v0.3

A Primer on Variational Laplace (VL)

Peter Zeidman*, Karl Friston, Thomas Parr

Wellcome Centre for Human Neuroimaging, UCL

* peter.zeidman@ucl.ac.uk

This article details a scheme for approximate Bayesian inference, which has underpinned thousands of neuroimaging studies since its introduction 15 years ago. Variational Laplace (VL) provides a generic approach for fitting linear or non-linear models, which may be static or dynamic, returning a posterior probability density over the model parameters and an approximation of log model evidence, which enables Bayesian model comparison. VL applies variational Bayesian inference in conjunction with quadratic or Laplace approximations of the evidence lower bound (free energy). Importantly, update equations do not need to be derived for each model under consideration, providing a general method for fitting a broad class of models. This primer is intended for experimenters and modellers who may wish to fit models to data using variational Bayesian methods, without assuming previous experience of variational Bayes or machine learning. Accompanying code demonstrates how to fit different kinds of model using the reference implementation of the VL scheme in the open-source Statistical Parametric Mapping (SPM) software package. In addition, we provide a standalone software function that does not require SPM, in order to ease translation to other fields, together with detailed pseudocode. Finally, the supplementary materials provide worked derivations of the key equations.

1. Introduction

Scientific enquiry typically involves inferring quantities that cannot be directly observed. For example, a neuroscientist may wish to investigate the activity of neural populations from electrical activity that can be measured on the scalp. Similarly, a seismologist may wish to make inferences about geophysical events that occur beneath the surface of the earth, from seismograph measurements taken at the surface. Both of these are examples of *ill-posed problems*, where multiple configurations of the underlying system of interest could lead to similar measurements. Consequently, any inferences that are made about the underlying mechanisms that generate the data will typically involve uncertainty. This uncertainty needs to be quantified when drawing conclusions, which is why probabilistic, or *Bayesian* inference methods are typically used.

Any account of approximate Bayesian inference must begin with Bayes rule. For a model with a vector of parameters $\boldsymbol{\theta}$, we start by defining a *prior* probability density over the parameters $P(\boldsymbol{\theta})$. This function defines our belief about the probability of any particular value of the parameters, or range of values, before seeing the data. The priors serve to regularize or constrain the estimation of the model, making ill-posed problems tractable – in the sense there is a unique solution. We also define a likelihood $P(\boldsymbol{y}|\boldsymbol{\theta})$, which is a function of the parameters $\boldsymbol{\theta}$ and returns the probability of observing the data $\boldsymbol{y}$ given those parameters. Together, the priors and likelihood form a *generative model* of the data. The goal of Bayesian inference is two-fold. First, to obtain an updated probability density over the parameters after seeing the data – the posterior $P(\boldsymbol{\theta}|\boldsymbol{y})$. Second, to obtain the *marginal likelihood* or *model evidence* $P(\boldsymbol{y})$, which scores the quality of the model and enables models to be compared. If the data 'look like' the kind of data that the model would

have predicted, then $P(\boldsymbol{y})$ will be large, whereas if the data are inconsistent with the model, then $P(\boldsymbol{y})$ will be smaller, and thus models can be compared on the basis of their evidence. This is called *Bayesian model comparison* and we will return later to why the model evidence is well-suited to comparing models. The posterior and evidence are related by Bayes rule:

$$P(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{P(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{P(\boldsymbol{y})} = \frac{P(\boldsymbol{y},\boldsymbol{\theta})}{P(\boldsymbol{y})} \quad (1$$

Where the model evidence is the integral or sum over possible settings of the parameters:

$$P(\boldsymbol{y}) = \int P(\boldsymbol{y},\boldsymbol{\theta})d\boldsymbol{\theta} \quad (2$$

Thus, calculating the model evidence involves marginalising (summing or integrating) over the parameters. Unfortunately, this integral typically lacks an analytic solution and cannot be calculated directly. This is problematic for the calculating model evidence, but also the posterior probability (which requires the evidence).

It is the intractability of the integral in Eq 2 that necessities approximate Bayesian inference. Traditional sampling methods are a common approach for approximating the posterior, which eschew the need to tackle this integral, however they can be very computationally intensive and do not provide a straight-forward way to approximate the evidence, thereby precluding Bayesian model comparison. Instead, the approach described here is to construct a quantity that lower bounds the log of the evidence (i.e., is always smaller than or equal to it). This is often referred to as an evidence lower bound (ELBO) or a free energy functional[1]. An algorithm is then derived to maximise this bound – making it as close as possible to the log evidence. Methods that involve this optimisation of a bound – and therefore ensure that the ELBO or free energy approximates the log evidence – are collectively referred to as 'Variational Bayes' (VB). These methods were first introduced by Richard Feynman in statistical physics (Feynman, 1972) and terms such as 'free energy' were retained when it was subsequently applied in different fields. Variational methods were introduced into machine learning though ensemble learning (Hinton and Van Camp, 1993, MacKay, 1995b, MacKay, 1995a). Later, schemes like expectation maximisation (EM) were considered in the light of VB (Bishop, 1998, Neal and Hinton, 1998, Beal, 2003), which proved particularly useful for fitting graphical models (Jordan et al., 1999).

A key practical challenge for the routine use of VB is deriving the necessary model-fitting algorithm for a given model. This is time-consuming and requires a certain degree of skill with variational calculus. Approaches have therefore been developed over the years for implementing VB for a sufficiently broad class of models that new models can be introduced and fitted to data using 'plug-and-play' software routines. Variational Laplace (VL) is one such approach (Friston et al., 2007), which has been employed in a large body of work in neuroscience, including the dynamic causal modelling (DCM) framework for modelling neural circuitry (Friston et al., 2003). The reference implementation of VL is a MATLAB function, spm_nlsi_gn[2], implemented in the Statistical Parametric Mapping (SPM) software package (https://www.fil.ion.ucl.ac.uk/spm/).

[1] A *functional* is a function of a function.

[2] This routine is particularly useful for fitting continuous data when the likelihood has a Gaussian form (whose expectation and covariance may be non-linear functions of the parameters). An alternative MATLAB function, spm_nlsi_Newton, allows the

Previous technical reports have set out the relevant mathematics in detail (Daunizeau, 2017, Stephan et al., 2005, Friston et al., 2007). Here, our aim is didactic, with worked derivations of the key equations as well as exemplar code. Whereas the original description of the VL scheme assumed some familiarity with variational methods, statistical physics, Bayesian inference, as well as neuroscience, here we explain VL from first principles. This is particularly timely, as there is increasing interest in applying VL to fields beyond neuroimaging, including theoretical neurobiology (Smith et al., 2022), robotics (Lanillos et al., 2021, Lanillos and van Gerven, 2021) and epidemiology (Friston et al., 2020). We begin by deriving a score for the quality of a model: the free energy bound on the log evidence. Then, we derive the algorithm that optimises this bound, providing estimates of the log evidence and posterior over parameters. We illustrate this with worked examples, code for which is provided with this paper. Finally, in the discussion, we consider some advantages and disadvantages of this scheme and its relation to other variational inference software tools currently in use. An outline of mathematical notation appears in the footnote[3].

2. The generative model

The problem we will tackle involves modelling a vector of observed data $\boldsymbol{y}$ of length D. We have a model $g(\boldsymbol{\beta})$, where $\boldsymbol{\beta}$ are the model parameters:

$$\boldsymbol{y} = g(\boldsymbol{\beta}) + \boldsymbol{\epsilon_y} \tag{3}$$

The vector of errors or residuals $\boldsymbol{\epsilon_y}$ has a multivariate normal density, with mean zero, and precision matrix $\mathbf{\Pi_y}$ of dimension D (which is the inverse of the covariance matrix):

$$\boldsymbol{\epsilon_y} \sim N\left(\mathbf{0}, \mathbf{\Pi_y}^{-1}\right) \tag{4}$$

Elements on the leading diagonal of matrix $\mathbf{\Pi_y}$ are the precision (inverse variance) of each measurement, and any non-zero off-diagonal elements encode the conditional dependencies among measurements, which determines their correlation. To parameterize $\mathbf{\Pi_y}$ – such that it can be estimated from the data – we decompose it into a weighted mixture of K precision components $\boldsymbol{Q_{k=1,\ldots,K}} \in \mathbb{R}_{D\times D}$, each of which has a corresponding scalar hyperparameter $\lambda = (\lambda_1, \ldots \lambda_k)$:

$$\mathbf{\Pi_y}(\boldsymbol{\lambda}) = \exp(\lambda_1)\,\boldsymbol{Q_1} + \cdots + \exp(\lambda_k)\,\boldsymbol{Q_k} \tag{5}$$

The exponential of each hyperparameter is taken to enforce positivity (because precisions and variances cannot be negative). This approach provides a convenient method for allowing different mixtures of observations to share variance. At its simplest, a single precision component can be used, set to the identity matrix, $\boldsymbol{Q_1} = \boldsymbol{I_D}$, in which case the corresponding hyperparameter λ_1 encodes the log precision of the observation noise.

specification of generic likelihood functions (for example, it could be used for fitting binary data using likelihood densities compatible with logistic regression models).

[3] The following mathematical notation is used. **Variables**: Lower case italic text (x) is used for scalar variables, bold uppercase letters ($\boldsymbol{Y}$) are used for matrices and bold lower-case letters are used for vectors ($\boldsymbol{y}$). The expected value or average of variable x under the density Q is written $E_Q[x]$. **Calculus**: The partial derivative of a function $h(x)$ with respect to x is written $\partial_x\, h(x)$, therefore $\partial_x = \frac{\partial}{\partial_x}$. The second derivative of $h(x)$ is written $\partial_{xx}\, h(x)$. **Variational calculus:** The *variation* of a functional δ may be thought of as the derivative of the functional with respect to the function. For instance, for a functional of the form $A = \int S\big(h(x)\big)\, dx$, the variation of A with respect to h is: $\delta_{h(x)} S$.

To make this into a statistical model, we define a likelihood function $P(\boldsymbol{y}|\boldsymbol{\theta})$ and a prior probability density $P(\boldsymbol{\theta})$, where the parameters and hyperparameters are $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\lambda})$. As set out in the introduction, the likelihood $P(\boldsymbol{y}|\boldsymbol{\theta})$ returns the probability of observing the data $\boldsymbol{y}$ given a particular setting of the parameters $\boldsymbol{\theta}$. This has a multivariate normal density:

$$P(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{\lambda}) = N\left(g(\boldsymbol{\beta}), \boldsymbol{\Pi}_{\mathbf{y}}(\boldsymbol{\lambda})^{-\mathbf{1}}\right) \tag{6}$$

Eq 6 states that the observations are expected to be centred on the prediction of the model $g(\boldsymbol{\beta})$, with the level of observation noise given by the precision matrix $\boldsymbol{\Pi}_{\mathbf{y}}(\boldsymbol{\lambda})$. Next, the prior density $P(\boldsymbol{\theta})$ quantifies our belief about the probability of any given value of the parameters before performing the analysis. Here, we define multivariate normal densities as priors over the parameters and hyperparameters:

$$\begin{aligned} P(\boldsymbol{\beta}) &= N\left(\boldsymbol{\eta}_{\boldsymbol{\beta}}, \boldsymbol{\Pi}_{\boldsymbol{\beta}}^{-1}\right) \\ P(\boldsymbol{\lambda}) &= N\left(\boldsymbol{\eta}_{\boldsymbol{\lambda}}, \boldsymbol{\Pi}_{\boldsymbol{\lambda}}^{-1}\right) \end{aligned} \tag{7}$$

Having defined the model, we next set out methods for approximating the model evidence $P(\boldsymbol{y})$ and the posterior $P(\boldsymbol{\theta}|\boldsymbol{y})$.

3. Variational Bayes

We will convert the difficult problem of calculating the integral in Eq 2 into a simpler optimization or search problem. This begins by defining a functional called the free energy, which is a *lower bound* on the log of the model evidence $\ln P(\boldsymbol{y})$. This means that by construction, it can only return values that are less than or equal to the log evidence. Then we'll search for a setting of the parameters of the free energy (*variational parameters*) that maximize it, making it as close as possible to the unknown log evidence. Helpfully, the parameters that maximize the bound will turn out to approximate the posterior $P(\boldsymbol{\theta}|\boldsymbol{y})$.

3.1 Constructing a lower bound on the log evidence

The log evidence is defined as:

$$\ln P(\boldsymbol{y}) = \ln \int P(\boldsymbol{y}, \boldsymbol{\theta}) d\boldsymbol{\theta} \tag{8}$$

To construct a lower bound on this quantity, we will make use of Jensen's inequality, which says that the *average of a log* is always less than or equal to the *log of an average* (Figure 1). That means that if we can express $\ln P(\boldsymbol{y})$ as the average of a log, then to construct a lower bound we simply need to rearrange terms to get the log of an average.

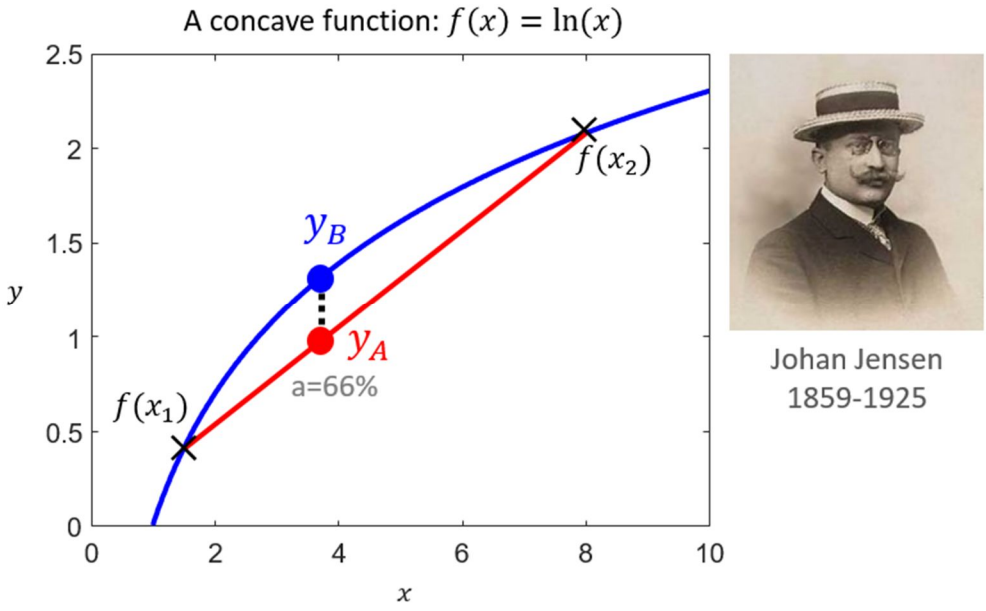


Figure 1 Jensen's inequality. The curved blue line could be any concave function $y = f(x)$. Here as an example, the function $f(x) = \ln(x)$ is shown, which has been evaluated in the range $x = [0, 10]$. The curved red *secant line* joints two arbitrary points: $f(x_1)$ and $f(x_2)$. Any point on the secant line, such as y_A, is a weighted average of $f(x_1)$ and $f(x_2)$: $y_A = af(x_1) + (1-a)f(x_2)$. Jensen's inequality says that this average-of-functions will always be less than or equal to the corresponding function-of-the-average: $y_B = f(ax_1 + (1-a)x_2)$. Thus, using the notation of expected values, $E[f(x)] \leq f(E[X])$. Portrait in the public domain via *Wikipedia. https://en.wikipedia.org/wiki/Johan_Jensen_(mathematician)* .

To express the log evidence as the average of a log, we first define a probability density $Q(\boldsymbol{\theta})$ that will form our approximation of the posterior over parameters. We then introduce this density into the log evidence (from Eq 8), multiplying and dividing so that it causes no overall change to the evidence:

$$\begin{aligned} \ln P(\boldsymbol{y}) &= \ln \int \frac{Q(\boldsymbol{\theta})}{Q(\boldsymbol{\theta})} P(\boldsymbol{y}, \boldsymbol{\theta})\, d\boldsymbol{\theta} \\ &= \ln\, E_{Q(\theta)}\left[\frac{P(\boldsymbol{y}, \boldsymbol{\theta})}{Q(\boldsymbol{\theta})}\right] \end{aligned} \tag{9}$$

Where $E[\cdot]$ is the expected value or average[4]. This is the log of an average, so (by Jensen's inequality) a lower bound on this will be the average of the log, which is called the free energy F:

$$F[Q(\boldsymbol{\theta})] = E_{Q(\boldsymbol{\theta})}\left[\ln \frac{P(\boldsymbol{y}, \boldsymbol{\theta})}{Q(\boldsymbol{\theta})}\right] \tag{10}$$

Here, we have written the free energy as a functional (a function of a function), with $Q(\boldsymbol{\theta})$ as its input. By construction, for any choice of probability density $Q(\boldsymbol{\theta})$, it holds that $F[Q(\boldsymbol{\theta})] \leq \ln P(\boldsymbol{y})$. Note that the free energy is sometimes defined to be the negative of this quantity – the definition given here is consistent with common conventions in statistics and machine learning. This is because the next step will be to find the probability density $Q(\boldsymbol{\theta})$ that maximizes the free energy, bringing it close to the log evidence, so $F \approx \ln P(\boldsymbol{y})$.

