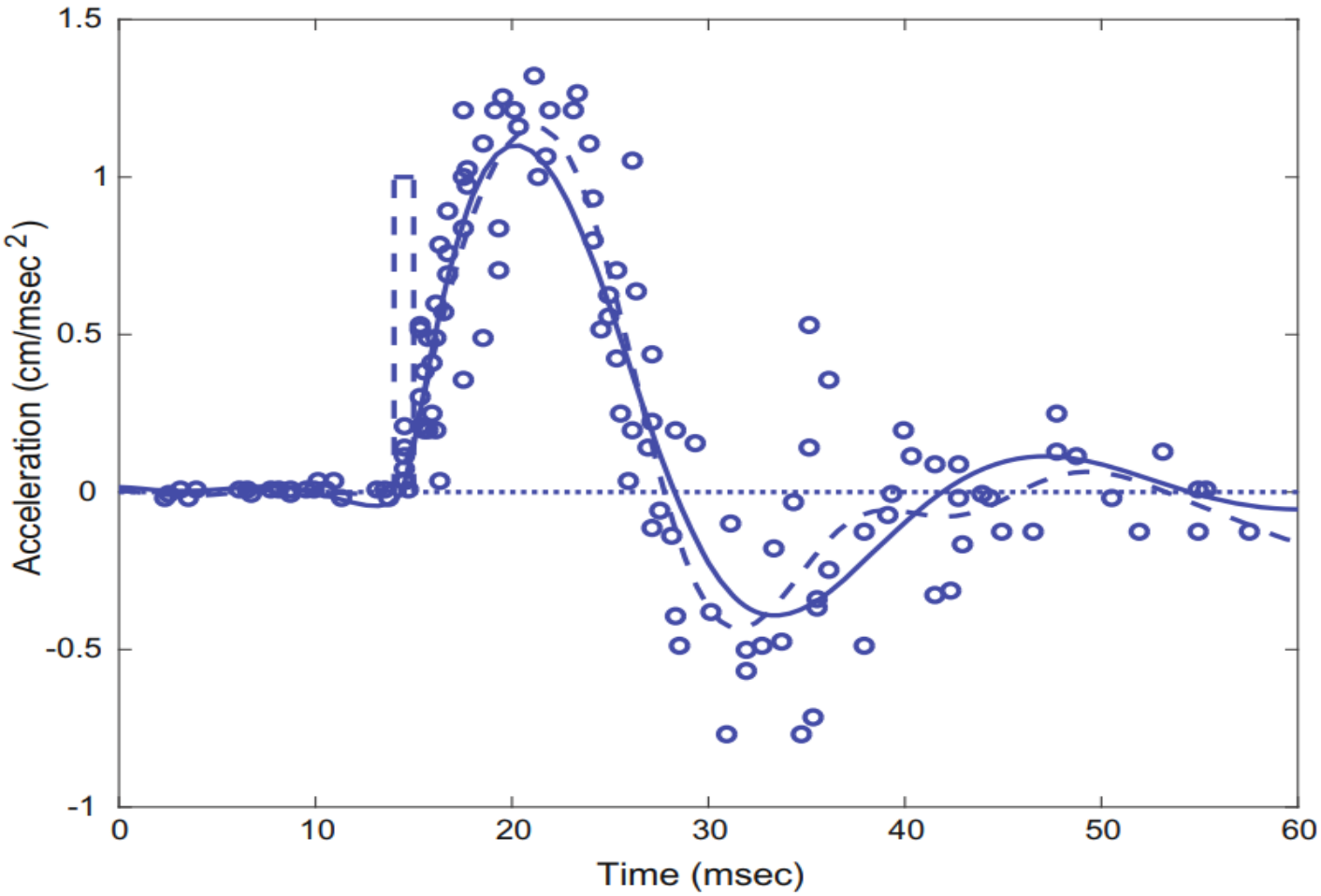
Bayesian intervals all cover the true values.

Table of Contents

Head Impacts

- The data were collected in a study to measure the effects of motorcycle accident on the driver's brain tissue.
- They were measurements of acceleration of brain tissue within the cranium of a corpse before and after being struck by a blunt object with a force typical in a collision by a motorcycle driver's head with a hard surface.

Head Impacts



Model

- The shape of the data indicates that there is a rapidly decaying oscillation in acceleration after the impact.
- We therefore analyzed the data using a three-parameter damped harmonic equation:

$$\mathrm{D}^2 x = \beta_0 + \beta_1 x + \beta_2 \mathrm{D} x + \beta_3 u(t)$$

where u is the box function located at the impact time and with unit height and width which is displayed in the figure. β_1 defines the period or frequency of oscillation(stiffness), β_2 determines the rate of decay of the oscillations. Impact pulse u is an external input to the system described by the first two terms on the right(a forcing term). β_3 is essentially a regression coefficient that determines the amplitude of the oscillations(gain in the system).

