A general method for Bayesian time series modelling

Description of the method and the R software ctsmod

Coryn A.L. Bailer-Jones
Max Planck Institute for Astronomy, Heidelberg (calj@mpia.de)
http://www.mpia.de/homes/calj/ctsmod.html
24 October 2015
Code version v23 using R-3.1.2

Abstract

I introduce a general, Bayesian method for modelling univariate time series data assumed to be drawn from a continuous, stochastic process. The method accommodates arbitrary temporal sampling, and takes into account measurement uncertainties for arbitrary error models (not just Gaussian) on both the time and signal variables. Any model for the deterministic component of the variation of the signal with time is supported, as is any model of the stochastic component on the signal and time variables. Models illustrated here are constant, linear, sigmoidal, sinusoidal and quasi-sinusoidal models for the signal mean (and combinations thereof) combined with a Gaussian stochastic component, as well as purely stochastic models, the Wiener process and Ornstein–Uhlenbeck process. The posterior probability distribution over model parameters is determined via MCMC (Metropolis, parallel tempering or emcee). Models can be compared using the evidence or the "cross-validation likelihood", in which the posterior-averaged likelihood for different partitions of the data are summed. In principle this is more robust to changes in the prior than is the evidence (the prior-averaged likelihood).

Much of the description of the mathematical method in this technical note is taken from the journal article (Bailer-Jones 2012), but is repeated here so that this technical note can stand alone. The journal article describes the application to a simulated data set and to brown dwarf light curves, which is not covered here. Conversely, this technical note describes the various time series models as well as subsequent developments such as additional models and MCMC methods. This note also describes the implementation of the method into software using R, some of its internal workings, and how to use it. The code, `ctsmod`, is designed for use under Linux (including Mac OS X), but not Windows (as it relies on forking for parallel processing). The software is available from the above web site.

Contents

1 Introduction

When confronted with a univariate time series, we are often interested in answering one or more of three questions. Which model best describes the data? What values of the parameters of this model best explain the data? What range of values does the model predict for the signal at some arbitrary time? These are questions of inference from data, and can be summarized as model comparison, parameter estimation and prediction, respectively.

Probabilistic modelling provides a self-consistent and logical framework for answering these questions. In this article I introduce a general method for time series model comparison and parameter estimation. The principle is straight forward. The time series data are a series of measurements of the signal at various times ("events"), with measurement uncertainties generally in both signal and time. Working in the time domain, we write down a parametrized model for the variation of the signal as a function of time. This could be a deterministic function or a stochastic model or, more generally, a combination of the two. An example of a combined model is a sinusoidal variation of the mean of the signal on top of which is a Gaussian stochastic variation in the signal itself, which is *not* measurement noise. A purely stochastic model is one in which the expected signal evolves according to a random distribution, e.g. a random walk. There may also be an intrinsic stochastic variation in the times of the events, which again is unrelated to measurement errors. Given these generative models, we then calculate the likelihood distribution of the data for different values of the model parameters. Rather than identifying just the single best fitting parameters, I use a Monte Carlo method to sample the posterior probability density function (PDF) over the model parameters. In addition to providing uncertainties on the inferred parameters, this also used to determine a single number for the goodness-of-fit of the overall model, such as the marginal likelihood (evidence), or the cross-validation likelihood (defined here). In this way we can identify the best overall model from a set, something which frequentist hypothesis testing can be notoriously bad at (e.g. Berger & Sellke 1987, Kass & Raftery 1996, Jaynes 2003, Christensen 2005, Bailer-Jones 2009).

There of course exist numerous time series analysis methods which attempt to answer one or more of the questions posed, so the reader may wonder why we need another one. For example, if we focus on periodic (Fourier) models, then we can calculate the power spectrum or periodogram in order to identify the most significant periods and to estimate the amplitudes of the components. Or if we work in the time domain, we can do least squares fitting of a parametrized model (e.g. Chatfield 1996, Brockwell & Davis 2002). However, many of these methods can only answer one of the posed questions, are limited to a restricted set of models or specific types of problems, cannot deal with uncertainties in the signal and/or time, are limited to equally-spaced data, do not provide uncertainty estimates on the parameters, or make other restrictive assumptions. The method introduced in the present work is quite general, and firmly embedded in a probabilistic approach to data modelling. This makes it powerful, but at the price of considerably higher computational speed. However, in many applications this is a price we should be willing to pay for hard-won data, and should often be preferred to ad hoc, suboptimal recipes.

This article describes this method. An example application can be found in the journal article on this method, Bailer-Jones (2012). The method developed here is related to the `artmod` method introduced in Bailer-Jones (2011; hereafter CBJ11), which is a model for time-of-arrival time series. The present method extends this to model time series with noisy signal values at each measured time. `artmod` can be applied to data sets in which there is a signal varying with time, but only if that signal can be interpreted as a normalized probability density (such as the probability of an asteroid impact) rather than noisy measurements associated with each event.

The notation is summarized in Table 1.

Table 1: Primary notation

s_j	measured time of j^{th} event
σ_{s_j}	standard deviation in s_j
t_j	(unknown) true time of the j^{th} event
y_j	measured signal of j^{th} event
σ_{y_j}	standard deviation in y_j
z_j	(unknown) true signal of the j^{th} event
D_j	$= (s_j, y_j)$ measurements for the j^{th} event
σ_j	$= (\sigma_{s_j}, \sigma_{y_j})$ estimated uncertainties in D_j
D	$= \{D_j\}$ set of measurements for J events
σ	$= \{\sigma_j\}$ estimated uncertainties in D
D_k	set of measurements for events in partition k
D_{-k}	set of measurements for all events not in in partition k
M	time series model
θ	$= (\theta_1, \theta_2, \theta_3)$, parameters of the time series model
$\eta(t; \theta_1)$	deterministic model of the expected true signal (TSMod1)
log	logarithm base 10
$\mathcal{N}(x; \mu, V)$	Gaussian in x with mean μ and variance V

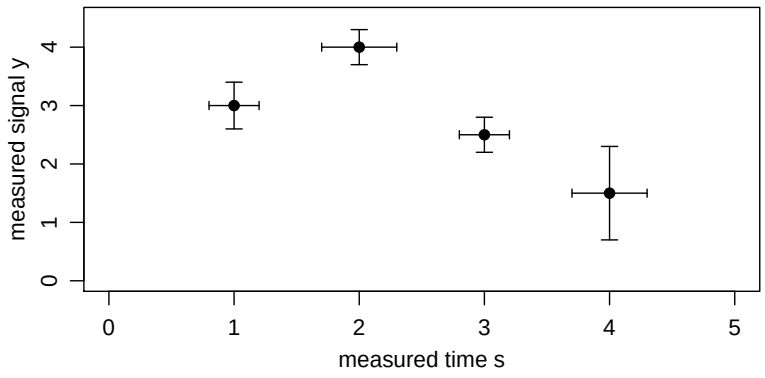

Figure 1: Example of a measured data set with four events

2 The time series method

2.1 Data and model definition

Let t and z denote the time and signal respectively. We have a set of J events. For each event j, our measurement of the time of the event, t_j, is s_j with a standard deviation (estimated measurement uncertainty) σ_{s_j}, and our measurement of the signal of the event, z_j, is y_j with a standard deviation (estimated measurement uncertainty) σ_{y_j} (see Figure 1). That is, t_j and z_j are the true, unknown values, not the measurements. Define $D_j = (s_j, y_j)$ and $\sigma_j = (\sigma_{s_j}, \sigma_{y_j})$. The *measurement model* (or noise model) describes the probability of observing the measured values for a single event given the true values and the estimated uncertainties, i.e. it gives $P(D_j|t_j, z_j, \sigma_j)$. Note that we consider σ_j as fixed parameters of the measurement model, and the conditioning on the measurement model is implicit (as we do not want to compare measurement models in

this work).