[4] For a probability density function $P(x)$, the expected value $E[P(x)]$ is a weighted average over the possible values of x, where each value is weighted by its probability, i.e., $E[P(x)] = \int xP(x)\, dx$. The expected value can also be taken with respect to another probability density function $Q(x)$, which we write as $E_{Q(x)}[P(x)] = \int Q(x)P(x)\, dx$. This is a weighted average of $P(x)$, where the weighting comes from $Q(x)$.

3.2 Useful properties of the free energy

The free energy can be rewritten in different ways to illustrate why it serves as a useful score for the quality of a model. First, we can rewrite it as the log evidence minus the Kullback-Leibler (KL) divergence between the true and approximate posterior:

$$\begin{aligned} F[Q(\boldsymbol{\theta})] &= E_{Q(\boldsymbol{\theta})}\left[\ln \frac{P(\boldsymbol{\theta}|\boldsymbol{y})P(\boldsymbol{y})}{Q(\boldsymbol{\theta})}\right] \\ &= E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{\theta}|\boldsymbol{y}) + \ln P(\boldsymbol{y}) - \ln Q(\boldsymbol{\theta})] \\ &= \ln P(\boldsymbol{y}) + E_{Q(\boldsymbol{\theta})}\left[\ln \frac{P(\boldsymbol{\theta}|\boldsymbol{y})}{Q(\boldsymbol{\theta})}\right] \\ &= \underbrace{\ln P(\boldsymbol{y})}_{\text{Log evidence}} - \underbrace{D_{KL}[Q(\boldsymbol{\theta}) \parallel P(\boldsymbol{\theta}|\boldsymbol{y})]}_{\text{Approximation error}} \end{aligned} \tag{11}$$

Where $D_{KL}[Q \parallel P]$ is the KL divergence from density P to Q, which is a (non-negative) measure of difference between the two densities. The log evidence is a fixed (but unknown) quantity, so if we can find a density $Q(\boldsymbol{\theta})$ that maximizes the free energy, then we minimize the divergence between the true posterior $P(\boldsymbol{\theta}|\boldsymbol{y})$ and the approximation $Q(\boldsymbol{\theta})$.

We can also re-write the free energy from Eq. 10 as the difference between the model's accuracy and its complexity:

$$\begin{aligned} F[Q(\theta)] &= E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{y}|\boldsymbol{\theta}) + \ln P(\boldsymbol{\theta}) - \ln Q(\boldsymbol{\theta})] \\ &= E_{Q(\boldsymbol{\theta})}\left[\ln P(\boldsymbol{y}|\boldsymbol{\theta}) + \ln \frac{P(\boldsymbol{\theta})}{Q(\boldsymbol{\theta})}\right] \\ &= \underbrace{E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{y}|\boldsymbol{\theta})]}_{\text{Accuracy}} - \underbrace{D_{KL}[Q(\boldsymbol{\theta}) \parallel P(\boldsymbol{\theta})]}_{\text{Complexity}} \end{aligned} \tag{12}$$

Here, the accuracy term is the expected log likelihood, and the complexity is how far the parameters have diverged from the prior to the approximate posterior. Thus, if we select the model with the most positive free energy out of several candidate models, then we inherently select the model that offers the best trade-off between accuracy and complexity (c.f., Occam's razor). Importantly, this definition of complexity takes into account the covariance among parameters, and thus enables the free energy to serve as a better approximation of the log evidence than approximations such as the AIC or BIC, which quantify complexity as a function of the number of parameters, regardless of their covariance (Penny, 2012b).

Finally, we can re-arrange the free energy in Eq 10 as follows:

$$\begin{aligned} F[Q(\boldsymbol{\theta})] &= E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{y}, \boldsymbol{\theta})] - E_{Q(\boldsymbol{\theta})}[\ln Q(\boldsymbol{\theta})] \\ &= \underbrace{E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{y}, \boldsymbol{\theta})]}_{\text{Expected (negative) energy}} + \underbrace{H[Q(\boldsymbol{\theta})]}_{\text{Entropy}} \end{aligned} \tag{13}$$

By analogy with its applications in statistical physics, the first term is referred to as an *energy*, and the latter is the Shannon entropy H of the posterior density $Q(\boldsymbol{\theta})$. A probability density $Q(\boldsymbol{\theta})$ with high entropy will be smooth (or in the limit, flat) over the possible values of $\boldsymbol{\theta}$, whereas a probability density with low entropy will have peaks around particular values. This means that if we have two candidate $Q(\boldsymbol{\theta})$ densities, both equally likely under the priors, to maximize the free energy we would select the one that is smoother or, equivalently, with the higher entropy. This is referred to as Jaynes' Principle of Maximum Entropy (Jaynes, 1957), and means that we select the simplest explanation for the data where possible. (A further consequence of Jaynes' Principle is that as we come closer to maximizing the free energy, the free

energy forms a smoother landscape, aiding the performance of optimization algorithms. We will return to the notion of a free energy landscape in Section 5.)

4. Free energy under the Laplace approximation

We now build up to the definition of the free energy that is used in the VL scheme, by first defining the probability densities that appear in the definition of the free energy (Eq. 10) and then by introducing a *mean-field partition* over parameters and hyperparameters.

4.1 Quadratic approximations

In what follows, we will approximate the various unknown probability densities that appear in Eq 10 using multivariate normal densities, by way of *quadratic approximations* and *Laplace's method*, which we will first reprise. A quadratic approximation, also called a second order Taylor approximation, approximates any (twice differentiable) function $g(\boldsymbol{x})$ that has vector input $\boldsymbol{x}$, close to its peak or mode $\boldsymbol{x} = \boldsymbol{x_0}$, with the linear function:

$$T(\boldsymbol{x}) = \underbrace{g(\boldsymbol{x_0})}_{\text{Constant}} + \underbrace{\nabla g(\boldsymbol{x_0}) \cdot (\boldsymbol{x} - \boldsymbol{x_0})}_{\text{Linear term=0}} + \underbrace{\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x_0})^T H_g(\boldsymbol{x_0})(\boldsymbol{x} - \boldsymbol{x_0})}_{\text{Quadratic term}} \tag{14}$$

$$= \underbrace{g(\boldsymbol{x_0})}_{\text{Constant}} + \underbrace{\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x_0})^T H_g(\boldsymbol{x_0})(\boldsymbol{x} - \boldsymbol{x_0})}_{\text{Quadratic term}} \tag{15}$$

where $\nabla g(\boldsymbol{x_0})$ is the gradient of function g evaluated at $\boldsymbol{x_0}$ and $H_g(\boldsymbol{x_0})$ is the Hessian matrix—a matrix of second derivatives—of g evaluated at $\boldsymbol{x_0}$, i.e., $\left[H_g(\boldsymbol{x_0})\right]_{i,j} = \partial_{x_i x_j} g(\boldsymbol{x_0})$. At the mode of the density, $\boldsymbol{x} = \boldsymbol{x_0}$, the linear term equals zero and thus disappears in Eq 14. This has a very similar form to the log of a normal density, which is utilized in *Laplace's method* for function approximation.

4.2 Laplace's method

Laplace's method leverages the quadratic approximation in order to approximate a function g near its mode as a (scaled) normal density, as detailed in Fig 2. If $g(x)$ is a function of scalar variable x, the Laplace approximation $L(x)$ is:

$$g(x) \approx L(x) \propto N(x; \mu, \pi^{-1})$$

$$\mu = x_0 \tag{16}$$
$$\pi = -\partial_{x_0 x_0} \ln g(x_0)$$

where μ is the mean and π is the precision. When $g(\boldsymbol{x})$ is a function of a vector of variables $\boldsymbol{x}$, the Laplace approximation $L(\boldsymbol{x})$ returns a (scaled) multivariate normal density:

$$g(\boldsymbol{x}) \approx L(\boldsymbol{x}) \propto N(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Pi}^{-1})$$

$$\boldsymbol{\mu} = \boldsymbol{x_0} \tag{17}$$
$$\boldsymbol{\Pi} = -\partial_{x_0 x_0} \ln g(\boldsymbol{x_0})$$

Where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Pi}$ is the precision matrix, i.e., the inverse of the variance-covariance matrix. We will apply this method several times to approximate the different quantities that comprise the free energy.

Laplace approximation of a gamma density G

1. Define the log of the function to be approximated:

$$f(x; a, b) = \ln G(x; a, b)$$

2. Find its mode (peak) x_0

$$x_0 = (a-1)/b$$

3. Approximate $f(x)$ at $x = x_0$ using Taylor approximation $T(x)$:

$$f(x; a, b) \approx T(x) = f(x_0) + \frac{1}{2}\partial_{x_0 x_0} f(x_0)(x - x_0)^2$$

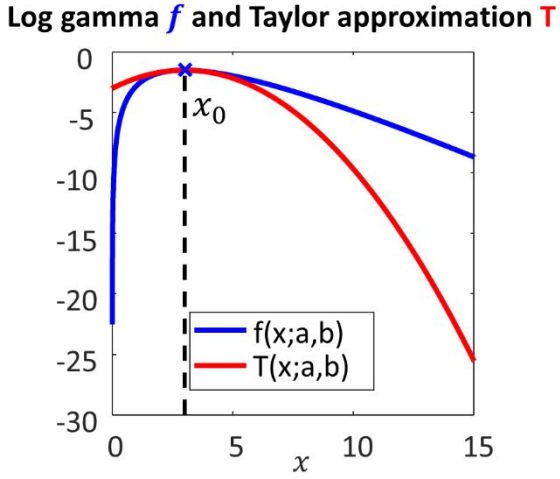
4. Take the exponential to approximate the underlying function:

$$L(x) = \exp[T(x; a, b)] = G(x_0; a, b) \exp\left[\frac{1}{2}\partial_{x_0 x_0} f(x_0)(x - x_0)^2\right]$$

Pierre-Simon Laplace
1749-1827

This is a (scaled) normal distribution with mean $\mu = x_0$ and precision $\pi = -\partial_{x_0 x_0} f(x_0)$:

$$L(x) = N(x; \mu, \pi^{-1}) \cdot G(x_0; a, b) \cdot \sigma\sqrt{2\pi}$$



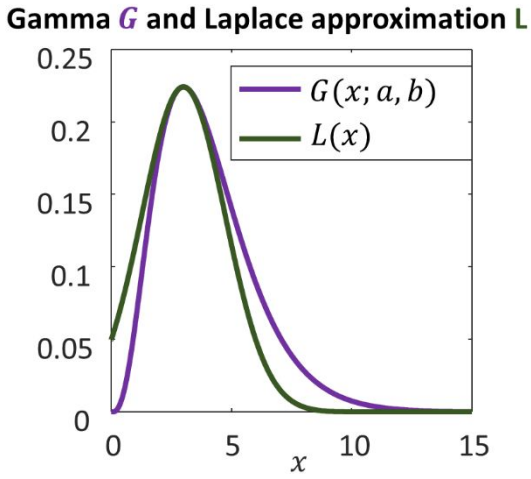


Figure 2 Laplace approximation of a non-normal density. This example illustrates Laplace's method by approximating a gamma probability density function $G(x; a, b)$ over the variable x with shape parameter a and inverse scale parameter b. The result is a scaled normal density $N(x; \mu, \pi^{-1})$ with mean $\mu = x_0$ and precision $\pi = -\partial_{x_0 x_0} \ln G(x_0)$, where x_0 is the mode of $\ln G$. After normalisation to ensure it is a proper probability density, the scaling becomes irrelevant, and we end up with a normal density. Portrait by James Posselwhite, in the public domain via *Wikipedia, https://en.wikipedia.org/wiki/Pierre-Simon_Laplace* .

4.3 Approximating the free energy

Recall Eq 10, the definition of the free energy:

$$F[Q(\boldsymbol{\theta})] = E_{Q(\boldsymbol{\theta})}\left[\ln \frac{P(\boldsymbol{y}, \boldsymbol{\theta})}{Q(\boldsymbol{\theta})}\right]$$

We will now write down approximations for the log joint $\ln P(\boldsymbol{y}, \boldsymbol{\theta})$ and the posterior $Q(\boldsymbol{\theta})$ and use these to form an expression for the free energy.

Approximating $\ln P(\boldsymbol{y}, \boldsymbol{\theta})$

We will apply a quadratic approximation for the log joint density. This is reasonable because densities tend to look normal close to their mode. Note that the mode $\boldsymbol{\mu}$ of the joint density, $P(\boldsymbol{y}, \boldsymbol{\theta})$, is the same as the mode of the posterior $P(\boldsymbol{\theta}|\boldsymbol{y})$,

because the posterior is simply a scaled version of the joint. (Also, $\boldsymbol{\mu}$ is the mode of the *log* of these densities, as taking the logarithm does not change the mode.) The approximation to the log joint is:

$$\ln P(\boldsymbol{y}, \boldsymbol{\theta}) \approx \ln P(\boldsymbol{y}, \boldsymbol{\mu}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^T \left[\partial_{\mu\mu} \ln P(\boldsymbol{y}, \boldsymbol{\mu})\right](\boldsymbol{\theta} - \boldsymbol{\mu}) \tag{18}$$

Choosing an approximate posterior $\boldsymbol{Q}$

We will similarly apply a quadratic approximation to the log posterior $\ln P(\boldsymbol{\theta}|\boldsymbol{y})$ around the posterior mode $\boldsymbol{\mu}$. Under Laplace's method, this means we will be using a normal density, $Q(\boldsymbol{\theta})$, in order to approximate the posterior $P(\boldsymbol{\theta}|\boldsymbol{y})$:

$$\begin{aligned} &\ln P(\boldsymbol{\theta}|\boldsymbol{y}) \approx \ln P(\boldsymbol{\mu}|\boldsymbol{y}) - \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}) \\ &\Longrightarrow P(\boldsymbol{\theta}|\boldsymbol{y}) \approx Q(\boldsymbol{\theta}) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &\qquad\qquad \boldsymbol{\Sigma^{-1}} = -\partial_{\boldsymbol{\mu\mu}} \ln p(\boldsymbol{\mu}|\boldsymbol{y}) = -\partial_{\boldsymbol{\mu\mu}} \ln p(\boldsymbol{\mu}, \boldsymbol{y}) \end{aligned} \tag{19}$$

Taking expectations

Next, we will apply the expectation operator[5] from Eq 10 to the previous two densities under the approximate posterior:

$$E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{y}, \boldsymbol{\theta})] \approx \ln P(\boldsymbol{y}, \boldsymbol{\mu}) + \frac{1}{2} E_{Q(\boldsymbol{\theta})}\left[(\boldsymbol{\theta} - \boldsymbol{\mu})^T \left(\partial_{\mu\mu} \ln P(\boldsymbol{y}, \boldsymbol{\mu})\right)(\boldsymbol{\theta} - \boldsymbol{\mu})\right] \tag{20}$$

$$E_{Q(\boldsymbol{\theta})}[\ln Q(\boldsymbol{\theta})] \approx -\frac{1}{2}[\ln(|\boldsymbol{\Sigma}|) + n \ln 2\pi] - \frac{1}{2} E_{Q(\theta)}[(\boldsymbol{\theta} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu})] \tag{21}$$

Where Eq 21 uses the definition of the log of the multivariate normal density for a variable with dimension n.