Solution

- By observing that we clearly start at a steadystate of 0. This means that we can assume initial conditions $(x(0), \mathrm{D}x(0)) = (0, 0)$ and that $\beta_0 = 0$.
- Expanding the equation to also describe velocity: $v = \mathrm{D}x$ results in

$$\mathrm{D}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \beta_1 & \beta_2 \end{pmatrix}\begin{pmatrix} x \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ u(t) \end{pmatrix}$$

Solution

- Appending the sensitivity equations

$$\mathrm{D}\begin{pmatrix} \partial_{\beta_1} v \\ \partial_{\beta_2} v \\ \partial_{\beta_3} v \end{pmatrix} = \beta_1 \begin{pmatrix} \partial_{\beta_1} x \\ \partial_{\beta_2} x \\ \partial_{\beta_3} x \end{pmatrix} + \beta_2 \begin{pmatrix} \partial_{\beta_1} v \\ \partial_{\beta_2} v \\ \partial_{\beta_3} v \end{pmatrix} + \begin{pmatrix} x \\ v \\ u \end{pmatrix} \mathrm{D}\begin{pmatrix} \partial_{\beta_1} x \\ \partial_{\beta_2} x \\ \partial_{\beta_3} x \end{pmatrix}$$
$$= \begin{pmatrix} \partial_{\beta_1} v \\ \partial_{\beta_2} v \\ \partial_{\beta_3} v \end{pmatrix}$$

  in which these six auxiliary equations also have initial conditions zero.

- We selected an initial guess $(\beta_1, \beta_2, \beta_3) = (-0.05, -0.075, -3)$ and then minimized squared error via 20 Gauss-Newton steps.
- The iteration is

$$\boldsymbol{\beta}_{j+1} = \boldsymbol{\beta}_j + \left[\partial_{\boldsymbol{\beta}} \mathbf{x}\left(\boldsymbol{\beta}_j\right)^T \partial_{\boldsymbol{\beta}} \mathbf{x}\left(\boldsymbol{\beta}_j\right)\right]^{-1} \partial_{\boldsymbol{\beta}} \mathbf{x}^T \left(\mathbf{y} - \mathbf{x}\left(\boldsymbol{\beta}_j\right)\right).$$

Results

- Running this algorithm reduces squared error from 252,265 to 69,912 resulting in parameters $(-0.072, -0.213, -19.1)$ and an estimated measurement standard deviation of 22.9.
- We can now calculate confidence intervals using the estimated variance

$$\text{var}(\hat{\boldsymbol{\beta}}) = \frac{1}{n}\|\mathbf{y} - \mathbf{x}(\boldsymbol{\beta})\|^2 \left[\partial_{\boldsymbol{\beta}}\mathbf{x}(\boldsymbol{\beta})^T \partial_{\boldsymbol{\beta}}\mathbf{x}(\boldsymbol{\beta})\right]^{-1}$$

resulting in confidence intervals $[-0.078, -0.066], [-0.257, -0.169]$ and $[-21.78 - 16.45]$ for β_1, β_2 and β_3 respectively.