M is a stochastic *time series model* with parameters θ. It specifies $P(t_j, z_j|\theta, M)$, the probability of observing an event at time t_j with signal z_j.

The goal is (1) to compare the posterior probability of different models M, and (2) to determine the posterior PDF over the model parameters for a given M. After describing the measurement and time series models in the next two subsections, I will then show how to combine them in order to calculate the likelihood.

2.2 Measurement model

If t and z have no bounds, or can be approximated as such, and the known measurement uncertainties are standard deviations, then an appropriate choice for the measurement model is a 2D Gaussian in the variables (s_j, y_j) for event j. If we assume no covariance between the variables then this reduces to product of two 1D Gaussians

$$P(D_j|t_j, z_j, \sigma_j) = \frac{1}{\sqrt{2\pi}\sigma_{s_j}} e^{-(s_j-t_j)^2/2\sigma_{s_j}^2} \; \frac{1}{\sqrt{2\pi}\sigma_{y_j}} e^{-(y_j-z_j)^2/2\sigma_{y_j}^2} \; . \tag{1}$$

(The two terms are normalized with respect to s_j and y_j respectively.) If we had additional information about the measurement, e.g. asymmetric error bars or strictly positive signals, then we should adopt a more appropriate distribution. If the uncertainty estimates we had were not standard deviations, then we may also want to use a different measurement model.

2.3 Time series model

Without loss of generality, the time series model can be written as the product of two stochastic components

$$P(t_j, z_j|\theta, M) \; = \; P(z_j|t_j, \theta, M) P(t_j|\theta, M) \tag{2}$$

which I will refer to as the signal and time components respectively. For many processes it is appropriate to express the signal component using two independent subcomponents: the stochastic model itself and a deterministic function which defines the time-dependence of its mean. This stochastic subcomponent describes the intrinsic variability of the true signal of the physical process at a given time, with the PDF $P(z_j|t_j, \theta', M)$. I refer to this as TSMod2. An example is a Gaussian

$$P(z_j|t_j, \theta', M) \; = \; \frac{1}{\sqrt{2\pi}\omega} e^{-(z_j-\eta[t_j])^2/2\omega^2} \tag{3}$$

where $\theta' = (\eta, \omega)$ are the parameters of the distribution: $\eta[t_j]$ is the expected true signal at true time t_j; ω is a parameter which reflects the degree of stochasticity in the process. This is illustrated schematically for a single point in Figure 2. The Gaussian is just an example, and would be inappropriate if z were a non-negative quantity.

The relationship between the expected true signal and the true time is given by a deterministic function, $\eta(t; \theta_1)$, where θ_1 denotes another set of parameters. I refer to this deterministic subcomponent as TSMod1. A simple example is a single frequency sinusoid

$$\eta \; = \; \frac{a}{2} \cos[2\pi(\nu t + \phi)] + b \tag{4}$$

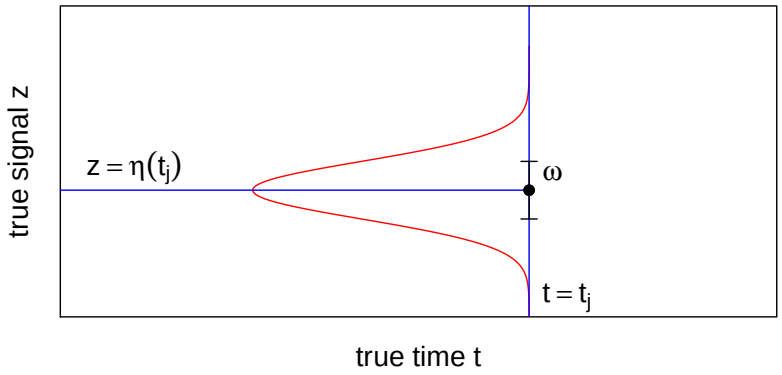

Figure 2: Conceptual representation of the stochastic nature of the signal component of the time series model, $P(z_j|t_j, \theta', M)$ (in red) of the true signal, z, at a true time t_j (here shown as a Gaussian).

which has parameters $\theta_1 = (a, \nu, \phi, b)$, the amplitude, frequency, phase, and offset. Having parametrized the signal component of the time series model in this way, it is convenient to write $\theta' = (\theta_1, \theta_2)$ in general, where $\theta_2 = \omega$ in the example of equation 3.

The second component of the time series model in equation 2 describes the intrinsic randomness in time of the events which make up the physical process. This is represented by $P(t_j|\theta_3, M)$, which I refer to this as TSMod3. In the simplest case this would be a flat distribution in t between the earliest possible and latest possible times, T_1 and T_2,

$$P(t_j|\theta_3, M) = \begin{cases} \dfrac{1}{T_2 - T_1} & \text{if } T_1 < t_j < T_2 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\theta_3 = (T_1, T_2)$ are its parameters. We may instead have a process in which we expect the probability of an event occurring to vary over time, as used in the modelling of impact cratering in Bailer-Jones (2011), for example. In the present paper I will use a uniform distribution.

This three-subcomponent approach (TSMod1,2,3) to the time series model is conceptually a little complex, so let us consider what it means. We have a physical process in which the expected value of the signal varies with time in a deterministic manner. This is given by $\eta(t; \theta_1)$, e.g. equation 4. At any given true time, the actual signal of the process can vary due to intrinsic randomness in the process. This is described by $P(z_j|t_j, \theta_1, \theta_2, M)$, an example of which is equation 3. Finally, although the mean of the process signal is considered to vary continuously in time, there may be a time varying probability that an event actually occurs (e.g. palaeontological mass extinctions). This is described by $P(t_j|\theta_3, M)$. Note that the stochasticity in the time series model has nothing to do with measurement noise. It is intrinsic to the process.

This description of the signal component as a stochastic model with a time-independent variance and a (deterministic) time-dependence for the mean we might refer to as a *partially* stochastic process. A *fully* stochastic process, in contrast, is one in which *all* the parameters of the PDF $P(z_j|t_j, \theta, M)$ have a time-dependence, so this decomposition of the signal component into TSMod1 and TSMod2 is not possible. An example is the Ornstein–Uhlenbeck process, which will be used in this work. It is described in appendix B.