To simplify these expressions, note that each quadratic term inside the square brackets is a scalar. This means we can use the 'trace trick', $tr(\boldsymbol{ABC}) = tr(\boldsymbol{CAB})$. Applying this gives the simpler expressions:

[5] Note that this step ensures the final result is the same regardless of whether $\boldsymbol{\mu}$ is the mode of $P(\boldsymbol{\theta}|\boldsymbol{y})$ (prior to optimisation of the approximate posterior, it may not be). The linear term in the expansion will still disappear under the expectation as it includes a factor of $E_{Q(\theta)}[\boldsymbol{\theta} - \boldsymbol{\mu}]$ = 0, recovering Eq 20.

$$
\begin{aligned}
E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{y},\boldsymbol{\theta})] &\approx \ln P(\boldsymbol{y},\boldsymbol{\mu}) + \frac{1}{2}E_{Q(\boldsymbol{\theta})}\left[tr\left(\underbrace{(\boldsymbol{\theta}-\boldsymbol{\mu})^T}_{A}\underbrace{[\partial_{\mu\mu}\ln P(\boldsymbol{y},\boldsymbol{\mu})]}_{B}\underbrace{(\boldsymbol{\theta}-\boldsymbol{\mu})}_{C}\right)\right] \\
&= \ln P(\boldsymbol{y},\boldsymbol{\mu}) + \frac{1}{2}tr\left(E_{Q(\boldsymbol{\theta})}\left[\underbrace{(\boldsymbol{\theta}-\boldsymbol{\mu})}_{C}\underbrace{(\boldsymbol{\theta}-\boldsymbol{\mu})^T}_{A}\right]\underbrace{[\partial_{\mu\mu}\ln P(\boldsymbol{y},\boldsymbol{\mu})]}_{B}\right) \\
&= \ln P(\boldsymbol{y},\boldsymbol{\mu}) + \frac{1}{2}tr\big(\boldsymbol{\Sigma}\,\partial_{\mu\mu}\ln P(\boldsymbol{y},\boldsymbol{\mu})\big)
\end{aligned}
\tag{22}
$$

$$
\begin{aligned}
E_{Q(\boldsymbol{\theta})}[\ln Q(\boldsymbol{\theta})] &= -\frac{1}{2}[\ln(|\boldsymbol{\Sigma}|) + n\ln 2\pi] - \frac{1}{2}E_{Q(\theta)}\left[\underbrace{(\boldsymbol{\theta}-\boldsymbol{\mu})^T}_{A}\underbrace{\boldsymbol{\Sigma}^{-1}}_{B}\underbrace{(\boldsymbol{\theta}-\boldsymbol{\mu})}_{C}\right] \\
&= -\frac{1}{2}[\ln(|\boldsymbol{\Sigma}|) + n\ln 2\pi] - \frac{1}{2}tr\left(E_{Q(\theta)}\left[\underbrace{(\boldsymbol{\theta}-\boldsymbol{\mu})}_{C}\underbrace{(\boldsymbol{\theta}-\boldsymbol{\mu})^T}_{A}\right]\underbrace{\boldsymbol{\Sigma}^{-1}}_{B}\right) \\
&= -\frac{1}{2}[\ln(|\boldsymbol{\Sigma}|) + n\ln 2\pi] - \frac{1}{2}tr(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}) \\
&= -\frac{1}{2}[\ln(|\boldsymbol{\Sigma}|) + n\ln 2\pi] - \frac{n}{2} \\
&= -\frac{1}{2}[\ln(|\boldsymbol{\Sigma}|) + n\ln 2\pi e]
\end{aligned}
\tag{23}
$$

Interim summary

Substituting these expectations into Eq 10, we get the free energy under the Laplace approximation:

$$
\begin{aligned}
F[Q(\boldsymbol{\theta})] &= E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{y},\boldsymbol{\theta}) - \ln Q(\boldsymbol{\theta})] \\
&= \ln P(\boldsymbol{y},\boldsymbol{\mu}) + \frac{1}{2}tr\big(\boldsymbol{\Sigma}\,\partial_{\boldsymbol{\mu\mu}}\ln P(\boldsymbol{y},\boldsymbol{\mu})\big) + \frac{1}{2}[\ln(|\boldsymbol{\Sigma}|) + n\ln 2\pi e]
\end{aligned}
\tag{24}
$$

Where $\boldsymbol{\Sigma}$ is the posterior covariance and n is the total number of parameters. A useful feature of this expression is that, if we set the partial derivative of the free energy with respect to the covariance to be zero, we find an analytic form for the covariance that maximises the free energy (in terms of the expectation or mode), and recovers Eq 19:

$$
\partial_{\boldsymbol{\Sigma}}F = 0 \Leftrightarrow \boldsymbol{\Sigma}^{-1} = -\partial_{\boldsymbol{\mu\mu}}\ln P(\boldsymbol{y},\boldsymbol{\mu}) \tag{25}
$$

4.4 Free energy under a mean-field approximation

The previous section assumed that all parameters are treated equally. However, it is often helpful to separate them out into two (or more) types. The estimation scheme outlined here alternates between updating the estimate of the model's parameters $\boldsymbol{\beta}$ and the hyperparameters $\boldsymbol{\lambda}$ that control the precision of the observation noise. We previously lumped these two kinds of parameter together as $\boldsymbol{\theta}$ but now separate them out. We can re-write the joint probability as follows:

$$
P(\boldsymbol{y},\boldsymbol{\beta},\boldsymbol{\lambda}) = P(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{\lambda})P(\boldsymbol{\beta})P(\boldsymbol{\lambda}) \tag{26}
$$

with normal densities for the likelihood and priors (Eq 6 and 7). The approximate posterior is chosen to factorise as follows:

$$
Q(\boldsymbol{\theta}) = Q(\boldsymbol{\beta},\boldsymbol{\lambda}) = Q(\boldsymbol{\beta})Q(\boldsymbol{\lambda}) \tag{27}
$$

This factorisation is known as a mean-field approximation. Each approximate posterior is a multivariate normal density:

$$Q(\boldsymbol{\beta}) = N(\boldsymbol{\mu_\beta}, \boldsymbol{\Sigma_\beta})$$
$$Q(\boldsymbol{\lambda}) = N(\boldsymbol{\mu_\lambda}, \boldsymbol{\Sigma_\lambda}) \quad (28$$

The free energy is easily extended for this factorisation (from Eq 24):

$$F[Q(\boldsymbol{\theta})] = \underbrace{\ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}) + \frac{1}{2} tr\left(\boldsymbol{\Sigma_\beta}\, \partial_{\boldsymbol{\mu_\beta}\boldsymbol{\mu_\beta}} \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda})\right) + \frac{1}{2} tr\left(\boldsymbol{\Sigma_\lambda}\, \partial_{\boldsymbol{\mu_\lambda}\boldsymbol{\mu_\lambda}} \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda})\right)}_{E_{Q(\boldsymbol{\theta})}[\ln P(\boldsymbol{y}, \boldsymbol{\theta})]}$$
$$+ \underbrace{\frac{1}{2}\left[\ln\left(\left|\boldsymbol{\Sigma_\beta}\right|\right) + \ln(|\boldsymbol{\Sigma_\lambda}|) + (p+h)\ln 2\pi e\right]}_{E_{Q(\boldsymbol{\theta})}[\ln Q(\boldsymbol{\theta})]} \quad (29$$

Where p is the number of parameters and h is the number of hyperparameters. We can also extend Eq 25, to obtain the precision matrices that maximize the free energy:

$$\partial_{\Sigma_\beta} F = 0 \Leftrightarrow \boldsymbol{\Sigma_\beta^{-1}} = -\partial_{\mu_\beta \mu_\beta} \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}) \quad (30$$
$$\partial_{\Sigma_\lambda} F = 0 \Leftrightarrow \boldsymbol{\Sigma_\lambda^{-1}} = -\partial_{\mu_\lambda \mu_\lambda} \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}) \quad (31$$

Substituting these into Eq 29, we get:

$$F[Q(\boldsymbol{\theta})] = \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}) + \frac{1}{2}\left[\ln\left(\left|\boldsymbol{\Sigma_\beta}\right|\right) + \ln(|\boldsymbol{\Sigma_\lambda}|) + (p+h)\ln 2\pi e - (p+h)\right] \quad (32$$
$$= \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}) + \frac{1}{2}\left[\ln\left(\left|\boldsymbol{\Sigma_\beta}\right|\right) + \ln(|\boldsymbol{\Sigma_\lambda}|) + (p+h)\ln 2\pi\right]$$

So, we now just need to define the log joint, which is the product of three normal densities in Eq 26. The log of a normal density with mean $\boldsymbol{m}$ covariance matrix $\boldsymbol{S}$ for a variable of dimension k is defined as:

$$\ln N(\boldsymbol{x}; \boldsymbol{m}, \mathbf{S}) = -\frac{1}{2}[\ln|\mathbf{S}| + (\boldsymbol{x}-\boldsymbol{m})^T \mathbf{S^{-1}} (\boldsymbol{x}-\boldsymbol{m}) + k\ln(2\pi)]$$
$$= -\frac{1}{2}[\ln|\boldsymbol{S}| + \boldsymbol{\epsilon_x^T} \mathbf{S^{-1}} \boldsymbol{\epsilon_x} + k\ln(2\pi)] \quad (33$$

Where the *error term* is $\boldsymbol{\epsilon_x} = \boldsymbol{x} - \boldsymbol{\mu}$. Substituting into Eq 31 (with $\dim(y) = k$):

$$
\begin{aligned}
F[Q(\boldsymbol{\theta})] &= \ln P(\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{\lambda}) + \ln P(\boldsymbol{\beta}) + \ln P(\boldsymbol{\lambda}) + \frac{1}{2}\left[\ln\left(|\boldsymbol{\Sigma}_{\boldsymbol{\beta}}|\right) + \ln(|\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}|) + (p+h)\ln 2\pi\right] \\
&= \underbrace{-\frac{1}{2}\left[\ln\left|\boldsymbol{\Pi}_{\mathbf{y}}^{-\mathbf{1}}\right| + \boldsymbol{\epsilon}_{\boldsymbol{y}}^{T}\boldsymbol{\Pi}_{\mathbf{y}}\boldsymbol{\epsilon}_{\boldsymbol{y}}\right]}_{\text{Likelihood}} \\
&\underbrace{-\frac{1}{2}\left[\ln\left|\boldsymbol{\Pi}_{\boldsymbol{\beta}}^{-\mathbf{1}}\right| + \boldsymbol{\epsilon}_{\boldsymbol{\beta}}^{T}\boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}\right]}_{\text{Prior (parameters)}} \\
&\underbrace{-\frac{1}{2}\left[\ln\left|\boldsymbol{\Pi}_{\boldsymbol{\lambda}}^{-\mathbf{1}}\right| + \boldsymbol{\epsilon}_{\boldsymbol{\lambda}}^{T}\boldsymbol{\Pi}_{\boldsymbol{\lambda}}\boldsymbol{\epsilon}_{\boldsymbol{\lambda}}\right]}_{\text{Prior (hyperparameters)}} \\
&\underbrace{+\frac{1}{2}\left[\ln\left(|\boldsymbol{\Sigma}_{\boldsymbol{\beta}}|\right) + \ln(|\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}|)\right]}_{\text{Posterior entropy}} \\
&\underbrace{-\frac{1}{2}k\ln(2\pi)}_{\text{Constants}}
\end{aligned}
\tag{34}
$$

With error terms:

$$
\begin{aligned}
\boldsymbol{\epsilon}_{y} &= \boldsymbol{y} - \boldsymbol{g}(\boldsymbol{\mu}_{\beta}) \\
\boldsymbol{\epsilon}_{\beta} &= \boldsymbol{\mu}_{\beta} - \boldsymbol{\eta}_{\beta} \\
\boldsymbol{\epsilon}_{\lambda} &= \boldsymbol{\mu}_{\lambda} - \boldsymbol{\eta}_{\lambda}
\end{aligned}
\tag{35}
$$

We then use Eq 34 as the objective function (i.e., the measure of the model's quality), which we seek to maximize in order to approximate the log evidence and identify the model's parameters.

5. Estimation scheme

Next, we derive an algorithm for finding the posterior parameter density $Q(\boldsymbol{\theta})$ that maximizes the free energy $F[Q(\boldsymbol{\theta})]$. Full derivations of these equations are provided in the supplementary material.

5.1 Overview of gradient ascent and Gauss-Newton

The simplest approach to numerically maximizing the free energy is *gradient ascent*. Conceptually, the free energy forms a landscape, the dimensions of which are the parameters. Gradient ascent or descent takes small steps in the same direction as the gradient, as if climbing or descending a hill. Writing the function to be optimized generically as $f(\boldsymbol{\mu})$, the parameters $\boldsymbol{\mu}$ are updated on each iteration according to $\boldsymbol{\mu} = \boldsymbol{\mu} + \boldsymbol{\Delta\mu}$, where:

$$
\boldsymbol{\Delta\mu} = \alpha\boldsymbol{\nabla}_{\boldsymbol{\mu}}[f(\boldsymbol{\mu})] \tag{36}
$$

and α is the step size (positive for ascent, and negative for descent). This is illustrated in Figure 3A, for finding the minimum of a function using gradient descent. From an initial guess (labelled 1), small steps are taken towards a minimum, which in this example happens to be the global minimum (shaded green sphere).

Gradient ascent or descent is rarely sufficient in practice. One issue regards its speed: because the update is proportional to the gradient, estimates advance very slowly in shallow regions of the landscape (as can be seen in Figure 3A). Conversely, in very steep regions, the estimates advance quickly – running the risk of taking too large a step and moving

away from an optimum. We would ideally like the opposite situation – to move quickly in shallow regions, and slowly in steeper regions.

The Gauss-Newton algorithm takes a different approach and can converge far more quickly. To minimize a function, a parabola (or U-shaped plane) is fitted to $f(\boldsymbol{\mu})$ at the current estimate of the parameters (Figure 3B). The algorithm then jumps to the minimum of the parabola, which becomes the new parameter estimate (labelled 2 in Figure 3B) and the process repeats. This can enable fast progression from the initial estimate to the global or local optimum (Figure 3C). For some vector of parameters $\boldsymbol{\mu}^*$, the parabola is a second-order quadratic approximation of $f(\boldsymbol{\mu})$ at the current estimate of the parameters $\boldsymbol{\mu}$:

$$f(\boldsymbol{\mu}^*) \approx f(\boldsymbol{\mu}) + \nabla_{\mu} f(\boldsymbol{\mu}) \cdot (\boldsymbol{\mu}^* - \boldsymbol{\mu}) + \frac{1}{2}(\boldsymbol{\mu}^* - \boldsymbol{\mu})\boldsymbol{H}_{\boldsymbol{f}}(\boldsymbol{\mu})(\boldsymbol{\mu}^* - \boldsymbol{\mu}) \quad \textbf{(37}$$

Where $\boldsymbol{H}_{\boldsymbol{f}}$ is the Hessian matrix of second derivatives. This approximation has a unique minimum at $\boldsymbol{\mu} - \boldsymbol{H}_{\boldsymbol{f}}(\boldsymbol{\mu})^{-1} \cdot \boldsymbol{\nabla}_{\boldsymbol{\mu}} f(\boldsymbol{\mu})$, which becomes the new estimate of the parameters, giving rise to the update equation:

$$\boldsymbol{\Delta\mu} = -\boldsymbol{H}_{\boldsymbol{f}}(\boldsymbol{\mu})^{-1} \cdot \boldsymbol{\nabla}_{\boldsymbol{\mu}}[f(\boldsymbol{\mu})] \quad \textbf{(38}$$

While Gauss-Newton is much faster than gradient ascent, it will perform poorly if the quadratic approximation is poor – a particular risk when taking large steps that are far from the current estimate. In these situations, it would be ideal to dynamically switch to a gradient ascent or descent. A continuous transition between gradient ascent and Gauss-Newton like behaviour is achieved by the variational Laplace scheme, as we will return to shortly. As a first step towards this, we will derive a gradient ascent algorithm for the free energy.

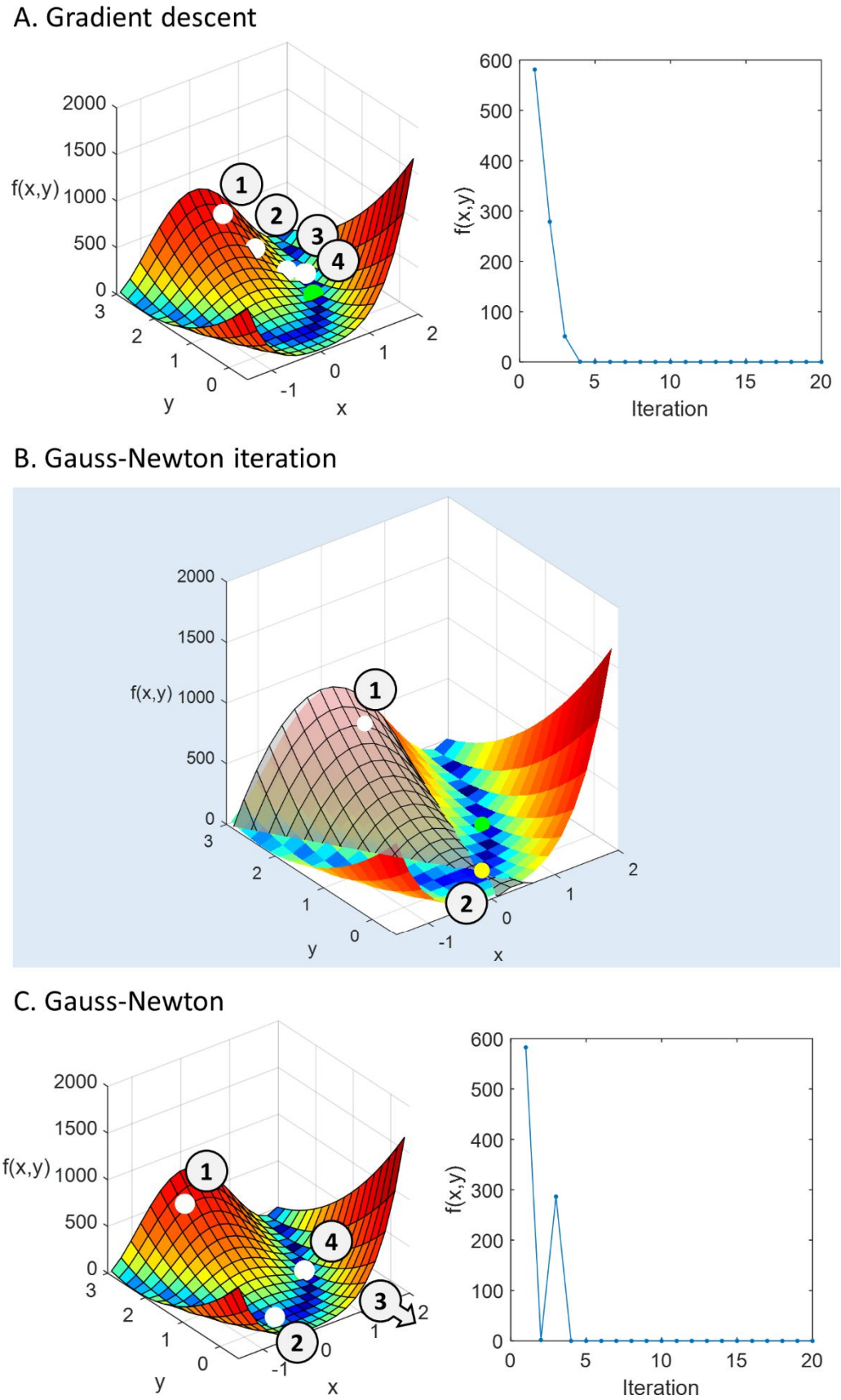


Figure 3 Comparison of Gradient ascent and Gauss-Newton optimization. The left panels illustrate an example function with two inputs $f(x, y)$ (the Rosenbrock function, with hyperparameters a=1, b=100). The global minimum is at x=1, y=1, indicated with a shaded green sphere in panel A. White spheres are estimates of the parameters (x,y) over successive iterations, indexed by the numbers in circles. The right panels show the evaluation of the function (i.e., the height of the surface) over successive iterations.