Results

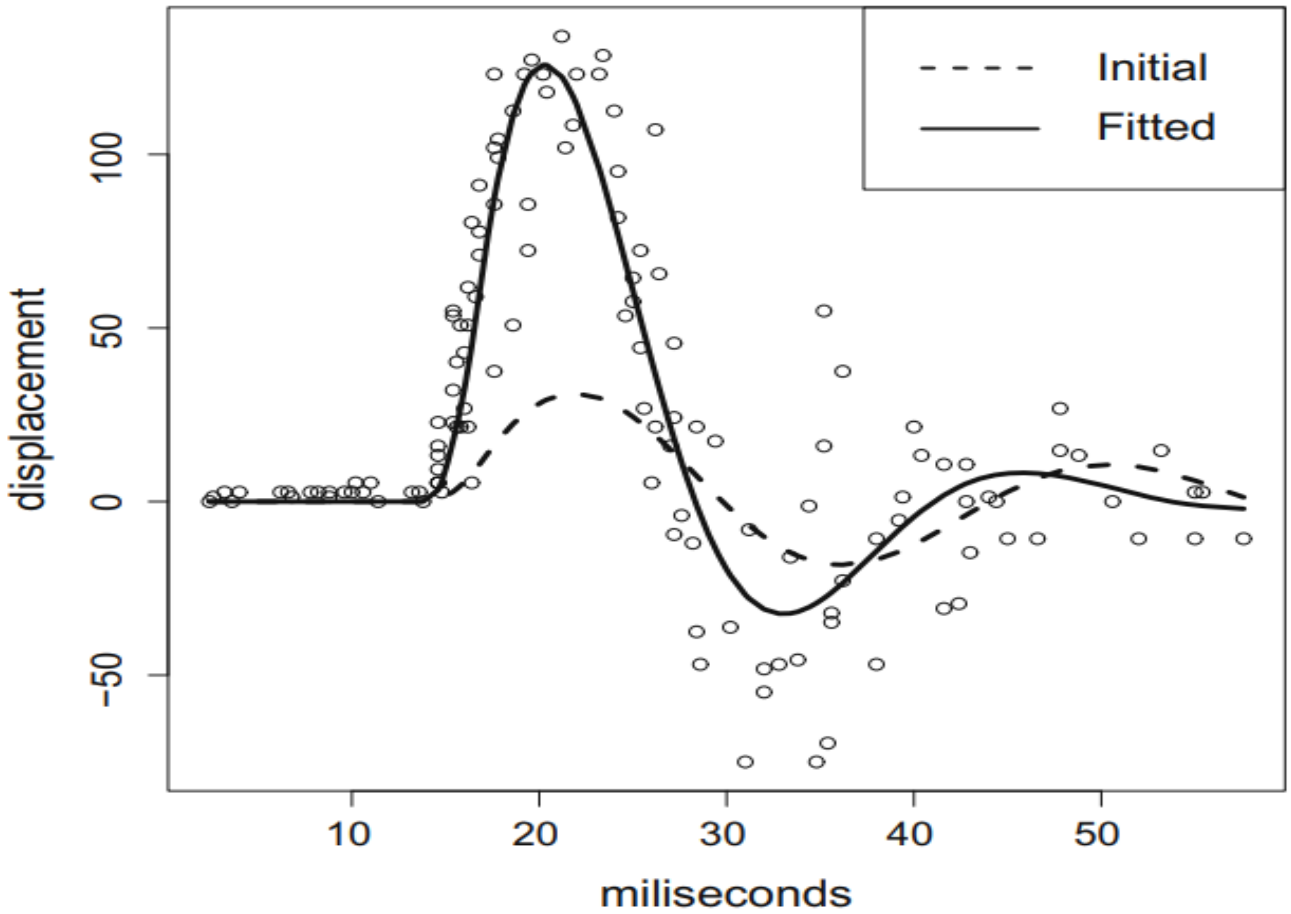

Initial
Fitted
displacement
100
50
0
−50
10
20
30
40
50
miliseconds

Table of Contents

Neural Ordinary Differential Equations

Paper 'Neural Ordinary Differential Equations' introduce a method combined with RNN and ODE solver.

- We treat time as the layers of a neural network:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{f}(t; \mathbf{x}(t), \boldsymbol{\theta}).$$

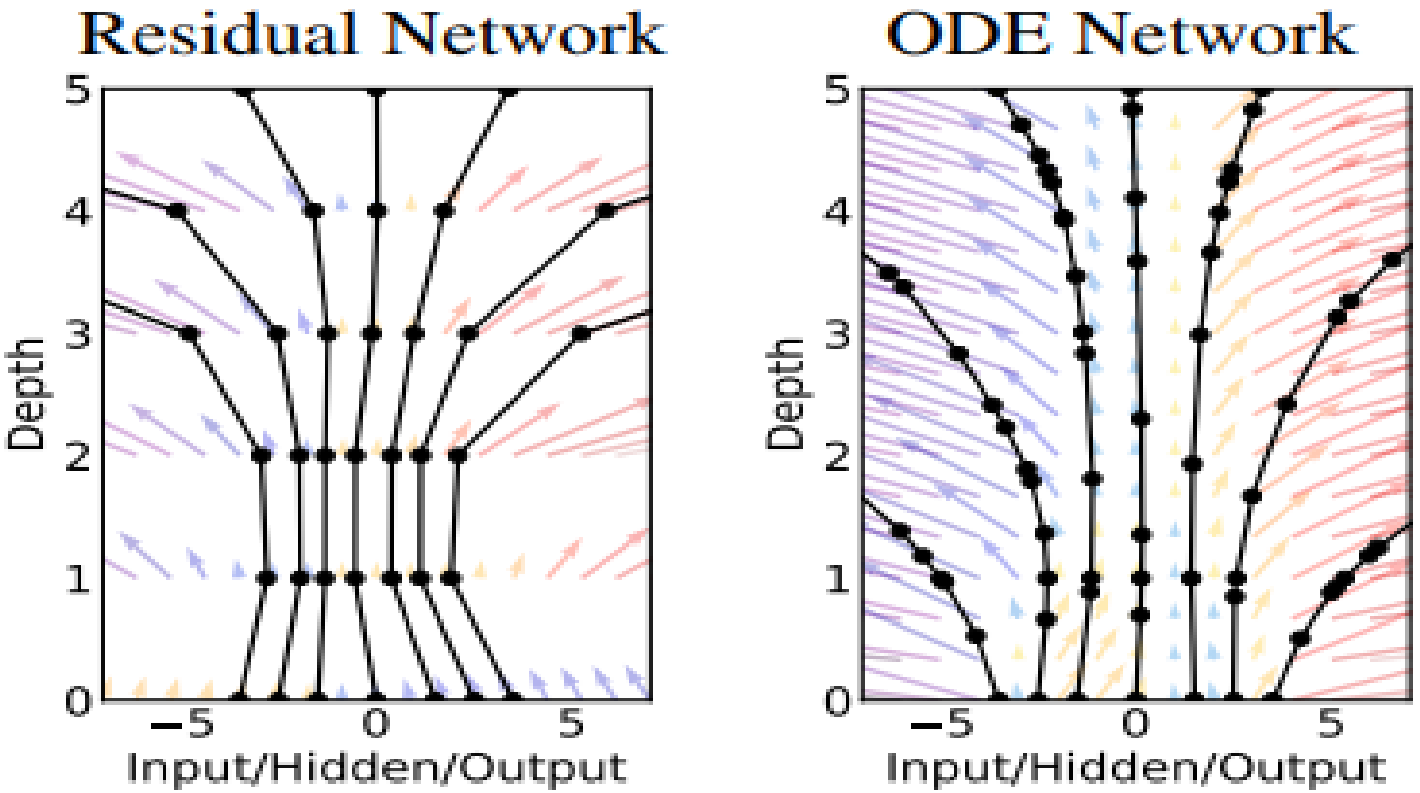
- In the limit, we parameterize the continuous dynamics of hidden units using an ordinary differential equation (ODE) specified by a neural network:

$$\mathrm{D}\mathbf{x}(t) = \mathbf{f}(t; \mathbf{x}(t), \boldsymbol{\theta}).$$

- Consider optimizing a scalar-valued loss function $L()$, whose input is the result of an ODE solver:

$$\begin{aligned} L\left(\mathbf{z}\left(t_1\right)\right) &= L\left(\mathbf{z}\left(t_0\right) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta) dt\right) \\ &= L\left(\text{ODESolve}\left(\mathbf{z}\left(t_0\right), f, t_0, t_1, \theta\right)\right). \end{aligned}$$

Neural Ordinary Differential Equations



Reverse-mode automatic differentiation of ODE solutions

To optimize L, we require gradients with respect to θ, let the adjoint be $\mathbf{a}(t) = \partial L / \partial \mathbf{z}(t)$.

- Determining how the gradient of the loss depends on the hidden state $\mathbf{z}(t)$ at each instant.

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^\top \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}}$$

- Computing the gradients with respect to the parameters θ requires evaluating a third integral, which depends on both $\mathbf{z}(t)$ and $\mathbf{a}(t)$:

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} \mathbf{a}(t)^\top \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt.$$

- The vector-Jacobian products $\mathbf{a}(t)^T \frac{\partial f}{\partial \mathbf{z}}$ and $\mathbf{a}(t)^T \frac{\partial f}{\partial \theta}$ can be efficiently evaluated by automatic differentiation.
- All integrals for solving $\mathbf{z}, \mathbf{a}$ and $\frac{\partial L}{\partial \theta}$ can be computed in a single call to an ODE solver.

Reverse-mode automatic differentiation of ODE solutions

Algorithm 1 Reverse-mode derivative of an ODE initial value problem

Input: dynamics parameters θ, start time t_0, stop time t_1, final state $\mathbf{z}(t_1)$, loss gradient $\partial L / \partial \mathbf{z}(t_1)$

$s_0 = [\mathbf{z}(t_1), \frac{\partial L}{\partial \mathbf{z}(t_1)}, \mathbf{0}_{|\theta|}]$ ▷ Define initial augmented state

def aug_dynamics$([\mathbf{z}(t), \mathbf{a}(t), \cdot], t, \theta)$: ▷ Define dynamics on augmented state

  return $[f(\mathbf{z}(t), t, \theta), -\mathbf{a}(t)^\mathsf{T} \frac{\partial f}{\partial \mathbf{z}}, -\mathbf{a}(t)^\mathsf{T} \frac{\partial f}{\partial \theta}]$ ▷ Compute vector-Jacobian products

$[\mathbf{z}(t_0), \frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}] = \text{ODESolve}(s_0, \text{aug\_dynamics}, t_1, t_0, \theta)$ ▷ Solve reverse-time ODE

return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}$ ▷ Return gradients

Benefits

- Memory efficiency: Compute gradients of a scalar-valued loss with respect to all inputs of any ODE solver, without backpropagating through the operations of the solver.
- Adaptive computation.
- Parameter efficiency: When the hidden unit dynamics are parameterized as a continuous function of time, the parameters of nearby "layers" are automatically tied together.
- Scalable and invertible normalizing flows.
- Continuous time-series models.