The overall time series model is the combination of these three subcomponents

$$P(t_j, z_j|\theta, M) = P(z_j|t_j, \theta_1, \theta_2, M)P(t_j|\theta_3, M) \tag{6}$$

Table 2: Time series models of the three types, with parameters θ. The penultimate columns shows possible prior PDFs over these parameters, which themselves have parameters α. These prior PDFs respect the limits on θ listed in the final column. (In some cases there is more than one prior form possible.)

Name	Function	θ	prior PDF, $P(\theta;\alpha)$	$\theta_{\min}, \theta_{\max}$
TSMod1 $= \eta(t;\theta_1)$				
`FuncUniform`	b	b	$\text{Gamma}(b;\text{shape},\text{scale})$	$0, \infty$
			$\mathcal{N}(b;\text{mean},\text{sd})$	$-\infty, \infty$
`FuncLinear`	$m(t-t_0)+z_0$	m	Cauchy(m) (see section A.1)	$-\infty, \infty$
		t_0	$\mathcal{N}(t_0;\text{mean},\text{sd})$	$-\infty, \infty$
		z_0	$\text{Gamma}(z_0;\text{shape},\text{scale})$	$0, \infty$
			$\mathcal{N}(z_0;\text{mean},\text{sd})$	$-\infty, \infty$
`FuncSigmoid`	$a\left(1+e^{-(t-t_0)/\lambda}\right)^{-1}$	a	$\text{Gamma}(a;\text{shape},\text{scale})$	$0, \infty$
		λ	$\mathcal{N}(\lambda;\text{mean},\text{sd})$	$-\infty, \infty$
		t_0	$\mathcal{N}(t_0;\text{mean},\text{sd})$	$-\infty, \infty$
`FuncSigmoidZero`	$a\left(1+e^{-(t-t_0)/\lambda}\right)^{-1} - \frac{a}{2}$	as `FuncSigmoid`		
`FuncSinusoid`	$\frac{a}{2}\left(\cos[2\pi(\nu t+\phi)]+1\right)$	a	$\text{Gamma}(a;\text{shape},\text{scale})$	$0, \infty$
		ν	$\text{Gamma}(\nu;\text{shape},\text{scale})$	$0, \infty$
		ϕ	$U(\phi)$	$0, 1$
`FuncSinusoidZero`	$\frac{a}{2}\cos[2\pi(\nu t+\phi)]$	as `FuncSinusoid`		
`FuncQuasiSinusoid`	$\frac{a}{2}\left(\cos[2\pi(\nu t+\phi+\Psi[t])]+1\right)$	a	$\text{Gamma}(a;\text{shape},\text{scale})$	$0, \infty$
		ν	$\text{Gamma}(\nu;\text{shape},\text{scale})$	$0, \infty$
		ϕ	$U(\phi)$	$0, 1$
	$\Psi(t;f_x)$ $x \in (1,2,3,4)$ see section A.2	f_x	$U(\phi)$	$-1, +1$
`FuncQuasiSinusoidZero`	$\frac{a}{2}\cos[2\pi(\nu t+\phi+\Psi[t])]$	as `FuncQuasiSinusoid`		
TSMod2				
`ProbGaussian`	$\frac{1}{\sqrt{2\pi}\sigma_z}\, e^{-(z-\eta[t;\theta_1])^2/2\sigma_z^2}$	σ_z	$\text{Gamma}(\sigma_z;\text{shape},\text{scale})$	$0, \infty$
`ProbOUprocess`	see section B.2	τ	$\text{Gamma}(\tau;\text{shape},\text{scale})$	$0, \infty$
		c	$\text{Gamma}(c;\text{shape},\text{scale})$	$0, \infty$
		b	$\mathcal{N}(b;\text{mean},\text{sd})$	$-\infty, \infty$
		$\mu[z_1]$	$\mathcal{N}(\mu[z_1];\text{mean},\text{sd})$	$-\infty, \infty$
		$\sqrt{V[z_1]}$	$\text{Gamma}(\sqrt{V[z_1]};\text{shape},\text{scale})$	$0, \infty$
`ProbWienerprocess`	see section B.4	c	$\text{Gamma}(c;\text{shape},\text{scale})$	$0, \infty$
		b	$\mathcal{N}(b;\text{mean},\text{sd})$	$-\infty, \infty$
		$\mu[z_1]$	$\mathcal{N}(\mu[z_1];\text{mean},\text{sd})$	$-\infty, \infty$
		$\sqrt{V[z_1]}$	$\text{Gamma}(\sqrt{V[z_1]};\text{shape},\text{scale})$	$0, \infty$
`ProbOUprocNaive`	$\frac{1}{\sqrt{2\pi}\sigma_z}\, e^{-(z-\eta[t;\theta_2])^2/2\sigma(t;\theta_2)^2}$	τ	$\text{Gamma}(\tau;\text{shape},\text{scale})$	$0, \infty$
	see section B.1 (footnote 12) for the functional forms of $\eta(t;\theta_2)$ and $\sigma(t;\theta_2)$	c	$\text{Gamma}(c;\text{shape},\text{scale})$	$0, \infty$
		t_0	$\mathcal{N}(t_0;\text{mean},\text{sd})$	$-\infty, \infty$
		z_0	none	$-\infty, \infty$
`ProbIntOUprocNaive`	as `ProbOUprocNaive`, but with different $\eta(t;\theta_2)$ and $\sigma(t;\theta_2)$	as `ProbOUprocNaive` plus:		
		w_0	$\mathcal{N}(w_0;\text{mean},\text{sd})$	$-\infty, \infty$
TSMod3				
`ProbUniform`	$\begin{cases} \frac{1}{T_2-T_1} & \text{if } T_1 < t_j < T_2 \\ 0 & \text{otherwise} \end{cases}$	T_1	none	$-\infty, \infty$
		T_2	none	$-\infty, \infty$

Table 3: Compound models for TSMod1. The uniform component can be removed by making it a fixed parameter

Name	No. parameters
`FuncCombUniformSigmoid`	4
`FuncCombUniformSigmoidZero`	4
`FuncCombUniformSinusoid`	4
`FuncCombUniformSinusoidZero`	4
`FuncCombUniformQuasiSinusoid`	8
`FuncCombUniformQuasiSinusoidZero`	8
`FuncCombUniformSigmoidSinusoid`	7
`FuncCombUniformSigmoidZeroSinusoidZero`	7

where $\theta = (\theta_1, \theta_2, \theta_3)$. For the cases shown above, this model has seven parameters, $\theta = (a, \nu, \phi, b, \omega, T_1, T_2)$, although possibly we would fix (T_1, T_2) based on inspection of the time range of the data.