5.2 Gradient ascent on free energy

The mean-field approximation $Q(\boldsymbol{\theta}) = Q(\boldsymbol{\beta})Q(\boldsymbol{\lambda})$ naturally gives rise to a gradient ascent algorithm that alternates between optimising the parameters and hyperparameters. With the mean-field approximation, the free energy (Eq 10) can be extended to:

$$F[Q(\boldsymbol{\theta})] = E_{Q(\boldsymbol{\beta})Q(\boldsymbol{\lambda})}\left[\ln \frac{P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})}{Q(\boldsymbol{\beta})Q(\boldsymbol{\lambda})}\right] \tag{39}$$

The gradient, or more specifically the *variation* of the free energy with respect to each factor of the approximate posterior is (ignoring constants):

$$\begin{aligned} \delta_{Q(\boldsymbol{\beta})} F &= -\ln Q(\boldsymbol{\beta}) + E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] \\ \delta_{Q(\boldsymbol{\lambda})} F &= -\ln Q(\boldsymbol{\lambda}) + E_{Q(\boldsymbol{\beta})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] \end{aligned} \tag{40}$$

Setting to zero and taking the exponential, we get the optimal approximate posteriors on each iteration of the algorithm:

$$\begin{aligned} \delta_{Q(\beta)} F = 0 &\Leftrightarrow Q(\boldsymbol{\beta}) \propto \exp\left(E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})]\right) \\ \delta_{Q(\lambda)} F = 0 &\Leftrightarrow Q(\boldsymbol{\lambda}) \propto \exp\left(E_{Q(\boldsymbol{\beta})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})]\right) \end{aligned} \tag{41}$$

As before (Eq 22), quadratic approximations can be used for the terms inside the expectations:

$$\begin{aligned} E_{Q(\beta)}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] &\approx \ln P\left(\boldsymbol{y}, \boldsymbol{\mu}_{\beta}, \boldsymbol{\lambda}\right) + \frac{1}{2} tr\left(\boldsymbol{\Sigma}_{\boldsymbol{\beta}}\, \partial_{\mu_\beta \mu_\beta} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu}_{\beta}, \boldsymbol{\lambda}\right)\right) \\ E_{Q(\lambda)}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] &\approx \ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\mu}_{\boldsymbol{\lambda}}) + \frac{1}{2} tr\left(\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}\, \partial_{\mu_\lambda \mu_\lambda} \ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\mu}_{\boldsymbol{\lambda}})\right) \end{aligned} \tag{42}$$

By definition, the modes of the approximate posterior densities over parameters $\boldsymbol{\mu}_{\beta}$ and hyperparameters $\boldsymbol{\mu}_{\boldsymbol{\lambda}}$ must maximise the above quantities (i.e., the logarithms of the approximate posteriors). This gives the following pair of equations:

$$\begin{aligned}\boldsymbol{\Delta\mu_\beta} &= \nabla_{\mu_\beta} E_{Q(\lambda)}\big[\ln P\big(\boldsymbol{y},\boldsymbol{\mu_\beta},\boldsymbol{\lambda}\big)\big] \\ &= \nabla_{\mu_\beta} \ln P\big(\boldsymbol{y},\boldsymbol{\mu_\beta},\boldsymbol{\mu_\lambda}\big) + \nabla_{\mu_\beta} \frac{1}{2} tr\big(\boldsymbol{\Sigma_\lambda}\, \partial_{\mu_\lambda\mu_\lambda} \ln P\big(\boldsymbol{y},\boldsymbol{\mu_\beta},\boldsymbol{\mu_\lambda}\big)\big) \\ &= \underbrace{\mathbf{J}_{\boldsymbol{g}}^{T}\boldsymbol{\Pi_y}\boldsymbol{\epsilon_y}}_{\text{Likelihood}} - \underbrace{\boldsymbol{\Pi_\beta}\boldsymbol{\epsilon_\beta}}_{\text{Prior}} + \underbrace{\sum_{j=1}^{h}\big[(\boldsymbol{\Sigma_\lambda})_{jj}\boldsymbol{J_g^T}\boldsymbol{P_j}\boldsymbol{\epsilon_y}\big]}_{\text{Hyperparameters }\lambda}\end{aligned} \tag{43}$$

$$\begin{aligned}(\boldsymbol{\Delta\mu_\lambda})_i &= \partial_{\mu_{\lambda_i}} E_{Q(\beta)}[\ln P(\boldsymbol{y},\boldsymbol{\beta},\boldsymbol{\mu_\lambda})] \\ &= \partial_{\mu_{\lambda_i}} \ln P\big(\boldsymbol{y},\boldsymbol{\mu_\beta},\boldsymbol{\mu_\lambda}\big) + \partial_{\mu_{\lambda_i}} \frac{1}{2} tr\left(\boldsymbol{\Sigma_\beta}\partial_{\mu_\beta\mu_\beta} \ln P\big(\boldsymbol{y},\boldsymbol{\mu_\beta},\boldsymbol{\mu_\lambda}\big)\right) \\ &= \underbrace{\frac{1}{2} tr\big(\boldsymbol{P_i}\boldsymbol{\Pi_y^{-1}}\big) - \frac{1}{2}\boldsymbol{\epsilon_y^T}\,\boldsymbol{P_i}\boldsymbol{\epsilon_y}}_{\text{Likelihood}} - \underbrace{\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}}\boldsymbol{\Pi_\lambda}\boldsymbol{\epsilon_\lambda}}_{\text{Prior}} - \underbrace{\frac{1}{2} tr\big(\boldsymbol{\Sigma_\beta}\boldsymbol{J_g^T}\,\boldsymbol{P_i}\boldsymbol{J_g}\big)}_{\text{Parameters }\beta}\end{aligned}$$

Where vector $\boldsymbol{\Delta\mu_\beta}$ is the change in the parameters at each iteration of the algorithm, $(\boldsymbol{\Delta\mu_\lambda})_i$ is the change in the i-th hyperparameter at each iteration, $\nabla_{\boldsymbol{\mu_\beta}}[\cdot]$ is the gradient with respect to the parameters and $\boldsymbol{J_g}$ is the Jacobian matrix[6] of first partial derivatives with $\big(\mathbf{J}_{\boldsymbol{g}}\big)_{ij} = \partial_{\mu_j} g_i\big(\boldsymbol{\mu_\beta}\big)$ for observation i and parameter j. For hyperparameter i, the derivative term $\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})$ is a vector with value of unity in index i and zero elsewhere, and $\boldsymbol{P_i} = \partial_{\mu_{\lambda_i}}\big(\boldsymbol{\Pi_y}\big) = \exp(\lambda_i)\boldsymbol{\Pi_i}$. Terms including the second derivative of $g(\boldsymbol{\theta})$ have been omitted on the assumption that it is locally approximately linear. Similarly, the final term for the parameters is usually ignored, because its contribution is usually trivial in relation to the other terms, and is zero when the precision is linear in the hyperparameters.

These updates depend on first calculating the covariance of the parameters $\boldsymbol{\Sigma_\beta}$ and hyperparameters $\boldsymbol{\Sigma_\lambda}$, which are given by the expressions[7]:

$$\begin{aligned}\big(\boldsymbol{\Sigma_\beta}\big)^{-1} &= -\partial_{\mu_\beta\mu_\beta} E_{Q(\lambda)}\big[\ln P\big(\boldsymbol{y},\boldsymbol{\mu_\beta},\boldsymbol{\lambda}\big)\big] \\ &\approx \boldsymbol{J_g^T}\boldsymbol{\Pi_y}\boldsymbol{J_g} + \boldsymbol{\Pi_\beta}\end{aligned} \tag{44}$$

$$\begin{aligned}[(\boldsymbol{\Sigma_\lambda})^{-1}]_{i,i} &= -\partial_{\mu_{\lambda_i}\mu_{\lambda_i}} E_{Q(\beta)}[\ln P(\boldsymbol{y},\boldsymbol{\beta},\boldsymbol{\mu_\lambda})] \\ &= (\boldsymbol{\Pi_\lambda})_{ii} - \frac{1}{2} tr\big(\boldsymbol{P_i}\boldsymbol{\Sigma_y} - \boldsymbol{P_i}\boldsymbol{\Sigma_y}\boldsymbol{P_i}\boldsymbol{\Sigma_y}\big) + \frac{1}{2}\boldsymbol{\epsilon_y^T}\boldsymbol{P_i}\boldsymbol{\epsilon_y} + \frac{1}{2} tr\big(\boldsymbol{\Sigma_\beta}\boldsymbol{J_g^T}\boldsymbol{P_i}\boldsymbol{J_g}\big)\end{aligned}$$

[6] In the canonical implementation of the VL scheme (spm_nlsi_gn), the Jacobian is the negative of that defined here. For this reason, the sign of various terms is switched here compared to the original implementation.

[7] An important point here is that $\boldsymbol{\Sigma}$ is treated as constant with respect to the posterior expectation (seemingly contradicting the idea that the former is an analytic function of the latter). This is consistent with the Laplace approximation, as the assumption that the log joint is approximately quadratic implies that its curvature (and therefore the posterior covariance) is constant.

Where $\boldsymbol{\Sigma_y} = \boldsymbol{\Pi_y^{-1}}$. The gradient ascent proceeds by alternately applying the two updates in Eq 43. However, as mentioned above, a gradient ascent is rarely sufficient for robust performance, motivating an algorithm that dynamically switches between gradient ascent and Gauss-Newton-like updates. This is set out in the next section.

5.3 Updates in continuous time

Parameter updates in optimization are generally treated as occurring in discrete steps, $\boldsymbol{\Delta\mu_\beta}$ and $\boldsymbol{\Delta\mu_\lambda}$, as described above. However, Friston et al. (2007) considered updates occurring in continuous time, which provides a principled way to transition between gradient ascent and Gauss-Newton-like updates. In continuous time, the gradient ascent on the parameters can then be written in terms of the time derivative $\dot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}(t)$ at time t:

$$\dot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}(t) = \boldsymbol{\nabla}_{\boldsymbol{\mu_\beta}}\left[I_\beta\left(\boldsymbol{\mu_\beta}[t]\right)\right] \tag{45}$$

Where $I_\beta(\boldsymbol{\mu_\beta}) = E_{Q(\lambda)}\left[\ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\lambda})\right]$. Integrating Eq 45 over time using *local linearization* (Ozaki, 1985), the update for a time interval t is:

$$\begin{aligned} \boldsymbol{\Delta\mu_\beta} &= \left(\exp\left[t\ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}\right] - \boldsymbol{I}\right)\ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}{}^{-1}\dot{\boldsymbol{\mu}}_{\boldsymbol{\beta}} \\ \ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}} &= \boldsymbol{\nabla}_{\boldsymbol{\mu_\beta}}\left[\dot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}\right] = \partial_{\mu_\beta\mu_\beta} I_\beta(\boldsymbol{\mu_\beta}) \end{aligned} \tag{46}$$

And a similar expression is applied to update the hyperparameters. When t is small and positive, the algorithm behaves like a gradient ascent, because the term $\left(\exp\left[t\ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}\right] - \boldsymbol{I}\right)$ in Eq 46 regularizes the update (i.e., reduces its size). This follows because the second derivatives $\ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}$ are negative. As t increases, the matrix exponential term $\exp\left[t\ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}\right]$ reduces to zero, resulting in a standard Gauss-Newton update (by around $t = 2$):

$$\boldsymbol{\Delta\mu_\beta} = -\ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}{}^{-1}\dot{\boldsymbol{\mu}}_{\boldsymbol{\beta}} \tag{47}$$

The integration time t can therefore be varied dynamically to adjust the behaviour of the algorithm. When the algorithm begins, the time is set to a small value, causing it to behave like a gradient scheme for stability. If the free energy has increased, i.e., improved as a result of an update, then regularization is decreased (by increasing t). Conversely, if the free energy has decreased, i.e., worsened, then regularization is increased (by decreasing t).

In practice, the step size *t* is generally specified with a (log) descent parameter v that is automatically scaled by the average curvature of the landscape α:

$$\begin{aligned} t &= \frac{\exp(v)}{\alpha} \\ \alpha &= \exp\left[\frac{\mathrm{Re}\left(\ln\left|\ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}\right|\right)}{n}\right] \end{aligned} \tag{48}$$

Here, n is the number of parameters. A cautious, slow descent corresponds to $v = -4$ (the default starting value). As the average real eigenvalue of $\ddot{\boldsymbol{\mu}}_{\boldsymbol{\beta}}$ increases (i.e., the curvature increases), the regularisation is increased by decreasing v. The algorithm stops when the change in free energy becomes sufficiently small. In more detail, the predicted value of $I_\beta(\boldsymbol{\mu_\beta})$ after making the step $\boldsymbol{\Delta\mu_\beta}$ is given by: $\boldsymbol{\nabla}_{\boldsymbol{\mu_\beta}}\left[I_\beta(\boldsymbol{\mu_\beta})\right] \cdot \boldsymbol{\Delta\mu_\beta}$. If this is small over a series of iterations, then the algorithm is considered to have converged. Pseudocode for the complete VL algorithm is provided in Appendix 1, and MATLAB code accompanies this article.

5.4 Interim summary

This section described an algorithm that searches for a probability density over the parameters $Q(\boldsymbol{\theta})$ that maximizes the free energy. The algorithm ascends the free energy landscape, with an adaptive step size. Readers experienced with machine learning may note some similarity with the Levenberg-Marquardt algorithm, however the continuous time approach reviewed here is derived from first principles without requiring the introduction of an arbitrary regularization term: see Friston et al. (2007) for a detailed comparison.

6. Examples

This section presents worked examples, where the same algorithm that was introduced above is applied to simulated data from different kinds of model. MATLAB code for each example is provided with this paper.

6.1 Static models

We start with a simple linear regression model with parameters β:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon} \tag{49}$$

For this example, the design matrix $\boldsymbol{X}$ has two columns: the first is a column of ones and the second consists of 100 evenly spaced values between -50 and 50. Thus, the parameter β_1 encodes the mean and β_2 encodes the regression slope. The observation noise is encoded by a single precision component, controlled by log-precision parameter λ:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \boldsymbol{N}\left(\mathbf{0}, \boldsymbol{\Pi}_{\boldsymbol{y}}^{-\mathbf{1}}\right) \\ \boldsymbol{\Pi}_{\boldsymbol{y}} &= \exp(\lambda)\, \boldsymbol{I}_{\boldsymbol{n}} \end{aligned} \tag{50}$$

Note that a truly linear model would converge within a single iteration of the VL scheme, whereas here there was an exponential function in Eq 50, in order to ensure positivity of the hyperparameter λ. This introduces non-linearity into the model, thereby requiring several iterations to converge. Figure 4A shows that the parameters and hyperparameters used to generate the simulated data (grey bars) were successfully recovered (blue bars with 90% credible intervals). Note that the covariance among (hyper)parameters was also calculated by the VL scheme, but is not shown.

The second example considers the situation where there is *heteroskedasticity* – observations having different levels of observation noise. This is a common situation, for example when parts of the data come from different measurement channels. This can be modelled by having two precision components:

$$\boldsymbol{\Pi}_{\boldsymbol{y}} = \exp(\lambda_1)\, \boldsymbol{Q_1} + \exp(\lambda_2)\, \boldsymbol{Q_2} \tag{51}$$

Here, the first precision component matrix, $\boldsymbol{Q_1}$ has values of one on the leading diagonal for the first half of the observations, and zeros for the second half. It therefore captures the precision of the first half of the observations. Similarly, $\boldsymbol{Q_2}$ captures the precision of the second half of the observations. As shown in Figure 4B, the parameters and hyperparameters used to generate the data were correctly recovered. Next, we illustrate dynamic models, i.e., those modelling continuous changes over time.