Table 2 shows some specific time series models and their parametrizations. Some clarification is necessary

- the choice of form for the prior for a given parameter is dictated in part by the support of the parameter, as discussed in section 3.3
- the `FuncLinear` is explained in more detail in section A.1
- a uniform PDF has no parameters when the range is fixed. This is the case for the phase parameter, ϕ, for `FuncSinusoid`
- in all TSMod1 models except `FuncUniform`, a is the total (peak-to-peak) signal amplitude
- in `FuncSigmoid` and `FuncSinusoid` the minimum value of the function (if evaluated over an arbitrarily long time scale) is zero, whereas the two functions of the similar name but ending in "`Zero`" are symmetric about zero. The former are better suited for signals which are positive everywhere
- "none" in the penultimate column means that these model parameters are fixed, typically derived from the data or other prior assumptions (formally the prior is a delta function)
- other (compound) models for TSMod1 are formed by summing the listed functions, e.g. `FuncCombUniformSinusoidZero` (4 parameters) or `FuncCombUniformSinusoidSigmoid` (7 parameters). A complete list is given in Table 3

It should be apparent that this method (with the partially stochastic models) can be used to model any 2D data set, not just temporal data. In particular, using the linear model (`TSMod1=FuncLinear`) it offers a full Bayesian solution to the problem of fitting a straight line to data sets with arbitrary errors in both variables (sometimes solved conventionally using "total least squares").

2.4 Likelihood

The probability of observing data D_j from time series model M with parameters θ when the uncertainties are σ_j, is $P(D_j|\sigma_j, \theta, M)$, the *event likelihood*. We can write this as

$$\begin{aligned} P(D_j|\sigma_j, \theta, M) &= \int_{t_j, z_j} P(D_j, t_j, z_j|\sigma_j, \theta, M)\ dt_j dz_j \\ &= \int_{t_j, z_j} P(D_j|t_j, z_j, \sigma_j, \theta, M) P(t_j, z_j|\sigma_j, \theta, M)\ dt_j dz_j \\ &= \int_{t_j, z_j} \underbrace{P(D_j|t_j, z_j, \sigma_j)}_{\text{Measurement model}}\ \underbrace{P(t_j, z_j|\theta, M)}_{\text{Time series model}}\ dt_j dz_j \end{aligned} \tag{7}$$

where the time series model and its parameters drop out of the first term because D_j is independent of this once conditioned on the true variables, and the measurement model (via σ_j) drops out of the the second term because it has nothing to do with the predictions of the time series model. For specific, but common, situations, this 2D integral can be approximated by a 1D integral or even a function evaluation (see section C).

If we have a set of J events for which the ages and signals have been estimated independently of one another, then the probability of observing these data $D = \{D_j\}$, the *likelihood*, is

$$P(D|\sigma, \theta, M) = \prod_j P(D_j|\sigma_j, \theta, M) \tag{8}$$

where $\sigma = \{\sigma_j\}$.

3 Model comparison

3.1 Evidence

The *evidence*, E, is obtained by marginalizing the likelihood over the parameter prior probability distribution, $P(\theta|M)$.

$$\begin{aligned} E \equiv P(D|\sigma, M) &= \int_\theta P(D, \theta|\sigma, M)\ d\theta \\ &= \int_\theta \underbrace{P(D|\sigma, \theta, M)}_{\text{likelihood}}\ \underbrace{P(\theta|M)}_{\text{prior}}\ d\theta\ . \end{aligned} \tag{9}$$

Note that the evidence is conditioned on both the measurement model (via σ) and the time series model, M. For a given set of data, we calculate this evidence for the different models we wish to compare, each parametrized by θ. The parameter prior, $P(\theta|M)$, encapsulates our prior knowledge of the plausibility of different parameters. (This is independent of σ, which is why it was removed in the above equation.) As the evidence (and the likelihood) have an uninterpretable scale, we usually examine the ratio of the evidence of a model relative to some base model (the Bayes factor).

The final step is to use Bayes' theorem to calculate the *model posterior probabilities* for each model, which

for one particular model M_0 is (omitting the explicit dependence of the evidence on σ)

$$\begin{aligned} P(M_0|D) &= \frac{P(D|M_0)P(M_0)}{P(D)} \\ &= \frac{P(D|M_0)P(M_0)}{\sum_{k=0}^{k=K} P(D|M_k)P(M_k)} \\ &= \frac{1}{1+\frac{\sum_{k=1}^{k=K} P(D|M_k)P(M_k)}{P(D|M_0)P(M_0)}} \qquad (10) \\ &= \frac{1}{1+\sum_{k=1}^{k=K} BF_{k0}R_{k0}} \, . \qquad (11) \end{aligned}$$

where $P(M_k)$ is the prior probability for model k, $BF_{k0} \equiv P(D|M_k)/P(D|M_0)$ is the Bayes factor of model M_k relative to model M_0, and $R_{k0} \equiv P(M_k)/P(M_0)$ is the ratio of prior probabilities for these models. These model posterior probabilities are usually difficult to determine, because it demands that we can specify all plausible models.

3.2 Cross validation likelihood

The evidence is often sensitive to the parameter prior PDF. For example, in a single-parameter model, if the likelihood were constant over the range $0 < \theta < 1$ but zero outside of this, then the evidence calculated using a prior uniform over $0 < \theta < 2$ would be half that calculated using a prior uniform over $0 < \theta < 1$. In a model with p such parameters the factor would be 2^p. If we had no reason to limit the prior range, then the evidence would be of limited use in this example. In cases where the parameters have a physical interpretation and/or where we have reasonable prior information, then we may be able to justify a reasonable choice for the prior. But in any case we should explore the sensitivity of the evidence to "fair" changes in the prior. A fair change is one which we have no reason not to make. For example, if there were no reason to prefer a prior which is uniform over frequency rather than period, then this would be a fair change. (See Bailer-Jones 2011 for an illustration of this on real data.) If the evidence changes enough to alter the significance of the Bayes factors when making fair changes, then the evidence is over-sensitive to the choice of prior, making it impossible to draw robust conclusions without additional information.

When the parameters do not have a physical interpretation, it will be very difficult to identify a reasonable prior PDF, and if neutral changes are a problem, then the evidence is not useful. We may then choose to use one of the many "information criteria" which have been defined. The two most popular are (e.g. Kadane & Lazar 2004)

$$\text{Akaike Information Criterion (AIC)} = -2\ln\hat{L} + 2N_\theta \qquad (12a)$$

$$\text{Bayesian Information Criterion (BIC)} = -2\ln\hat{L} + 2N_\theta J \qquad (12b)$$

where $\hat{L} = P(D|\sigma, \hat{\theta}, M)$ is the maximum likelihood, N_θ is the number of model parameters, J is the number of data points (events), and the natural logarithm is used. In both cases the second term acts a regularizer to the bare maximum likelihood. In both cases $\hat{L}$ would normally be found via a direct maximization of the likelihood function (not currently implemented in `ctsmod`).

Another criterion is based on the deviance, $\mathcal{D}(\theta) = -2\ln L + C$, where $L = P(D|\sigma, \theta, M)$ is the likelihood and C is a constant which will cancel out in any useful measures based on the deviance (Spiegelhalter et al. 2002). This is used to define the

$$\text{Deviance Information Criterion (DIC)} = 2\overline{\mathcal{D}(\theta)} - \mathcal{D}(\overline{\theta}) \qquad (13)$$

where the means are taken over samples of θ drawn from the posterior PDF, typically by MCMC.[1] The advantage of these various information criteria is that they are simple recipes and quick to calculate, but they all make (possibly unreasonable) assumptions. Extensive discussions can be found in the literature.