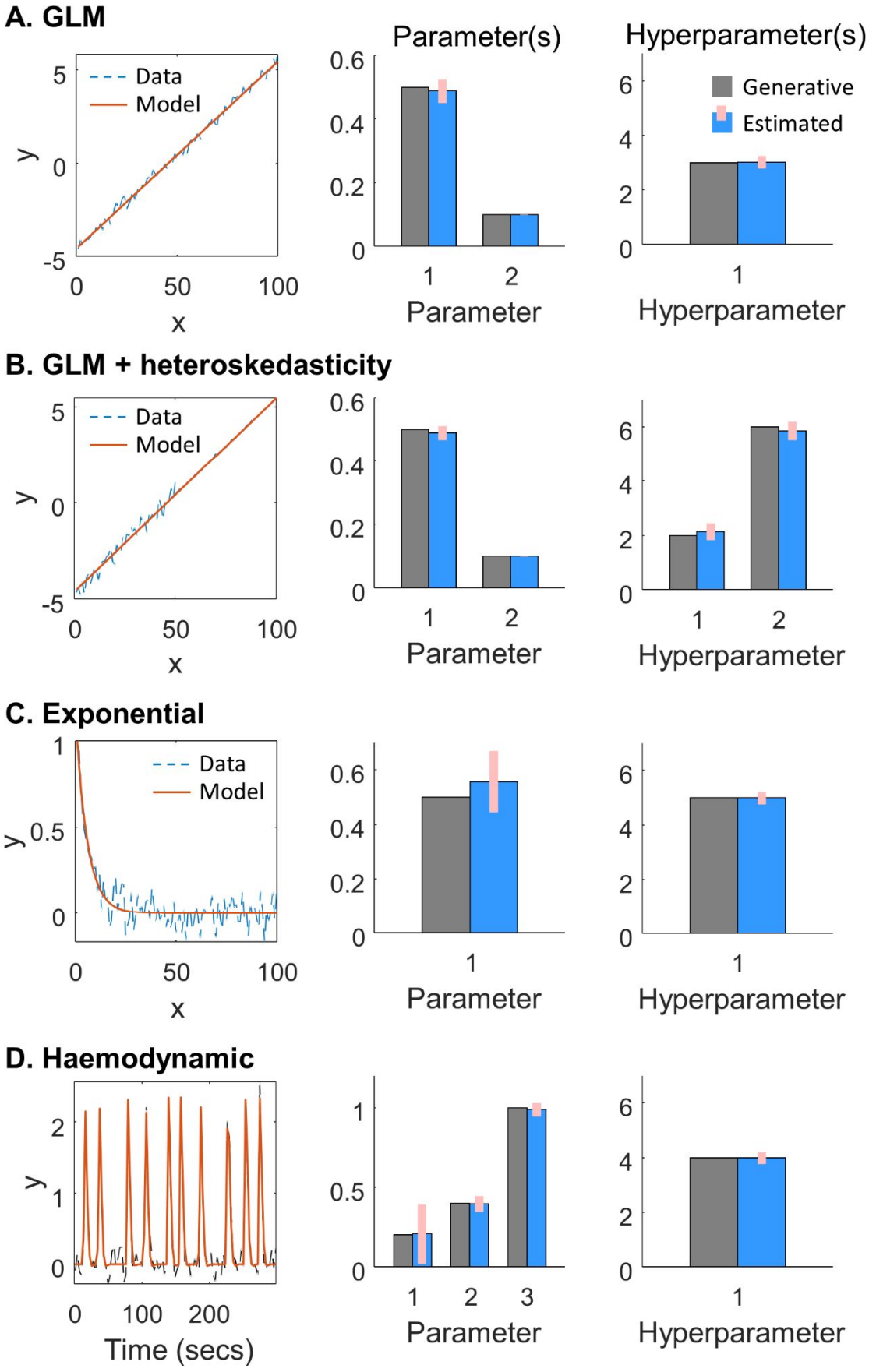


Figure 4 Example applications of the VL scheme. A-D illustrate the VL scheme applied to different models described in the text. **Left panels**: The simulated data (dashed blue lines) and prediction from the model (solid red lines). **Middle and right panels:** Parameter or hyperparameter values used to generate the data (grey bars without error bars) and posterior estimated values (blue bars with error bars).

6.2 Dynamic models

A typical application of the VL algorithm is to estimate the parameters of dynamic models that are specified as ordinary differential equations (ODEs). This is accomplished by splitting the model into two parts: a model f of the dynamics of the unobserved (latent) variables $\boldsymbol{x}$, and a static observation model g that translates the latent variables into observations $\boldsymbol{y}$:

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), \boldsymbol{\beta}) \\ \boldsymbol{y}(t) &= g\big(\boldsymbol{x}(t)\big) + \boldsymbol{\epsilon}\end{aligned} \tag{52}$$

Where $\boldsymbol{x}(t)$ is approximated using a standard numerical integration scheme, i.e.

$$\boldsymbol{x}(t) = \int_0^{\mathrm{t}} \dot{\boldsymbol{x}}(t)\, dt \tag{53}$$

Here, we perform this integration using the local linearization approach of Ozaki (1985). A simple example is estimating the rate of an exponential decay of a single observed variable x, with rate parameter β, for which the model specification corresponds to:

$$\begin{aligned}f(x(t), \beta) &= -\exp(\beta)\, x(t) \\ g\big(x(t)\big) &= x(t)\end{aligned} \tag{54}$$

The model fit and estimated parameters are shown in Figure 4C.

Finally, we consider a more involved example – the haemodynamic model of Stephan et al. (2007), which is used in the analysis of functional magnetic resonance imaging (fMRI) data to infer neural dynamics. For a detailed walkthrough of this model, please see Appendix 5 of Zeidman et al. (2019a). In brief, there are four hidden states (vasoactive signal s, blood inflow f_{in}, blood volume v, deoxyhaemoglobin q) and three time-invariant parameters that are estimated from the timeseries data (haemodynamic transit time τ_h, vasoactive signal decay rate κ and stimulus efficacy z). The dynamics of the four hidden states are governed by the following equations, which together constitute f in Eq 52:

$$\begin{aligned}\dot{f}_{in} &= s \\ \dot{s} &= z(t) - \kappa s - \gamma(f_{in} - 1) \\ \tau_h \dot{v} &= f_{in}(t) - f_{out}(v, t) \\ \tau_h \dot{q} &= f_{in}(t)\frac{1 - (1 - E_0)^{\frac{1}{f_{in}(t)}}}{E_0} - \frac{f_{out}(v, t) q(t)}{v(t)}\end{aligned} \tag{55}$$

Where $f_{out}(v, t) = v(t)^{\frac{1}{\alpha}}$ and α, γ are fixed parameters. The observation model (function g in Eq 52) translates from the latent variables for blood flow v and deoxyhaemoglobin q to the fMRI timeseries y:

$$\begin{aligned}y &= V_0\left(k_1(1 - q) + k_2\left(1 - \frac{q}{v}\right) + k_3(1 - v)\right) \\ k_1 &= 4.3 \cdot \vartheta_0 \cdot E_0 \cdot TE \\ k_2 &= \epsilon_h \cdot r_0 \cdot E_0 \cdot TE \\ k_3 &= 1 - \epsilon_h\end{aligned} \tag{56}$$

Where for this example, $\vartheta_0, E_0, r_0, TE, \epsilon_h$ are fixed parameters. Figure 4D shows that the parameters were recovered successfully, although the precision with which the transit time parameter could be recovered was lower than the other two parameters, as reflected in the larger 90% credible interval (pink bar).

6.3 Bayesian model comparison

The main purpose of the VL scheme is to enable Bayesian model comparison, which is comparing the relative evidence for different models, where the log evidence is approximated by the free energy. If each model encodes a hypothesis for how the data were generated, then Bayesian model comparison enables different hypotheses to be compared, in terms of how well they trade off accuracy and complexity (see section 3.2).

Models to be compared can differ in their likelihood – i.e., the definition of their forward model g – and/or in the specification of their priors. The only requirement is that all models have been fitted to the same data (where this is not the case, an alternative approach referred to as *Bayesian Data Comparison* may be considered, see Zeidman et al. (2019c)). Where models differ only in their priors, there is no need to fit each model separately to the data using the VL scheme – it is sufficient to fit one 'full' model, and use *Bayesian model reduction* to analytically compute the free energy and parameters of the alternative reduced models, under Laplace approximations (Friston et al., 2018). Through these methods, decisions such as whether to include particular variables in the model or how to capture their interactions can be decided in a principled manner.

To illustrate this, we will use GLM example above, where the first half of the observations were noisier than the second half (illustrated in Figure 4B, dashed lines, left panel). We will then use Bayesian model comparison to ask which is the best explanation for the data: a model with one precision component (i.e., all observations had the same level of noise), two precision components (the first and second half of the observations had different levels of observation noise) or three components (each third of the data had a different level of noise). Including more precision components could increase the model's accuracy by introducing more degrees of freedom, but would also increase model complexity. To assess which of the three models optimized the accuracy / complexity trade-off, we specified each model and calculated their free energy using the VL scheme. The values were $F_1 = -18.21, \mathrm{F}_2 = 39.61, \mathrm{F}_3 = 14.32$, where a more positive free energy is better.

These free energies can then be compared in terms of the *Bayes factor* (Kass and Raftery, 1995) and posterior probabilities. The Bayes factor is the ratio of evidence for a model m_1 relative to another model m_2:

$$B_1 = \frac{p(y|m_1)}{p(y|m_2)} \tag{57}$$

The larger the ratio, stronger the evidence in favour of model m_1. Taking the log, division becomes subtraction, therefore the log Bayes factor is simply the difference in log evidence. The log Bayes factor in favour of model 1 is:

$$\ln B_1 = \ln p(y|m_1) - \ln p(y|m_2) \approx F_1 - F_2 \tag{58}$$

The log Bayes factor can be computed for more than two models by selecting a model to serve as the baseline or reference. Here, we chose the worst model, m_1, as the reference model, and calculated log Bayes factor for models m_2 and m_3 relative to m_1, as shown in Figure 5 (left panel). As expected, m_2 was the best (as the data were generated using two precision components), m_3 was the second best, and m_1 was the worst model. This example also makes a key point, that accuracy is not an apt measure of model quality. The third model here would have been the most accurate because it has more degrees of freedom. However, its added complexity was correctly penalized by the free energy, ensuring it would be discarded in favour of the model that is the simplest explanation for the data – but not too simple.

It can aid interpretation to report not only the log Bayes factor, but also the posterior probability for each model, e.g., $p(m_1|y)$. Under equal prior probability for each model, by application of Bayes rule, the posterior probabilities are given by a softmax function of the log Bayes factors:

$$p(m_i|y) = \frac{p(y|m_i)}{p(y)} = \frac{1}{1 + \exp(-\ln B_i)} \quad (59$$

This is illustrated in Figure 5 (right panel). This demonstrates that m_2 had posterior probability close to unity, meaning that we could be extremely confident that it provided the best explanation for the data. This procedure may be applied with any number of models, and therefore forms the basis for hypothesis testing in Bayesian inference.

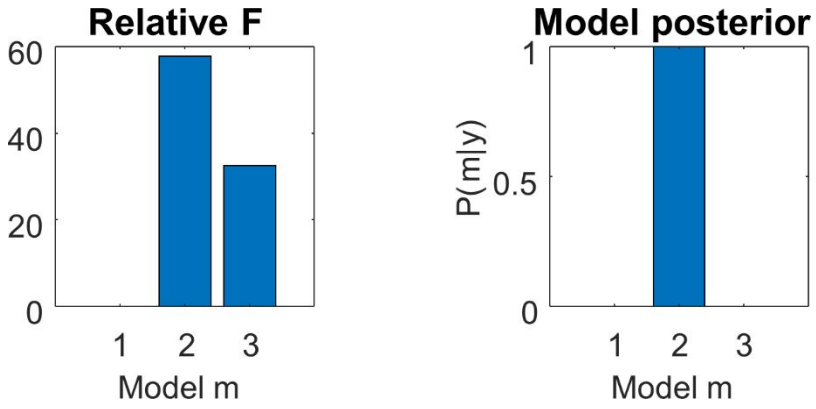


Figure 5 Bayesian model comparison. Three general linear models (GLMs) were fitted to the data using the VL scheme, and then compared based on their free energy. The models differed in whether they had one, two or three precision components. **Left:** The log Bayes factor for each model relative to model 1. **Right:** The same results transformed to a posterior probability for each model.

7. Discussion

The VL scheme described here underwrites thousands of neuroimaging studies, and is beginning to find applications in other fields. We considered it important, therefore, to clearly explain how it works, what assumptions it makes, and how to interpret the outputs. To this end, sections 1-3 set out the key challenge of Bayesian inference – the intractable integral within the model evidence (Eq 2) – and how this can be resolved using variational Bayes (a.k.a., approximate Bayesian inference). This involves defining a lower bound on the log evidence (the free energy, Eq 10), and then identifying a probability density over the parameters that maximizes this bound, bringing it as close as possible to the log evidence. In effect, variational procedures convert an impossible integration or marginalisation problem into a tractable optimisation problem. Section 4 set out the implementation of variational Bayes typically used in neuroimaging (Eq 34), where Laplace approximations (i.e., normal densities) are used to approximate the free energy bound. Section 5 then described an efficient algorithm for maximizing this (approximate) free energy, as illustrated for static and dynamic models in Section 6. Worked derivations are provided in the supplementary materials, pseudocode is provided in the appendix and standalone MATLAB code accompanies this article.

This scheme has several advantages over alternative methods, both in terms of the free energy approximation of the log evidence and the algorithm used to maximize it. First, the free energy serves as a better approximation of the log evidence than other commonly used heuristics like the AIC and BIC. These heuristic approximations can fail dramatically even in

relatively benign settings – see (Penny, 2012a) for some unsettling examples. While all three measures can be decomposed into accuracy and complexity terms, only the free energy takes into account uncertainty in the parameters for the accuracy term, as well as the covariance among parameters in the complexity term (Penny, 2012b). VL also has advantages over non-variational sampling methods. While sampling is highly effective for profiling the shape of a probability density, it does not provide a straightforward way to estimate the log evidence (one common approach, the *harmonic mean*, has been described as the "Worst Monte Carlo Method Ever" for its poor performance[8]). The lower computational cost of VL relative to sampling schemes, together with the fact that it is deterministic (so always provides the same results given the same data) provide further advantages. Regarding the algorithm described here for maximizing the free energy, the closest alternative is Expectation Maximization (EM). EM differs from variational Bayes in that EM ignores uncertainty about the hyperparameters. Thus, an advantage of VL is that it conveys the uncertainty of both the parameters and hyperparameters to the next iteration of the algorithm (i.e., EM is a special case of the variational Bayes where uncertainty about the hyperparameters is ignored.)

There are potential drawbacks of the VL scheme. First, the Laplace assumption may not be suitable for all applications. For instance, a Gaussian posterior may not be appropriate where the true posterior is multimodal, if a multimodal posterior is important for making inferences. To evaluate this for a particular application, the validity of the Laplace assumption can be assessed using sampling methods. This can be particularly useful when dealing with highly nonlinear models. Typically, variational Laplace accommodates nonlinearities by applying gaussian assumptions to nonlinear transformations of the parameters. A nice example of this is the treatment of hyperparameters above (Eq 50-51). By taking the exponential of the hyperparameter, one is effectively assuming a log normal prior, which ensures positivity for scale parameters of this sort (a scale parameter is a nonnegative parameter, such as a rate or time constant, variance or distance measure). Using the same device in hierarchical and nonlinear models allows one to accommodate a large range of (weakly) non-linear models within variational Laplace. However, it is sometimes necessary to check the robustness to violations of the Laplace assumption with reference to sampling schemes.

Second, the algorithm presented here is highly likely to converge to an optimal value, but only if the initial values of the parameters are well chosen (typically, the prior expected values of the parameters are used). In other words, if there are multiple local optima, then the algorithm may not be able to escape the local optimum that is easiest to reach from the starting value (this is sometimes expressed as starting within the basin of attraction of a fixed point in the free energy landscape). To overcome this, multi-start approaches have been used in conjunction with the VL scheme. For example, in the analysis of neuroimaging data from multiple test subjects, a common approach is to iteratively restart the algorithm from the group average parameter values (Friston et al., 2015). Finally, it should be noted that variational Bayes methods commonly suffer from overconfidence in their posterior parameter estimates, as demonstrated in the context of DCM for fMRI by Daunizeau et al. (2012).

Various extensions and variants of the VL scheme have been developed to handle a broader range of models in the context of neuroimaging, which we have not had space to detail in this article. For example, VL has been applied to modelling data in the frequency domain (complex cross-spectra), which is routinely used in the analysis of

[8] https://radfordneal.wordpress.com/2008/08/17/the-harmonic-mean-of-the-likelihood-worst-monte-carlo-method-ever/

electrophysiological data (Moran et al., 2009) and resting state fMRI data (Friston et al., 2014). Similar approaches have been introduced to invert stochastic differential equation models, namely Dynamic Expectation Maximization (DEM) and Generalised Filtering (GF), which estimate a model's hidden states and parameters, treating both as time-dependent random variables (Friston et al., 2010, Friston et al., 2008). More recently, the Parametric Empirical Bayes (PEB) framework was introduced to extend the VL scheme to hierarchical experimental designs, where for example, data have been sampled from multiple subjects at multiple time points (Zeidman et al., 2019b, Friston et al., 2016).