Another approach is a form of K-fold cross validation (CV). We (randomly) split the data set (J events) into K disjoint partitions, where $K \leq J$. Denote the data in the $k^{\rm th}$ partition as D_k and its complement as D_{-k}. The idea is to calculate the likelihood of D_k using D_{-k}, without having an additional dependence on a specific choice of model parameters. That is, we want $P(D_k|D_{-k},\sigma,M)$, which tells us how well, in model M, some of the data are predicted using the other data. Combining these likelihoods for all $k = 1 \ldots K$ gives an overall measure of the fit of the model. By marginalization

$$\begin{aligned} P(D_k|D_{-k},\sigma,M) &= \int_\theta P(D_k|D_{-k},\sigma,\theta,M)P(\theta|D_{-k},\sigma,M)d\theta \\ &= \int_\theta \underbrace{P(D_k|\sigma_k,\theta,M)}_{\text{likelihood}} \underbrace{P(\theta|D_{-k},\sigma_{-k},M)}_{\text{posterior}} d\theta \end{aligned} \tag{15}$$

where D_{-k} drops out of the first term because the model predictions are independent of these data once the parameters have been specified. (σ_{-k} and σ_k drop out of the first and second terms, respectively, also for reasons of independence.) (Cf. equation 10 of Vehtari & Lampinen 2002.) If we draw a sample $\{\theta_n\}$ of size N from the posterior $P(\theta|D_{-k},\sigma_{-k},M)$, then the Monte Carlo approximation of this integral is

$$L_k \equiv P(D_k|D_{-k},\sigma,M) \approx \frac{1}{N}\sum_{n=1}^{n=N} P(D_k|\sigma_k,\theta_n,M) \tag{16}$$

i.e. the mean of the likelihood of the data in partition k. I will call L_k the *partition likelihood*. Note that here the posterior is sampled using the data D_{-k} only.

Because L_k is the product of event likelihoods, it scales multiplicatively with the number of events in partition k. An appropriate combination of the partition likelihoods over all partitions is therefore to take their product, i.e.

$$L_{\rm CV} = \prod_{k=1}^{k=K} L_k \quad \text{or} \quad \log L_{\rm CV} = \sum_k \log L_k \tag{17}$$

which I call the *K-fold cross validation likelihood*, for $1 \leq K \leq J$. If $K > 1$ and $K < J$ then its value will depend on the choice of partitions. If $K = J$ there is one event per partition (a unique choice). This is *leave-one-out CV (LOO-CV)*, the likelihood for which I will denote with $L_{\rm LOO-CV}$. If $K = 1$, we just use all of the data to calculate both the likelihood and the posterior. I will refer to this as the *posterior-averaged likelihood*, and denote it with $L_{\rm PA}$. This is not a very correct measure of goodness-of-fit, however, because it uses all of the data both to draw the posterior samples and to calculate the likelihood. It is analogous to the evidence, which is the likelihood averaged over the prior. We can also compare it to $-\mathcal{D}(\overline{\theta})/2$, which is the (log base e) likelihood at the average (over the posterior) parameters.

The required posterior PDF is given by Bayes' theorem. It is sufficient to use the unnormalized posterior (as indeed we must, because the normalization term is the evidence), which is

$$P(\theta|D_{-k},\sigma_{-k},M) \propto P(D_{-k}|\sigma_{-k},\theta,M)P(\theta|M) \tag{18}$$

[1] I note in passing that $N_{\rm eff} = \overline{\mathcal{D}(\theta)} - \mathcal{D}(\overline{\theta})$ is considered as the effective number of parameters, so

$$\text{DIC} = \mathcal{D}(\overline{\theta}) + 2N_{\rm eff}\ , \tag{14}$$

which makes the parallel to the AIC clearer (and $-\mathcal{D}(\overline{\theta})/2$ is a log likelihood). Spiegelhalter et al. (2002) make the point that the number of free parameters in a Bayesian analysis is far from obvious, as we realize when we think of marginalizing over parameters in a hierarchical model: do we count them? This undermines the usefulness of AIC and BIC.

i.e. the product of the likelihood and the prior. $L_{\rm CV}$ therefore still depends on the choice of prior (discussed in section 3.3). However, the likelihood will often dominate the prior (unless the data are very indeterminate), in which case $L_{\rm CV}$ will be less sensitive to the prior than is the evidence.

There is a close relationship between the partition likelihood and the evidence. Whereas the evidence involves integrating the likelihood (for D) over the *prior* (equation 9), the partition likelihood involves integrating the likelihood (for D_k) over the *posterior* (for D_{-k}) (equation 15). We can further use the product rule to write the partition likelihood as

$$L_k \equiv P(D_k|D_{-k},\sigma_k,M) = \frac{P(D|\sigma,M)}{P(D_{-k}|\sigma_{-k},M)} . \tag{19}$$

The partition likelihood is equal to the ratio of the evidence calculated over all the data to the evidence calculated on the subset of the data used in the posterior sampling. As the same prior PDF enters into both terms, it will, in some vague sense, "cancel" out (although I stress that there is still a prior dependence).

It is important to realize that the model complexity is taken into account by the model comparison with the K-fold CV likelihood (and therefore the LOO-CV likelihood), just as it is with the Bayesian evidence. That is, more complex models are not penalized simply on account of having more parameters.

3.3 Parameter priors

All of the model measures mentioned – the evidence, the K-fold CV likelihood, also the DIC – are calculated by averaging the likelihood over the model parameter space. This parameter space must therefore be sampled, and this requires that we specify a prior PDF, $P(\theta|M)$, over these parameters (the parameters of the time series model).

We invariably have some information about values of the model parameters, such as bounds or plausible values. For example, standard deviations, frequencies and amplitudes cannot be negative, and a phase must lie be between 0 and 1. I therefore adopt simple distributions which respect these constraints, in particular the gamma distribution for the former case. This distribution is characterized by two parameters, shape and scale (both positive). It may be useful to recall that the mean of this distribution is shape times scale, and its variance is the mean times scale.

The components of the time series models used in this article, along with their prior distributions, are shown in Table 2.

We have to assign values for the parameters, α, of the prior PDFs. Although we rarely have sufficient knowledge to specify these precisely, we can use our knowledge of the problem and the general scale of the data to assign them. I adopt the following procedure for assigning what I call the *canonical priors* (appropriate for the data which are analysed in Bailer-Jones 2012). Some parameters are set according to the standard deviation of the signal values, $\varsigma_y = \sqrt{\frac{1}{J-1}\sum_j (y_j - \overline{y_j})^2}$, where $\overline{y_j}$ is the mean signal.

- For the Off model (parameter b), I use a Gaussian with zero mean and standard deviation 1–2 times ς_y. The exact value is determined by visual inspection of the light curve.
- For the Sin model, I use a gamma prior on the frequency, ν, with shape=1.5 and scale=0.5 for all light curves (Fig. 3). This assigns significant prior probability to a broad range of frequencies believed to be plausible based on knowledge of the problem, the temporal sampling, and the total span of the light curves. For the amplitude, a, I use a gamma prior with shape=2 and scale 1–3 times ς_y.
- For the Stoch model (parameter ω), I use gamma prior with shape=2 and scale 1–2 times ς_y.

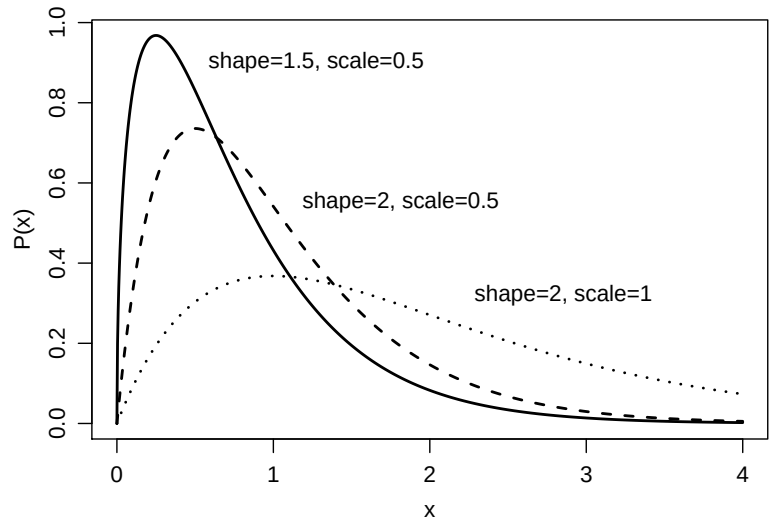

Figure 3: Three examples of the gamma PDF, used as the prior for non-negative model parameters. The solid line, with shape=1.5, scale=0.5, is used as the prior PDF over frequency in units of inverse hours.

- For the OU process, I use a gamma prior on both τ and c with shape=1.5. τ is a decay timescale, so I set its scale parameter to one quarter of the duration of the time series. The long-term variance of the OU process is $c\tau/2$. Equating this to ς_y^2, I therefore set the scale of the diffusion coefficient, c, to be $2\varsigma_y^2/\tau$. The parameters b and $\mu[z_1]$ are both assigned Gaussian priors with a standard deviation equal to ς_y. The mean of the former is set to zero, and the mean of the latter to y_1, the signal value of the first data point. The final parameter, $\sqrt{V[z_1]}$, a standard deviation, is assigned a gamma distribution with shape=1.5 and scale=ς_y.

This scheme of "data-based" priors was arrived at after some experimentation, and generally the results are robust to small changes in the priors (as demonstrated later).

4 Parameter estimation

For an arbitrary sample of parameters, obtained in any way, we can explicitly calculate the posterior PDF at each parameter and then just plot this. However, unless the sample has been well chosen to sample all the variations, it may not show the relevant structure of the PDF. In particular, it may miss the peak(s). If instead the set of parameters, $\{\theta\}$, has been drawn from the posterior PDF itself (typically by MCMC), then this gives an "ideal" sampling. In that case we could (should) instead use a density estimation algorithm to estimate the PDF directly from these samples, i.e. without using the posterior density values themselves. (Indeed, some MCMC algorithms, such as `MCMCmetrop1R{MCMCpack}` in R, only return the samples and not the function estimates.) Note that in both cases the distributions plotted will not be normalized.

When the samples have been drawn directly from the posterior, then we can easily estimate statistical quantites of the distribution. For example, the mean and standard deviation of the posterior are just the mean and

standard deviation of the samples, i.e.

$$\overline{\theta} = \frac{1}{N}\sum_n \theta_n \tag{20}$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_n (\theta_n - \overline{\theta})^2} \tag{21}$$

respectively. If the samples have instead been drawn from some other distribution $Q(\theta)$ (which also need not be normalized), then we have to reweight them in order to calculate statistical quantities, a method known as importance sampling. In this case

$$\overline{\theta} = \frac{1}{\sum_n w_n}\sum_n w_n \theta_n \tag{22}$$

where $w_n = P(\theta_n)/Q(\theta_n)$ and $P(\theta)$ is the posterior PDF. The more different $Q(\theta)$ is from $P(\theta)$, the less accurate this estimate will be (or the more samples we will need to achieve given accuracy).

There is no single correct answer to the question of how we best summarize the posterior PDF. The mean and mode are common, but which is preferred depends on many factors. The PDF should always be plotted, in 1D or 2D projections. The global mode – sometimes called the Maximum A Posterior or MAP estimate – can easily be found, but it may not correspond to the maximum of the posterior PDF found by density estimation. This is because density estimation smooths (and then samples) the posterior PDF. A common approach is to use 1D density estimation to plot the posterior PDFs over each of the parameters separately, thereby marginalizing over all the other parameters. The maxima of these – the "1D modes" – will generally be more stable than the MAP. The `ctsmod` code provides utilities for estimating both the 1D modes and the MAP (see section D.7).

One should be careful when summarizing PDFs for parameters with periodic boundary conditions. Take the phase a sinusoidal model, with domain 0–1. If the true value were near 1, then the MCMC sampling could have values slightly above 0 and slightly less than 1. Their mean would be 0.5. The mode would be a better estimator here, but only if the periodicity of the value is taken into account. We would need to modify our density estimation routine to take this into account. This is done in the function `plot.postSamp1d()` in the file `utilities.R` by wrapping the data.

5 Numerical sampling and integration

Whether calculating the evidence or the K-fold CV likelihood, we need appropriate methods for doing numerical integration. This is done by using a Monte Carlo method of sampling the appropriate probability distribution. I use four different methods:

1. simple Monte Carlo samping of the prior to estimate the evidence;
2. Metropolis sampling of the posterior to estimate the K-fold CV likelihood;
3. parallel tempering sampling of the posterior to estimate the K-fold CV likelihood. The resulting chains can also be used to estimate the evidence (known as thermodynamic integration);
4. sampling the posterior using the affine-invariant ensemble sampler of Goodman & Weare (2010) to estimate the K-fold CV likelihood.

The first samples the prior. All the others are types of Markov Chain Monte Carlo (MCMC) method and are used to sample the posterior. The resulting samples can therefore be used to estimate the parameter posterior probability distributions also. These methods are now explained. Alternative possible methods for estimating the evidence are summarized in section F.

5.1 Simple Monte Carlo

This can be used to approximate the evidence (equation 9).

Uniform sampling. The most straight-forward approach is to sample θ from a uniform distribution over a hyperrectangle with sufficiently large bounds. In one dimension the Monte Carlo approximation is

$$\int_\theta f(\theta)d\theta \approx \sum_{n=1}^{n=N} f(\theta_n)\delta\theta = \frac{\Delta\Theta}{N}\sum_{n=1}^{n=N} f(\theta_n) \tag{23}$$

where $f(\theta) = P(D|\theta, M)P(\theta|M)$, the sample $\{\theta_n\}$ is drawn from a uniform distribution, $\Delta\Theta$ is the range of θ from which they are drawn, and $\delta\theta = \Delta\Theta/N$ is the average spacing between the samples. (We could instead sample regularly.) If $P(\theta|M) = 1/\Delta\Theta$ then this just reduces to the average of samples. This was used in Bailer-Jones (2011), but is not used in `ctsmod`.