Since the introduction of the VL scheme, other algorithms and software tools for variational Bayesian inference have been introduced that serve a similar role. The Variational Message Passing (VMP) algorithm pre-dates VL (Winn et al., 2005) and is now used in the Microsoft infer.NET programming language and the ForneyLab Julia package (Cox et al., 2019). Like VL, this is a deterministic algorithm, which inverts models that can be expressed as *Bayesian networks* or *Forney Factor Graphs* (with the prerequisite that nodes are conjugate to their parent). The distributed nature of the VMP algorithm has also enabled its use as a model for how inference is performed in biological neural networks (Parr et al., 2019). Another prominent algorithm is automatic differentiation variational inference (ADVI) (Kucukelbir et al., 2017), which is implemented in multiple probabilistic programming frameworks including Stan (Kucukelbir et al., 2015), Turing.jl for Julia (Ge et al., 2018), PyMC3 for Python and Tensorflow Probability (TFP). Like VL, ADVI optimizes the free energy, however this is performed without Laplace approximations of the free energy functional. Instead, the log joint in Eq 10 is evaluated automatically from a given graphical model, and the expected value is approximated using Monte Carlo integration methods, when the free energy is evaluated. Having defined the free energy functional, the next step is to maximize it, and VL and ADVI differ in how they do this. In VL, the gradient of the free energy under the Laplace approximation is given by Eq 43, which depends on first computing the gradient of the function to optimized, $g(\boldsymbol{\theta})$, using numerical methods (finite differences). ADVI estimates the gradient of the free energy using Automatic differentiation (AD), which involves reducing the free energy into a graph of elemental math operations and then repeatedly applying the chain rule (Griewank, 1989, Iri, 1984). These gradients are then supplied to a stochastic gradient descent algorithm. An interesting future direction would be to compare the performance of the VL scheme against VMP and ADVI for the kind of models typically applied in neuroimaging. We would predict that VL and VMP would be significantly faster because they eschew sampling, however ADVI would offer the most accurate posteriors in situations where the Laplace approximation is violated.

The code accompanying this paper illustrates applications of the VL scheme with a variety of models. Readers interested in learning more about the applications of VL may wish to proceed to recent tutorials on modelling neuroimaging data using dynamic causal modelling (DCM) (Zeidman et al., 2019a) and behavioural data using active inference (Smith et al., 2022).

Code availability

MATLAB code accompanying this paper can be downloaded from https://github.com/pzeidman/vl-tutorial .

Appendix 1 – Variational Laplace pseudocode

Inputs:

Generative model $g(\boldsymbol{\mu_\beta})$

Data vector $\boldsymbol{y}$

Starting values for (hyper)parameters $\boldsymbol{\mu_\beta}$ and $\boldsymbol{\mu_\lambda}$

Prior expected values $\boldsymbol{\eta_\beta}$ and $\boldsymbol{\eta_\lambda}$

Prior precision matrices $\boldsymbol{\Pi_\beta}$ and $\boldsymbol{\Pi_\lambda}$

Precision component matrices $\boldsymbol{Q_1} \ldots \boldsymbol{Q_k}$

Outputs:

Posterior expected values $\boldsymbol{\mu_\beta}$ and $\boldsymbol{\mu_\lambda}$

Posterior covariances $\boldsymbol{\Sigma_\beta}$ and $\boldsymbol{\Sigma_\lambda}$

Data precision matrix $\boldsymbol{\Pi_y}$

Free energy F

// Initialize to a high level of regularization

v = -4

until convergence

// Compute numerical derivative of g wrt parameters

$(\mathbf{J_g})_{ij} = \partial_{\mu_j} g_i(\boldsymbol{\mu_\beta})$

/* Optionally, if the Jacobian is unstable, make small parameter updates and re-try computing the Jacobian */

// Optimize hyperparameters

until convergence

// Compute data precision $\boldsymbol{\Pi_y}$ and data precision per component $\boldsymbol{P_i}$

$\boldsymbol{P_i} = \exp(\boldsymbol{\mu_\lambda})_i \, \boldsymbol{Q_i}$

$\boldsymbol{\Pi_y} = \sum_i \boldsymbol{P_i}$

// Compute posterior covariance over parameters

$\boldsymbol{\Sigma_\beta} = \left(\boldsymbol{J_g^T \Pi_y J_g} + \boldsymbol{\Pi_\beta}\right)^{-1}$

// Compute error terms

$\boldsymbol{\epsilon_\lambda} = \boldsymbol{\mu_\lambda} - \boldsymbol{\eta_\lambda}$

$\boldsymbol{\epsilon_y} = \boldsymbol{y} - \boldsymbol{g}(\boldsymbol{\mu_\beta})$

// Compute 1st and 2nd derivatives of the expected hyperparameters wrt g

$\dot{\boldsymbol{\mu}}_{\boldsymbol{\lambda}} = \partial_{\mu_{\lambda_i}} E_{Q(\beta)}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})]$

$= \frac{1}{2} tr\left(\boldsymbol{P_i \Pi_y^{-1}}\right) - \frac{1}{2} \boldsymbol{\epsilon_y^T P_i \epsilon_y} - \partial_{\mu_{\lambda_i}} (\boldsymbol{\epsilon_\lambda})^{\mathrm{T}} \boldsymbol{\Pi_\lambda \epsilon_\lambda} - \frac{1}{2} tr\left(\boldsymbol{\Sigma_\beta J_g^T P_i J_g}\right)$

$\ddot{\boldsymbol{\mu}}_{\boldsymbol{\lambda}} = \partial_{\mu_{\lambda_i} \mu_{\lambda_i}} E_{Q(\beta)}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})]$

$= -(\boldsymbol{\Pi_\lambda})_{ii} + \frac{1}{2} tr\left(\boldsymbol{P_i \Sigma_y - P_i \Sigma_y P_i \Sigma_y}\right) - \frac{1}{2} \boldsymbol{\epsilon_y^T P_i \epsilon_y} - \frac{1}{2} tr\left(\boldsymbol{\Sigma_\beta J_g^T P_i J_g}\right)$

// Scale a relaxed regularization value (v=4) by the curvature

$t = \exp(4 - \mathrm{Re}(\ln|\ddot{\boldsymbol{\mu}}_{\boldsymbol{\lambda}}|)/n)$

// Update hyperparameters

$\boldsymbol{\mu_\lambda} = \boldsymbol{\mu_\lambda} + (\exp[t \times \ddot{\boldsymbol{\mu}}_{\boldsymbol{\lambda}}] - \boldsymbol{I}) \ddot{\boldsymbol{\mu}}_{\boldsymbol{\lambda}}^{-1} \dot{\boldsymbol{\mu}}_{\boldsymbol{\lambda}}(t)$

end

```
        // Calculate free energy
        F = … // (see Eq 33)

        // Assess performance
        if F has improved or we're early in the optimization then
                // Compute errors
                ϵ_β = μ_β − η_β
                ϵ_y = y − g(μ_β)

                /* Compute 1st and 2nd derivatives of the expected parameters wrt g
                for parameter update */
                μ̇_β = J_g^T Π_y ϵ_y − Π_β ϵ_β
                μ̈_β = −J_g^T Π_y J_g − Π_β

                // Decrease regularization
                v = v + 1/2
        else
                // Free energy has got worse
                μ_β , μ_λ ← restore previous parameter estimates

                // Increase regularization
                v = v - 2
        end

        // Scale the log-regularization v by the curvature
        t = exp(v − Re(ln|μ̈_β|)/n)

        // Update parameters
        μ_β = μ_β + (exp[t × μ̈_β] − I)μ̈_β^{-1} μ̇_β

        if the change in F is repeatedly below criterion
                // convergence
                return
        end
end
```

References

BEAL, M. J. 2003. *Variational algorithms for approximate Bayesian inference (PhD Thesis)*, University of London.

BISHOP, C. M. 1998. Latent variable models. *Learning in graphical models.* Springer.

COX, M., VAN DE LAAR, T. & DE VRIES, B. 2019. A factor graph approach to automated design of Bayesian signal processing algorithms. *International Journal of Approximate Reasoning,* 104**,** 185-204.

DAUNIZEAU, J. 2017. The variational Laplace approach to approximate Bayesian inference. *arXiv preprint arXiv:1703.02089*.

DAUNIZEAU, J., STEPHAN, K. E. & FRISTON, K. J. 2012. Stochastic dynamic causal modelling of fMRI data: should we care about neural noise? *Neuroimage,* 62**,** 464-481.

FEYNMAN, R. 1972. Statistical Mechanics. *W. A. Benjamin*.

FRISTON, K., PARR, T. & ZEIDMAN, P. 2018. Bayesian model reduction. *arXiv preprint arXiv:1805.07092*.

FRISTON, K., STEPHAN, K., LI, B. & DAUNIZEAU, J. 2010. Generalised filtering. *Mathematical Problems in Engineering,* 2010.

FRISTON, K., ZEIDMAN, P. & LITVAK, V. 2015. Empirical Bayes for DCM: A Group Inversion Scheme. *Front Syst Neurosci,* 9**,** 164.

FRISTON, K. J., HARRISON, L. & PENNY, W. 2003. Dynamic causal modelling. *Neuroimage,* 19**,** 1273-1302.

FRISTON, K. J., KAHAN, J., BISWAL, B. & RAZI, A. 2014. A DCM for resting state fMRI. *Neuroimage,* 94**,** 396-407.

FRISTON, K. J., LITVAK, V., OSWAL, A., RAZI, A., STEPHAN, K. E., VAN WIJK, B. C., ZIEGLER, G. & ZEIDMAN, P. 2016. Bayesian model reduction and empirical Bayes for group (DCM) studies. *Neuroimage,* 128**,** 413-431.

FRISTON, K. J., MATTOUT, J., TRUJILLO-BARRETO, N., ASHBURNER, J. & PENNY, W. 2007. Variational free energy and the Laplace approximation. *Neuroimage,* 34**,** 220-234.

FRISTON, K. J., PARR, T., ZEIDMAN, P., RAZI, A., FLANDIN, G., DAUNIZEAU, J., HULME, O. J., BILLIG, A. J., LITVAK, V., MORAN, R. J., PRICE, C. J. & LAMBERT, C. 2020. Dynamic causal modelling of COVID-19. *Wellcome Open Research,* 5.

FRISTON, K. J., TRUJILLO-BARRETO, N. & DAUNIZEAU, J. 2008. DEM: a variational treatment of dynamic systems. *Neuroimage,* 41**,** 849-885.

GE, H., XU, K. & GHAHRAMANI, Z. Turing: a language for flexible probabilistic inference. International conference on artificial intelligence and statistics, 2018. PMLR, 1682-1690.

GRIEWANK, A. 1989. On automatic differentiation. *Mathematical Programming: recent developments and applications,* 6**,** 83-107.

HINTON, G. & VAN CAMP, D. Keeping neural networks simple by minimizing the description length of the weights. in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory, 1993. Citeseer.

IRI, M. 1984. Simultaneous computation of functions, partial derivatives and estimates of rounding errors—Complexity and practicality—. *Japan Journal of Applied Mathematics,* 1**,** 223-252.

JAYNES, E. T. 1957. Information theory and statistical mechanics. *Physical review,* 106**,** 620.

JORDAN, M. I., GHAHRAMANI, Z., JAAKKOLA, T. S. & SAUL, L. K. 1999. An introduction to variational methods for graphical models. *Machine learning,* 37**,** 183-233.

KASS, R. E. & RAFTERY, A. E. 1995. Bayes factors. *Journal of the american statistical association,* 90**,** 773-795.

KUCUKELBIR, A., RANGANATH, R., GELMAN, A. & BLEI, D. 2015. Automatic variational inference in Stan. *Advances in neural information processing systems,* 28.

KUCUKELBIR, A., TRAN, D., RANGANATH, R., GELMAN, A. & BLEI, D. M. 2017. Automatic differentiation variational inference. *Journal of machine learning research*.

LANILLOS, P., MEO, C., PEZZATO, C., MEERA, A. A., BAIOUMY, M., OHATA, W., TSCHANTZ, A., MILLIDGE, B., WISSE, M. & BUCKLEY, C. L. 2021. Active Inference in Robotics and Artificial Agents: Survey and Challenges. *arXiv preprint arXiv:2112.01871*.

LANILLOS, P. & VAN GERVEN, M. 2021. Neuroscience-inspired perception-action in robotics: applying active inference for state estimation, control and self-perception. *arXiv preprint arXiv:2105.04261*.

MACKAY, D. J. 1995a. Free energy minimisation algorithm for decoding and cryptanalysis. *Electronics Letters,* 31**,** 446-447.

MACKAY, D. J. 1995b. Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks. *Network: computation in neural systems,* 6**,** 469-505.

MORAN, R. J., STEPHAN, K. E., SEIDENBECHER, T., PAPE, H.-C., DOLAN, R. J. & FRISTON, K. J. 2009. Dynamic causal models of steady-state responses. *Neuroimage,* 44**,** 796-811.

NEAL, R. M. & HINTON, G. E. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models.* Springer.

OZAKI, T. 1985. Non-linear time series models and dynamical systems. *Handbook of statistics.*

PARR, T., MARKOVIC, D., KIEBEL, S. J. & FRISTON, K. J. 2019. Neuronal message passing using Mean-field, Bethe, and Marginal approximations. *Scientific reports,* 9**,** 1-18.

PENNY, W. D. 2012a. Comparing dynamic causal models using AIC, BIC and free energy. *Neuroimage,* 59**,** 319-30.

PENNY, W. D. 2012b. Comparing dynamic causal models using AIC, BIC and free energy. *Neuroimage,* 59**,** 319-330.

SMITH, R., FRISTON, K. J. & WHYTE, C. J. 2022. A step-by-step tutorial on active inference and its application to empirical data. *Journal of Mathematical Psychology,* 107**,** 102632.

STEPHAN, K. E., FRISTON, K. J. & PENNY, W. D. 2005. Computing the objective function in DCM. Technical report, Wellcome Department of Imaging Neuroscience, ION, UCL.

STEPHAN, K. E., WEISKOPF, N., DRYSDALE, P. M., ROBINSON, P. A. & FRISTON, K. J. 2007. Comparing hemodynamic models with DCM. *Neuroimage,* 38**,** 387-401.

WINN, J., BISHOP, C. M. & JAAKKOLA, T. 2005. Variational message passing. *Journal of Machine Learning Research,* 6.

ZEIDMAN, P., JAFARIAN, A., CORBIN, N., SEGHIER, M. L., RAZI, A., PRICE, C. J. & FRISTON, K. J. 2019a. A guide to group effective connectivity analysis, part 1: First level analysis with DCM for fMRI. *NeuroImage,* 200**,** 174-190.

ZEIDMAN, P., JAFARIAN, A., SEGHIER, M., LITVAK, V., CAGNAN, H., PRICE, C. J. & FRISTON, K. 2019b. A guide to group effective connectivity analysis, part 2: Second level analysis with PEB. *NeuroImage,* 200**,** 12-25.

ZEIDMAN, P., KAZAN, S. M., TODD, N., WEISKOPF, N., FRISTON, K. J. & CALLAGHAN, M. F. 2019c. Optimizing data for modeling neuronal responses. *Frontiers in neuroscience,* 12**,** 986.

Supplementary: Derivation of variational updates

Zeidman, Friston, Parr, *A Primer on Variational Laplace (VL)* (2022)

These supplementary materials provide worked derivations of many of the equations in the main text. We begin with some fundamental identities which are used repeatedly through the paper – the gradients and second derivatives of the log joint distribution $\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})$.

1. Key gradients and derivatives

1.1 Gradient of the log joint w.r.t. parameters

From Eq 34, the log joint is:

$$\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda}) = \underbrace{-\frac{1}{2}\left[\ln\left|\boldsymbol{\Pi}_{\mathbf{y}}^{-\mathbf{1}}\right| + \boldsymbol{\epsilon}_{\boldsymbol{y}}^{T}\boldsymbol{\Pi}_{\mathbf{y}}\boldsymbol{\epsilon}_{\boldsymbol{y}}\right]}_{\text{Likelihood}} \underbrace{-\frac{1}{2}\left[\ln\left|\boldsymbol{\Pi}_{\boldsymbol{\beta}}^{-\mathbf{1}}\right| + \boldsymbol{\epsilon}_{\boldsymbol{\beta}}^{T}\boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}\right]}_{\text{Prior (parameters)}} \underbrace{-\frac{1}{2}\left[\ln\left|\boldsymbol{\Pi}_{\boldsymbol{\lambda}}^{-\mathbf{1}}\right| + \boldsymbol{\epsilon}_{\boldsymbol{\lambda}}^{T}\boldsymbol{\Pi}_{\boldsymbol{\lambda}}\boldsymbol{\epsilon}_{\boldsymbol{\lambda}}\right]}_{\text{Prior (hyperparameters)}} \tag{S1}$$

Dropping terms that do not depend on the parameters $\boldsymbol{\beta}$, and evaluating at the (approximate) posterior modes:

$$\nabla_{\mu_\beta} \ln P(\boldsymbol{y}, \boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\mu}_{\boldsymbol{\lambda}}) = \nabla_{\mu_\beta}\left[-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}^{T}\boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}\right] + \nabla_{\mu_\beta}\left[-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{y}}^{T}\boldsymbol{\Pi}_{\mathbf{y}}\boldsymbol{\epsilon}_{\boldsymbol{y}}\right] \tag{S2}$$

Tackling the first term in element-wise fashion:

$$\begin{aligned}
\left(\nabla_{\boldsymbol{\mu}_{\boldsymbol{\beta}}}\left[-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}^{T}\boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}\right]\right)_i &= -\frac{1}{2}\partial_{\mu_{\beta_i}}\sum_j\sum_k \epsilon_{\beta_j}\epsilon_{\beta_k}\Pi_{\beta_{jk}} \\
&= -\frac{1}{2}\sum_j\sum_k \partial_{\mu_{\beta_i}}\left(\epsilon_{\beta_j}\epsilon_{\beta_k}\Pi_{\beta_{jk}}\right) \\
&= -\frac{1}{2}\sum_j\sum_k \partial_{\mu_{\beta_i}}\left(\epsilon_{\beta_j}\right)\epsilon_{\beta_k}\Pi_{\beta_{jk}} - \frac{1}{2}\sum_j\sum_k \partial_{\mu_{\beta_i}}\left(\epsilon_{\beta_k}\right)\epsilon_{\beta_j}\Pi_{\beta_{jk}} \\
&= -\partial_{\mu_{\beta_i}}(\boldsymbol{\epsilon}_{\boldsymbol{\beta}})^{\mathrm{T}}\boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}
\end{aligned} \tag{S3}$$

$$\Rightarrow \nabla_{\boldsymbol{\mu}_{\boldsymbol{\beta}}}\left[-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}^{T}\boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}\right] = -\partial_{\mu_\beta}(\boldsymbol{\epsilon}_{\boldsymbol{\beta}})^{\mathrm{T}}\boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}} = -\boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}}$$

Similarly, for the second term:

$$\nabla_{\boldsymbol{\mu}_{\boldsymbol{\beta}}}\left[-\frac{1}{2}\boldsymbol{\epsilon}_{\boldsymbol{y}}^{T}\boldsymbol{\Pi}_{\mathbf{y}}\boldsymbol{\epsilon}_{\boldsymbol{y}}\right] = -\partial_{\mu_\beta}(\boldsymbol{\epsilon}_{\boldsymbol{y}})^{\mathrm{T}}\boldsymbol{\Pi}_{\mathbf{y}}\boldsymbol{\epsilon}_{\boldsymbol{y}} = g'(\boldsymbol{\mu}_{\boldsymbol{\beta}})\boldsymbol{\Pi}_{\mathbf{y}}\boldsymbol{\epsilon}_{\boldsymbol{y}} = \mathbf{J}_{\boldsymbol{g}}^{T}\boldsymbol{\Pi}_{\mathbf{y}}\boldsymbol{\epsilon}_{\boldsymbol{y}} \tag{S4}$$

Where $\boldsymbol{J}_{\boldsymbol{g}}$ is the Jacobian matrix of first partial derivatives, with $\left(\mathbf{J}_{\boldsymbol{g}}\right)_{ij} = \partial_{\mu_j} g_i(\boldsymbol{\mu}_{\boldsymbol{\beta}})$ for observation i and parameter j.

Putting the two terms together:

$$\nabla_{\mu_\beta} \ln P(\boldsymbol{y}, \boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\mu}_{\boldsymbol{\lambda}}) = \mathbf{J}_{\boldsymbol{g}}^{T}\boldsymbol{\Pi}_{\boldsymbol{y}}\boldsymbol{\epsilon}_{\boldsymbol{y}} - \boldsymbol{\Pi}_{\boldsymbol{\beta}}\boldsymbol{\epsilon}_{\boldsymbol{\beta}} \tag{S5}$$

1.2 Gradient of the log joint w.r.t. hyperparameters

The data precision matrix is a linear mixture of precision components $\boldsymbol{Q_1} \ldots \boldsymbol{Q_k}$, each weighted by the exponential of a hyperparameter $\boldsymbol{\lambda} = (\lambda_1 \ldots \lambda_k)$. Re-writing the log joint to include only terms that depend on $\boldsymbol{\lambda}$, the partial derivative with respect to $\boldsymbol{\lambda}$ is:

$$\nabla_{\mu_\lambda} \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}) = \nabla_{\mu_\lambda}\left[-\frac{1}{2}\boldsymbol{\epsilon_\lambda^T \Pi_\lambda \epsilon_\lambda}\right] + \nabla_{\mu_\lambda}\left[-\frac{1}{2}\ln\left|\boldsymbol{\Pi_y^{-1}}\right|\right] + \nabla_{\mu_\lambda}\left[-\frac{1}{2}\boldsymbol{\epsilon_y^T \Pi_y \epsilon_y}\right] \quad \textit{(S6)}$$

The first term is:

$$\begin{aligned}\left(\nabla_{\mu_\lambda}\left[-\frac{1}{2}\boldsymbol{\epsilon_\lambda^T \Pi_\lambda \epsilon_\lambda}\right]\right)_i &= -\frac{1}{2}\sum_j\sum_k \partial_{\mu_{\lambda_i}}\left(\epsilon_{\lambda_j}\epsilon_{\lambda_k}\Pi_{\lambda_{jk}}\right) \\ &= -\frac{1}{2}\sum_j\sum_k \partial_{\mu_{\lambda_i}}\left[\epsilon_{\lambda_j}\right]\epsilon_{\lambda_k}\Pi_{\lambda_{jk}} + \epsilon_{\lambda_j}\,\partial_{\mu_{\lambda_i}}\left[\epsilon_{\lambda_k}\right]\Pi_{\lambda_{jk}} + \epsilon_{\lambda_j}\epsilon_{\lambda_k}\partial_{\mu_{\lambda_i}}\left[\Pi_{\lambda_{jk}}\right] \\ &= -\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}}\boldsymbol{\Pi_\lambda \epsilon_\lambda}\end{aligned} \quad \textit{(S7)}$$

$$\Rightarrow \nabla_{\mu_\lambda}\left[-\frac{1}{2}\boldsymbol{\epsilon_\lambda^T \Pi_\lambda \epsilon_\lambda}\right] = -\partial_{\mu_\lambda}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}}\boldsymbol{\Pi_\lambda \epsilon_\lambda} = -\boldsymbol{\Pi_\lambda \epsilon_\lambda}$$

The second term is:

$$\begin{aligned}\left(\nabla_{\mu_\lambda}\left[-\frac{1}{2}\ln\left|\boldsymbol{\Pi_y^{-1}}\right|\right]\right)_i &= -\frac{1}{2}\partial_{\mu_{\lambda_i}}\left[\ln\left|\boldsymbol{\Pi_y^{-1}}\right|\right] \\ &= -\frac{1}{2}\partial_{\mu_{\lambda_i}}\left[-\ln\left|\boldsymbol{\Pi_y}\right|\right] \\ &= \frac{1}{2}tr\left(\boldsymbol{\Pi_y^{-1}P_i}\right) \\ &= \frac{1}{2}tr\left(\boldsymbol{P_i\Pi_y^{-1}}\right)\end{aligned} \quad \textit{(S8)}$$

Where $\boldsymbol{P_i} = \partial_{\mu_{\lambda_i}}(\boldsymbol{\Pi_y}) = \partial_{\mu_{\lambda_i}\mu_{\lambda_i}}(\boldsymbol{\Pi_y}) = \exp(\lambda_i)\boldsymbol{\Pi_i}$ and the third line used the identity $\partial(\ln|\boldsymbol{X}|) = tr(\boldsymbol{X}^{-1}\partial\boldsymbol{X})$ – see Eq 43 of (Petersen and Pedersen, 2008). Finally, the third term is:

$$\begin{aligned}&\left(\nabla_{\mu_\lambda}\left[-\frac{1}{2}\boldsymbol{\epsilon_y^T \Pi_y \epsilon_y}\right]\right)_i = -\frac{1}{2}\sum_j\sum_k \partial_{\mu_{\lambda_i}}\left(\epsilon_{y_j}\epsilon_{y_k}\Pi_{y_{jk}}\right) \\ &= -\frac{1}{2}\sum_j\sum_k \partial_{\mu_{\lambda_i}}\left(\epsilon_{y_j}\right)\epsilon_{y_k}\Pi_{y_{jk}} - \frac{1}{2}\sum_j\sum_k \epsilon_{y_j}\partial_{\mu_{\lambda_i}}\left(\epsilon_{y_k}\right)\Pi_{y_{jk}} - \frac{1}{2}\sum_j\sum_k \epsilon_{y_j}\epsilon_{y_k}\partial_{\mu_{\lambda_i}}\left(\Pi_{y_{jk}}\right) \\ &= -\frac{1}{2}\boldsymbol{\epsilon_y^T P_i \epsilon_y} \\ &= -\frac{1}{2}tr\left(\boldsymbol{P_i}\left(\boldsymbol{\epsilon_y\epsilon_y^T}\right)\right)\end{aligned} \quad \textit{(S9)}$$

Assembling these three terms:

$$\begin{aligned}\left(\nabla_{\mu_\lambda}\ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda})\right)_{\mathrm{i}} &= -\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}}\boldsymbol{\Pi_\lambda \epsilon_\lambda} + \frac{1}{2}tr\left(\partial_{\mu_{\lambda_i}}[\boldsymbol{\Pi_y}]\boldsymbol{\Pi_y^{-1}}\right) - \frac{1}{2}\boldsymbol{\epsilon_y^T}\partial_{\mu_{\lambda_i}}(\boldsymbol{\Pi_y})\boldsymbol{\epsilon_y} \\ &= -\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}}\boldsymbol{\Pi_\lambda \epsilon_\lambda} + \frac{1}{2}tr\left(\boldsymbol{P_i}\left(\boldsymbol{\Pi_y^{-1}} - \left(\boldsymbol{\epsilon_y\epsilon_y^T}\right)\right)\right)\end{aligned} \quad \textit{(S10)}$$

1.3 Second derivatives

The second partial derivatives of the log joint with respect to the parameters are arranged in the Hessian matrix $\boldsymbol{H_\beta}$ where $\left(\boldsymbol{H_\beta}\right)_{i,j} = \partial_{\mu_{\beta_i}\mu_{\beta_j}}\ln P(y, \mu_\beta, \lambda)$. This matrix is derived as follows:

$$\begin{aligned}\boldsymbol{H_\beta} = \partial_{\mu_\beta\mu_\beta} \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}) &= \partial_{\mu_\beta}\left(g'(\boldsymbol{\mu})^T \boldsymbol{\Pi_y \epsilon_y}\right) - \partial_{\mu_\beta}\left(\boldsymbol{\Pi_\beta \epsilon_\beta}\right) \\ &= \partial_{\mu_\beta}\left(g'(\boldsymbol{\mu})^T \boldsymbol{\Pi_y \epsilon_y}\right) - \boldsymbol{\Pi_\beta} \\ &= \partial_{\mu_\beta}\left(g'(\boldsymbol{\mu})\right)^T \boldsymbol{\Pi_y \epsilon_y} + g'(\boldsymbol{\mu})^T \partial_{\mu_\beta}(\boldsymbol{\Pi_y})\boldsymbol{\epsilon_y} + g'(\boldsymbol{\mu})^T \boldsymbol{\Pi_y} \partial_{\mu_\beta}(\boldsymbol{\epsilon_y}) - \boldsymbol{\Pi_\beta} \\ &\approx -\boldsymbol{J_g^T \Pi_y J_g} - \boldsymbol{\Pi_\beta}\end{aligned} \tag{S11}$$

Where $g'(\boldsymbol{\mu})$ and $g''(\boldsymbol{\mu})$ are shorthand for the first and second derivatives respectively. In SPM, terms that depend on the second derivative $g''(\boldsymbol{\mu})$ are dropped, under the assumption that the model is only weakly non-linear. The second derivatives of the log joint with respect to the hyperparameters are arranged on the leading diagonal of the Hessian matrix $\boldsymbol{H_\lambda}$:

$$\begin{aligned}(\boldsymbol{H_\lambda})_{i,i} = \partial_{\mu_{\lambda_i}\mu_{\lambda_i}} \ln P(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_{\lambda_i}}) \\ = -\partial_{\mu_{\lambda_i}}\left(\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}} \boldsymbol{\Pi_\lambda \epsilon_\lambda}\right) - \partial_{\mu_{\lambda_i}}\left(\frac{1}{2}\boldsymbol{\epsilon_y^T P_i \epsilon_y}\right) \\ = -\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}} \boldsymbol{\Pi_\lambda} \partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda}) - \frac{1}{2}\boldsymbol{\epsilon_y^T P_i \epsilon_y} \\ = -(\boldsymbol{\Pi_\lambda})_{i,i} - \frac{1}{2}\boldsymbol{\epsilon_y^T P_i \epsilon_y}\end{aligned} \tag{S12}$$

---

2. Derivation of Eq 41

Eq. 41 is the fundamental premise of mean-field variational Bayes, and the derivation is as follows. The free energy under the mean-field approximation is:

$$\begin{aligned}F[Q(\boldsymbol{\theta})] &= E_{Q(\boldsymbol{\beta})Q(\boldsymbol{\lambda})}\left[\ln \frac{P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})}{Q(\boldsymbol{\beta})Q(\boldsymbol{\lambda})}\right] \\ &= E_{Q(\boldsymbol{\beta})Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] - E_{Q(\boldsymbol{\beta})}[\ln Q(\boldsymbol{\beta})] - E_{Q(\boldsymbol{\lambda})}[\ln Q(\boldsymbol{\lambda})]\end{aligned} \tag{S13}$$

Next, we'll re-express the free energy, expanding the expected values and separating out terms relating to $Q(\boldsymbol{\beta})$, treating any other terms as constant.

$$F[Q(\boldsymbol{\theta})] = \int\int Q(\boldsymbol{\beta})Q(\boldsymbol{\lambda})[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda}) - \ln Q(\boldsymbol{\beta}) - \ln Q(\boldsymbol{\lambda})]d\boldsymbol{\beta}d\boldsymbol{\lambda} \tag{S14}$$

$$= \int_{\boldsymbol{\beta}} \left[Q(\boldsymbol{\beta}) E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] - Q(\boldsymbol{\beta}) \ln Q(\boldsymbol{\beta}) \right] d\boldsymbol{\beta} + c$$

For convenience we will label the integral within the free energy $I[Q(\boldsymbol{\beta})]$ and its integrand $\mathcal{L}[Q(\boldsymbol{\beta})]$:

$$I[Q(\boldsymbol{\beta})] = \int_{\boldsymbol{\beta}} \underbrace{Q(\boldsymbol{\beta}) E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] - Q(\boldsymbol{\beta}) \ln Q(\boldsymbol{\beta})}_{\mathcal{L}[Q(\boldsymbol{\beta})]} d\boldsymbol{\beta} \quad \textbf{(S15)}$$

The functional derivative of the free energy with respect to the parameters is therefore:

$$\delta_{Q(\boldsymbol{\beta})} F[Q(\boldsymbol{\beta})] = \delta_{Q(\boldsymbol{\beta})} \left[I[Q(\boldsymbol{\beta})] \right] \quad \textbf{(S16)}$$

This functional derivative can be simplified using the Euler-Lagrange equation, which states that for a functional $\mathcal{L}$ of a function $f(x)$, with the integral $I[f] = \int \mathcal{L}(f) dx$, we can set to zero and replace the functional derivative with a partial derivative of the inner functional $\mathcal{L}$:

$$\frac{\delta I}{\delta y} = 0 \Rightarrow \frac{\partial \mathcal{L}}{\partial f} = 0 \quad \textbf{(S17)}$$

Applying this to the integral above:

$$\begin{aligned} \delta_{Q(\boldsymbol{\beta})} I[Q(\boldsymbol{\beta})] = 0 &\Rightarrow \partial_{Q(\boldsymbol{\beta})} \left[Q(\boldsymbol{\beta}) E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] - Q(\boldsymbol{\beta}) \ln Q(\boldsymbol{\beta}) \right] = 0 \\ &\Rightarrow Q(\boldsymbol{\beta}) \underbrace{\partial_{Q(\boldsymbol{\beta})} E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})]}_{0} + \underbrace{\partial_{Q(\boldsymbol{\beta})} Q(\boldsymbol{\beta})}_{1} E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] \\ &- (1 + \ln Q(\boldsymbol{\beta})) = 0 \\ &\Rightarrow E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] - \ln Q(\boldsymbol{\beta}) - 1 = 0 \end{aligned} \quad \textbf{(S18)}$$

Therefore, up to a constant, the variation of the free energy with respect to $Q(\boldsymbol{\beta})$ is (Eq 40):

$$\delta_{Q(\boldsymbol{\beta})} F[Q(\boldsymbol{\beta})] = -\ln Q(\boldsymbol{\beta}) + E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})]$$

And by re-arranging and taking the exponential, the posteriors that maximize the free energy are (Eq 41):

$$Q(\boldsymbol{\beta}) \propto \exp\left(E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] \right)$$

A similar expression can be derived for $\delta_{Q(\boldsymbol{\lambda})} F$ and $Q(\boldsymbol{\lambda})$.