Sample the prior (arithmetic mean of likelihoods). The generalization of the Monte Carlo principle is that for a function $f'(\theta)$

$$\int_\theta f'(\theta)P(\theta)d\theta \approx \frac{1}{N}\sum_{n=1}^{n=N} f'(\theta_n) \tag{24}$$

where the sample $\{\theta_n\}$ has been drawn from the PDF $P(\theta)$.[2] With $f'(\theta) = P(D|\sigma, \theta, M)$ and $P(\theta) = P(\theta|M)$, we see that by drawing samples from the prior and then calculating the likelihood at these, the evidence for model M is just the average of these likelihoods. This is used in `ctsmod`. Examples of prior PDFs for the parameters of various models are shown in Table 2. We can assume that these priors are all independent, in which case the prior PDF over several parameters is just the product of these. Sampling from these function is easy and quick (e.g. `rgamma`, `rnorm` in R), so we do not need to use MCMC.

The first method above (uniform sampling) will be very inefficient for parameter spaces with dimensionality more than about two, because the vast majority of the samples will have a very small prior and/or likelihood, and so most will not contribute to the evidence. Thus our estimate of the evidence will be dominated by relatively few samples and may even miss the most significant parts of the likelihood function, implying an estimate with large variance. The second method diminishes this problem to some degree by sampling from the prior. But the more informative the data, the narrower the likelihood function becomes comapred to the prior, so again most of the samples will be negligible. This problem becomes more accute the higher the dimensionality of the parameter space.

[2] This is just equal to the expectation value of $f'(\theta)$ under $P(\theta)$. If $P(\theta)$ is not normalized, then we instead have

$$\frac{\int_\theta f'(\theta)P(\theta)d\theta}{\int_\theta P(\theta)d\theta} \approx \frac{1}{N}\sum_{n=1}^{n=N} f'(\theta)\,. \tag{25}$$

Note that the samples we drawn from $P(\theta)$ will be the same whether or not it is normalized, so we don't need a normalized PDF in order to calculate these expectations. The uniform distribution is a special case, however, because it only becomes a proper, i.e. normalizable, distribution if we specify a range, $\Delta\theta$. In that case $P(\theta) = 1$ and $\int_\theta P(\theta)d\theta = \Delta\theta$ and equation 25 reduces to equation 23.

5.2 Metropolis (and parameter transformations)

I use the standard Metropolis algorithm with a Gaussian sampling distribution. As this is a symmetric sampler, the Hastings factor is not needed. Usually I use a diagonal covariance matrix (although a non-diagonal matrix can be produced by specifying a common correlation coefficient between all parameters).

Those model parameters which do not naturally have an infinite range may be transformed in order to be commensurate with Gaussian sampling. (As of v20 this is done for all such parameters except phase.) When this is done, the Jacobian, J, must be taken into account in the Monte Carlo algorithm. Specifically, the normal Metropolis ratio $P(\theta_p)/P(\theta_c)$ – where θ_c is the current value of the parameter and θ_p is the proposed value – is replaced with[3]

$$\frac{P(\theta_p)}{P(\theta_c)}\frac{J(\theta_p)}{J(\theta_c)} \,. \tag{26}$$

As all the following transformations are one-dimensional, the total Jacobian for all parameters is just the product of the individual Jacobians (the determinant of the diagonal Jacobian matrix). Let x be the natural parameter and y its transformation. Suitable tranformations and their corresponding Jacobians, $J = \partial x/\partial y$ (note the orientation of this derivative), are as follows.

- All parameters with a range $(0, \infty)$, such as frequency, are logarithmically transformed, the inverse transformation of which is the exponential.

$$y = \log_{10} x \;\Leftrightarrow\; x = 10^y \qquad J = \ln(10)\, e^y \text{ with range } (0, \infty) \tag{27}$$

- Parameters with a range $(0, 1)$ *could be* transformed using the logit function, the inverse of which is the logistic (sigmoid).

$$y = \ln\frac{x}{1-x} \;\Leftrightarrow\; x = \frac{1}{1+e^{-y}} \qquad J = \frac{e^{-y}}{(1+e^{-y})^2} \text{ with range } (0, 1/4) \tag{28}$$

  While this could be used for the phase parameter in sinusoidal functions, I actually use no transformation at all. The parameter is kept within the range $(0, 1)$ by use of the (uniform) prior: proposals outside this region give zero prior probability, so will be rejected by the MCMC sampling algorithm.[4]

- Parameters with a range $(-1, +1)$ can be transformed using 2arctanh.

$$y = \ln\frac{1+x}{1-x} \;\Leftrightarrow\; x = \frac{1-e^{-y}}{1+e^{-y}} \qquad J = \frac{2\,e^{-y}}{(1+e^{-y})^2} \text{ with range } (0, 1/2) \tag{29}$$

  This is used for the variable phase parameters in `FuncQuasiSinusoid`, f_x (see section A.2).

[3] One could still use the original Metropolis algorithm if you instead express the posterior PDF as a function of the transformed variable. This is of course equivalent to using the Jacobian. This is of course not the same as just replacing the variable with its transformed version in the posterior!

[4] In v19 of the code I used a circular transformation for the phase parameter ϕ in all existing models (in which case there is no transformation) and did not impose any explicit prior. The inverse transformation of this is $y \bmod 1$, which keeps the phase within the range $(0, 1)$ in the function value. However, both this and the logit transformation still generate problems when used with emcee. Specifically, they do not prevent the *transformed* values of phase become very large (positive or negative), which can result in the proposed parameter in the emcee algorithm giving a numerical error. See section E.2. If we only used the Metropolis algorithm then the circular transformation is preferable, because all proposed phase steps have uniform non-zero PDF, and the transformed parameter (assuming a a reasonable standard deviation for the Gaussian sampler) will never reach numerically large values (more than around $1e + 310$) within any plausible number of iterations.

- For the straight line fit `FuncLinear`, I also use a transformation, but there to transform the infinite range of the gradient, m, into the range $(-\pi/2, +\pi/2)$ of the angle, A. I do this so that the Gaussian sampler can move continuously across what would otherwise be boundaries at $m = \pm\infty$. See section A.1 for an explanation. The transformation ($m \equiv x, A \equiv y$) is

$$y = \arctan(x) \;\Leftrightarrow\; x = \tan(y) \qquad J = \frac{1}{\cos^2 y} \text{ with range } (0, 1) \tag{30}$$

The Metropolis algorithm works with the transformed variables.[5] The standard deviations of the sampling covariance matrix are set to fixed, relatively small values, typically 0.05–0.1 for the logarithmically transformed parameters (these are then scale factors). A consequence of this scaling is that the parameter can never exactly reach the extreme values (zero for the log transformation), but this is not necessarily a disadvantage. Note, however, that we potentially get numerical problems at these extreme values (see section E.2).