---

3. Derivation of Eq 43

The gradient ascent update equations are derived as follows. From Eq 41:

$$\begin{aligned} \ln Q(\boldsymbol{\beta}) \propto E_{Q(\boldsymbol{\lambda})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] &= \ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\mu}_{\boldsymbol{\lambda}}) + \frac{1}{2} tr\left(\boldsymbol{\Sigma}_{\boldsymbol{\lambda}} \, \partial_{\mu_{\lambda} \mu_{\lambda}} \ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\mu}_{\boldsymbol{\lambda}}) \right) \\ \ln Q(\boldsymbol{\lambda}) \propto E_{Q(\boldsymbol{\beta})}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\lambda})] &= \ln P\left(\boldsymbol{y}, \boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\lambda} \right) + \frac{1}{2} tr\left(\boldsymbol{\Sigma}_{\boldsymbol{\beta}} \partial_{\mu_{\beta} \mu_{\beta}} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\lambda} \right) \right) \end{aligned} \quad \textbf{(S19)}$$

The algorithm alternates between updating the parameters to maximize $\ln Q(\boldsymbol{\beta})$ and the hyperparameters to maximize $\ln Q(\boldsymbol{\lambda})$. The update for each iteration is:

$$\begin{aligned}\boldsymbol{\Delta\mu_\beta} &= \nabla_{\mu_\beta} E_{Q(\lambda)}\left[\ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\lambda}\right)\right] \\ &= \nabla_{\mu_\beta} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}\right) + \nabla_{\mu_\beta} \frac{1}{2} tr\left(\boldsymbol{\Sigma_\lambda}\, \partial_{\mu_\lambda \mu_\lambda} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}\right)\right) \\ &= \nabla_{\mu_\beta} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}\right) + \frac{1}{2} \nabla_{\mu_\beta} tr(\boldsymbol{\Sigma_\lambda}\, \boldsymbol{H_\lambda})\end{aligned}$$

$$\nabla_{\mu_\beta} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\mu_\lambda}\right) = \mathbf{J}_{\boldsymbol{g}}^{T} \boldsymbol{\Pi_y} \boldsymbol{\epsilon_y} - \boldsymbol{\Pi_\beta} \boldsymbol{\epsilon_\beta}$$

$$\begin{aligned}\left(\nabla_{\mu_\beta} tr(\boldsymbol{\Sigma_\lambda}\, \boldsymbol{H_\lambda})\right)_i &= \partial_{\mu_{\beta_i}} tr(\boldsymbol{\Sigma_\lambda}\, \boldsymbol{H_\lambda}) \\ &= \partial_{\mu_{\beta_i}} \sum_{j=1}^{h} (\boldsymbol{\Sigma_\lambda})_{jj} \left[-(\boldsymbol{\Pi_\lambda})_{j,j} - \left[\frac{1}{2} \boldsymbol{\epsilon_y^T P}_j \boldsymbol{\epsilon_y}\right]\right] \\ &= -\partial_{\mu_{\beta_i}} \sum_{j=1}^{h} (\boldsymbol{\Sigma_\lambda})_{jj} (\boldsymbol{\Pi_\lambda})_{j,j} - \frac{1}{2} \partial_{\mu_{\beta_i}} \sum_{j=1}^{h} (\boldsymbol{\Sigma_\lambda})_{jj} \left[\boldsymbol{\epsilon_y^T P}_j \boldsymbol{\epsilon_y}\right] \\ &= -\frac{1}{2} \sum_{j=1}^{h} \partial_{\mu_{\beta_i}} \left[(\boldsymbol{\Sigma_\lambda})_{jj} \left[\boldsymbol{\epsilon_y^T P}_j \boldsymbol{\epsilon_y}\right]\right] \\ &= -\frac{1}{2} \sum_{j=1}^{h} (\boldsymbol{\Sigma_\lambda})_{jj}\, \partial_{\mu_{\beta_i}} \left[\boldsymbol{\epsilon_y^T P}_j \boldsymbol{\epsilon_y}\right] \\ &= -\frac{1}{2} \sum_{j=1}^{h} (\boldsymbol{\Sigma_\lambda})_{jj} \left[\partial_{\mu_{\beta_i}} \left[\boldsymbol{\epsilon_y^T}\right] \boldsymbol{P}_j \boldsymbol{\epsilon_y} + \boldsymbol{\epsilon_y^T P_j}\, \partial_{\mu_{\beta_i}} \left[\boldsymbol{\epsilon_y}\right]\right] \\ &= \sum_{j=1}^{h} (\boldsymbol{\Sigma_\lambda})_{jj}\, \partial_{\mu_{\beta_i}} \left[g(\boldsymbol{\mu_\beta})\right] \boldsymbol{P}_j \boldsymbol{\epsilon_y}\end{aligned} \qquad \textit{(S20)}$$

$$\Rightarrow \boldsymbol{\Delta\mu_\beta} = \mathbf{J}_{\boldsymbol{g}}^{T} \boldsymbol{\Pi_y} \boldsymbol{\epsilon_y} - \boldsymbol{\Pi_\beta} \boldsymbol{\epsilon_\beta} + \sum_{j=1}^{h} (\boldsymbol{\Sigma_\lambda})_{jj} \boldsymbol{J_g^T P}_j \boldsymbol{\epsilon_y}$$

Similarly, the update for the hyperparameters is:

$$\begin{aligned}(\Delta\mu_\lambda)_i &= \partial_{\mu_{\lambda_i}} E_{Q(\beta)}[\ln P(\boldsymbol{y}, \boldsymbol{\beta}, \boldsymbol{\mu_\lambda})] \\ &= \partial_{\mu_{\lambda_i}} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\lambda}\right) + \partial_{\mu_{\lambda_i}} \frac{1}{2} tr\left(\boldsymbol{\Sigma_\beta} \partial_{\mu_\beta \mu_\beta} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\lambda}\right)\right) \\ &= \partial_{\mu_{\lambda_i}} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\lambda}\right) + \partial_{\mu_{\lambda_i}} \frac{1}{2} tr\left(\boldsymbol{\Sigma_\beta H_\beta}\right) \\ &= \partial_{\mu_{\lambda_i}} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\lambda}\right) + \partial_{\mu_{\lambda_i}} \frac{1}{2} tr\left(-\boldsymbol{\Sigma_\beta} \mathbf{J}_{\boldsymbol{g}}^{T} \boldsymbol{\Pi_y} \boldsymbol{J_g} - \boldsymbol{\Sigma_\beta \Pi_\beta}\right) \\ &= \partial_{\mu_{\lambda_i}} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\lambda}\right) - \frac{1}{2} tr\left(\partial_{\mu_{\lambda_i}} \left[\boldsymbol{\Sigma_\beta} \mathbf{J}_{\boldsymbol{g}}^{T} \boldsymbol{\Pi_y} \boldsymbol{J_g}\right]\right) \\ &= \partial_{\mu_{\lambda_i}} \ln P\left(\boldsymbol{y}, \boldsymbol{\mu_\beta}, \boldsymbol{\lambda}\right) - \frac{1}{2} tr\left(\boldsymbol{\Sigma_\beta} \mathbf{J}_{\boldsymbol{g}}^{T} \boldsymbol{P_i J_g}\right) \\ &= -\partial_{\mu_{\lambda_i}} (\boldsymbol{\epsilon_\lambda})^{\mathrm{T}} \boldsymbol{\Pi_\lambda} \boldsymbol{\epsilon_\lambda} + \frac{1}{2} tr\left(\boldsymbol{P_i} \boldsymbol{\Pi}_{\boldsymbol{y}}^{-1}\right) - \frac{1}{2} \boldsymbol{\epsilon_y^T P_i} \boldsymbol{\epsilon_y} - \frac{1}{2} tr\left(\boldsymbol{\Sigma_\beta} \mathbf{J}_{\boldsymbol{g}}^{T} \boldsymbol{P_i J_g}\right)\end{aligned} \qquad \textit{(S21)}$$

4. Derivation of Eq 44

The expression required to compute the posterior covariance over parameters is:

$$\begin{aligned}\partial_{\mu_\beta\mu_\beta}E_{Q(\lambda)}\left[\ln P\left(\boldsymbol{y},\boldsymbol{\mu_\beta},\boldsymbol{\lambda}\right)\right] &= \partial_{\mu_\beta}\left[\mathbf{J}_{\boldsymbol{g}}^{T}\boldsymbol{\Pi_y}\boldsymbol{\epsilon_y}\right] - \partial_{\mu_\beta}\left[\boldsymbol{\Pi_\beta}\boldsymbol{\epsilon_\beta}\right] + \partial_{\mu_\beta}\left[\sum_{j=1}^{h}(\boldsymbol{\Sigma_\lambda})_{jj}\boldsymbol{J_g^T}\boldsymbol{P}_j\boldsymbol{\epsilon_y}\right] \\ &= \partial_{\mu_\beta}\left[\boldsymbol{J_g^T}\right]\boldsymbol{\Pi_y}\boldsymbol{\epsilon_y} + \mathbf{J}_{\boldsymbol{g}}^{T}\boldsymbol{\Pi_y}\partial_{\mu_\beta}\left[\boldsymbol{\epsilon_y}\right] - \boldsymbol{\Pi_\beta} + \sum_{j=1}^{h}\partial_{\mu_\beta}\left[(\boldsymbol{\Sigma_\lambda})_{jj}\boldsymbol{J_g^T}\boldsymbol{P}_j\boldsymbol{\epsilon_y}\right] \\ &\approx -\mathbf{J}_{\boldsymbol{g}}^{T}\boldsymbol{\Pi_y}\boldsymbol{J_g} - \boldsymbol{\Pi_\beta} - \sum_{j=1}^{h}(\boldsymbol{\Sigma_\lambda})_{jj}\boldsymbol{J_g^T}\boldsymbol{P}_j\boldsymbol{J_g} \\ &\approx -\mathbf{J}_{\boldsymbol{g}}^{T}\boldsymbol{\Pi_y}\boldsymbol{J_g} - \boldsymbol{\Pi_\beta}\end{aligned} \tag{S22}$$

For the hyperparameter this is a little more involved:

$$\begin{aligned}&\partial_{\mu_{\lambda_i}\mu_{\lambda_i}}E_{Q(\beta)}[\ln P(\boldsymbol{y},\boldsymbol{\beta},\boldsymbol{\mu_\lambda})] \\ &= -\partial_{\mu_{\lambda_i}}\left[\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}}\boldsymbol{\Pi_\lambda}\boldsymbol{\epsilon_\lambda}\right] + \frac{1}{2}\partial_{\mu_{\lambda_i}}\left[tr\left(\boldsymbol{P_i}\boldsymbol{\Pi_y^{-1}}\right)\right] - \frac{1}{2}\partial_{\mu_{\lambda_i}}\left[\boldsymbol{\epsilon_y^T}\boldsymbol{P_i}\boldsymbol{\epsilon_y}\right] \\ &-\frac{1}{2}\partial_{\mu_{\lambda_i}}tr\left(\boldsymbol{\Sigma_\beta}\boldsymbol{J_g^T}\boldsymbol{P_i}\boldsymbol{J_g}\right)\end{aligned} \tag{S23}$$

Tackling each of the four terms in Eq S23, first:

$$-\partial_{\mu_{\lambda_i}}\left[\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}}\boldsymbol{\Pi_\lambda}\boldsymbol{\epsilon_\lambda}\right] = -\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda})^{\mathrm{T}}\boldsymbol{\Pi_\lambda}\partial_{\mu_{\lambda_i}}(\boldsymbol{\epsilon_\lambda}) = -(\boldsymbol{\Pi_\lambda})_{ii} \tag{S24}$$

Second:

$$\begin{aligned}&\frac{1}{2}\partial_{\mu_{\lambda_i}}\left[tr\left(\boldsymbol{P_i}\boldsymbol{\Sigma_y}\right)\right] = \frac{1}{2}tr\left(\partial_{\mu_{\lambda_i}}\left[\boldsymbol{P_i}\boldsymbol{\Sigma_y}\right]\right) \\ &= \frac{1}{2}tr\left(\boldsymbol{P_i}\boldsymbol{\Sigma_y} + \boldsymbol{P_i}\,\partial_{\mu_{\lambda_i}}\left[\boldsymbol{\Sigma_y}\right]\right) \\ &= \frac{1}{2}tr\left(\boldsymbol{P_i}\boldsymbol{\Sigma_y} + \boldsymbol{P_i}\left(-\boldsymbol{\Sigma_y}\boldsymbol{P_i}\boldsymbol{\Sigma_y}\right)\right) \\ &= \frac{1}{2}tr\left(\boldsymbol{P_i}\boldsymbol{\Sigma_y} - \boldsymbol{P_i}\boldsymbol{\Sigma_y}\boldsymbol{P_i}\boldsymbol{\Sigma_y}\right)\end{aligned} \tag{S25}$$

Where $\boldsymbol{\Sigma_y} = \boldsymbol{\Pi_y^{-1}}$. Third:

$$-\frac{1}{2}\partial_{\mu_{\lambda_i}}\left[\boldsymbol{\epsilon_y^T}\boldsymbol{P_i}\boldsymbol{\epsilon_y}\right] = -\frac{1}{2}\boldsymbol{\epsilon_y^T}\boldsymbol{P_i}\boldsymbol{\epsilon_y} \tag{S26}$$

And fourth:

$$-\frac{1}{2}\partial_{\mu_{\lambda_i}}tr\left(\boldsymbol{\Sigma_\beta}\boldsymbol{J_g^T}\boldsymbol{P_i}\boldsymbol{J_g}\right) = -\frac{1}{2}tr\left(\partial_{\mu_{\lambda_i}}\left[\boldsymbol{\Sigma_\beta}\boldsymbol{J_g^T}\boldsymbol{P_i}\boldsymbol{J_g}\right]\right) = -\frac{1}{2}tr\left(\boldsymbol{\Sigma_\beta}\boldsymbol{J_g^T}\boldsymbol{P_i}\boldsymbol{J_g}\right) \tag{S27}$$

Thus:

$$\partial_{\mu_{\lambda_i}\mu_{\lambda_i}} E_{Q(\beta)}[\ln P(\boldsymbol{y},\boldsymbol{\beta},\boldsymbol{\lambda})] = -(\boldsymbol{\Pi}_{\boldsymbol{\lambda}})_{ii} + \frac{1}{2}tr\big(\boldsymbol{P_i\Sigma_y} - \boldsymbol{P_i\Sigma_y P_i\Sigma_y}\big) - \frac{1}{2}\boldsymbol{\epsilon_y^T P_i \epsilon_y} - \frac{1}{2}tr\big(\boldsymbol{\Sigma_\beta J_g^T P_i J_g}\big) \quad \textit{(S28)}$$

---

5. Derivation of Eq 46

This is an application of the local linearization scheme of Ozaki (1985), for approximating the integration of a continuous function via discrete steps. For one or more model parameters $\mu(t)$ at time t we have the gradient ascent:

$$\dot{\mu}(t) = f\big(\mu(t)\big) \quad \textit{(S29)}$$

In the context of the VL scheme, the function on the right-hand side is $f\left(\mu_\beta(t)\right) = \nabla_{\mu_\beta}\big[I_\beta\big(\mu_\beta[t]\big)\big]$ for updating the parameters and $f\big(\mu_\lambda(t)\big) = \nabla_{\mu_\lambda}[I_\lambda(\mu_\lambda[t])]$ for updating the hyperparameters**.** Taking the time derivative again, we obtain:

$$\ddot{\mu}(t) = J\big[f\big(\mu(t)\big)\big]\dot{\mu}(t)$$
$$J\big(\mu(t)\big) = \partial_{\mu(t)} f\big(\mu(t)\big) \quad \textit{(S30)}$$

If we assume that $f\big(\mu(t)\big)$ is linear over a short interval $t \in [0, \Delta t]$, we can replace J with the Jacobian at the origin of the dynamical system, $J_0 = \partial_{\mu(0)} f\big(\mu(0)\big)$. Thus:

$$\ddot{\mu}(t) = J_0\, \dot{\mu}(t) \quad \textit{(S31)}$$

Integrating over the short interval $t \in [0, \tau]$, where $\tau \in [0, \Delta t]$, gives:

$$\dot{\mu}(\tau) = \dot{\mu}(0)\exp(J_0\tau) \quad \textit{(S32)}$$

Integrating again, this time over $\tau \in [0, \Delta t]$, gives:

$$\mu(\Delta t) = \mu(0) + J_0^{-1}(\exp[J_0\Delta t] - I)f\big(\mu(0)\big) \quad \textit{(S33)}$$

---

7. Derivation of Eq 59

The log Bayes factor for model m_1 relative to m_2 is converted to a posterior probability in favour of m_1, under equal priors for both models, via Bayes rule:

$$P(m_1|y) = \frac{P(y|m_1)P(m)}{P(y)} \quad \textit{(S34)}$$

$$= \frac{1}{1 + B_2}$$
$$= \frac{1}{1 + \exp[\ln B_2]}$$
$$= \frac{1}{1 + \exp[-\ln B_1]}$$

Where $P(m) = P(m_1) = P(m_2)$ and B_x is the Bayes factor in favour of model x. We have written this in such a way to demonstrate that the posterior probability a softmax function of the log Bayes factor.

8. Supplementary references

OZAKI, T. 1985. Non-linear time series models and dynamical systems. *Handbook of statistics.*
PETERSEN, K. B. & PEDERSEN, M. S. 2008. The matrix cookbook. *Technical University of Denmark,* 7**,** 510.