5.3 Parallel tempering and thermodynamic integration

Parallel tempering

One of the problems with standard Metropolis is its fixed step sizes, i.e. the fixed covariance matrix in the sampling distribution. Steps which are too large will not sample the posterior finely enough; steps which are too small will take far too long to sample the distribution. This is particularly problematic if the posterior is more than a few dimensional, highly peaked and/or multimodal, one or more of which is often true. In practice we often end up with poor chains which poorly represent the true posterior.

Parallel tempering is a method of overcoming this, by effectively adapting the step size while maintaining the condition of detailed balance (in some limit). The idea is to run multiple chains in parallel, each at a different "temperature", and permit them to interact. The "hot" chains can make larger steps and search the space more widely. Their parameter samples can be passed down to the "cold" chain, which represents the target distribution, which searches more finely. For an overview see Earl & Deem (2005), and for an example see Gregory (2005).

We achieve parallel tempering by modifying the parameter posterior PDF (e.g. equation 18) to be

$$P(\theta|D, M, \beta) \;=\; P(D|\theta, M)^{\beta} P(\theta, M) \tag{31}$$

(to within a normalization constant, and dropping σ from the notation), where $0 \leq \beta \leq 1$ is the inverse temperature. $\beta = 1$ corresponds to the target distribution, and the lowest temperature (the "cold chain"). A value of β less than one tends to flatten out ("heat up") the distribution, thus making it more likely that transitions proposed by the normal Metropolis algorithm will be accepted. $\beta = 0$ modifies the likelihood to make it constant, in which case $P(\theta|D, M, \beta)$ is just equal to the prior.

In parallel tempering we set up N_{chain} chains (typically 5–10) with $\beta = (0, \ldots, \beta_c, \ldots, 1)$ (these could be equally spaced, although other choices are possible). We run the Metropolis algorithm on each in parallel, with the posterior defined as in equation 31. After every N_{swap} iterations (perhaps 10–50), we choose one

[5] The forward transformation functions, `trans.theta()`, are only applied once – to the initial conditions – within `sample.posterior()` before starting the MCMC. The inverse transformations, `invtrans.theta()`, are applied immediately before every function evaluation – in `calc.post.mcmc()` – as well as to the final sets of samples before they are returned (also in sample.posterior()). The Jacobian functions, `jacobian.theta()`, take the transformed parameters (y) as their inputs, so the function which computes the total Jacobian, `calc.jacobian.mcmc()`, does not need to apply the inverse transformation.

of the chains $c = (1, \ldots, N_{\rm chain} - 1)$, as a candidate for swapping its parameter values, θ_c, with those of its cooler neighbour, θ_{c+1}. We make this swap with probability

$$\min\left\{1,\ \frac{P(\theta_c|D,M,\beta_{c+1})}{P(\theta_c|D,M,\beta_c)}\frac{P(\theta_{c+1}|D,M,\beta_c)}{P(\theta_{c+1}|D,M,\beta_{c+1})}\right\}\ . \tag{32}$$

The first fraction in the above is the ratio of the PDF in chain c which we would have after making the swap, to what it was before we made the swap. The second is the same but for chain $c+1$. Thus if the swaps are both favourable, both terms are greater than one, and the swap will be made. If neither is favourable, the product is less than one, and the less favourable the less likely there will be a swap. If one is favourable and the other not, we may get a swap (with variable probability). Through these interactions, the broader searching of the hotter chains will explore parts of parameter space which can be taken up for finer search by the cooler chains. At the end, we only use the cold chain for calculating the likelihood.

The number of chains needs to be large enough to ensure close enough spacing and thus sufficient interaction. Initial tests suggest that in situations where the Metropolis algorithm results in poor samplng (e.g. chains with low acceptance rates), parallel tempering can improve this.

Note that the Jacobian for any variable transformations enters in the same way as for the basic Metropolis algorithm: the ratio of the posterior PDFs is replaced by the equation 26. The expression for the chain swap probability, equation 32, is unchanged because the Jacobians, which are only functions of θ, cancel.

Thermodynamic integration

We can also use the resulting chains to estimate the evidence, using the method of thermodynamic integration (e.g. Friel & Pettitt 2008, Lartillot & Philippe 2006, or Gregory 2005 section 12.7).

First we define a partition function as the integral of the modified posterior in equation 31 (which is sometimes called the "power posterior")[6]

$$Z(\beta)\ =\ \int P(D|\theta,M)^\beta P(\theta,M)d\theta \tag{33}$$

It can then easily be shown (see references) that

$$\begin{aligned}\frac{d}{d\beta}\ln Z(\beta)\ &=\ \frac{\int\ln[P(D|\theta,M)]\,P(D|\theta,M)^\beta\,P(\theta|M)\,d\theta}{\int P(D|\theta,M)^\beta\,P(\theta|M)\,d\theta}\\ &=\ \langle\ln P(D|\theta,M)\rangle_\beta\end{aligned} \tag{34}$$

i.e. the expectation of the natural logarithm of the likelihood with respect to the modified posterior with value β. This is easily estimated as the average of likelihoods obtained from the samples (chain) for that value of β (after the burn-in), which we denote with $\{\theta_n\}_\beta$, i.e.

$$\langle\ln P(D|\theta,M)\rangle_\beta\ \approx\ \frac{1}{N}\sum_{n=1}^{n=N}\ln P(D|\theta_{n,\beta},M) \tag{35}$$

where N is the number of samples. Now, it is clear from equation 34 that

$$\begin{aligned}\int_0^1 d\ln Z(\beta)\ &=\ \ln Z(1)-\ln Z(0)\\ &=\ \int_0^1\langle\ln P(D|\theta,M)\rangle_\beta\,d\beta\end{aligned} \tag{36}$$

[6] It would be more accurate to write this as $Z(D|\beta,M)$

From equation 33

$$Z(1) \;=\; \int P(D|\theta, M)P(\theta|M)d\theta \;=\; P(D|M) \quad \text{and} \tag{37}$$

$$Z(0) \;=\; \int P(\theta|M)d\theta \;=\; 1 \tag{38}$$

so we can finally write equation 36 as

$$\ln P(D|M) \;=\; \int_0^1 \langle \ln P(D|\theta, M)\rangle_\beta \; d\beta \; . \tag{39}$$

This equation tells us that the natural logarithm of the evidence is the integral over β of the quantities in equation 35. It is, in some sense, an average of evidences calculated for different temperatures. We can approximate this integral numerically, either using a simple average (as β varies from 0 to 1), or better, using the trapezium rule. The more chains we have, the more accurate this approximation will be.

5.4 Affine-invariant ensemble sampler (emcee)

The Metropolis algorithm is non-adaptive, in the sense that the typical step sizes in the parameter space are constant (dictated by the fixed covariance matrix). An alternative MCMC algorithm has been defined by Goodman & Weare (2010). (See also Foreman-Mackey et al. 2013, who call their implementation emcee.) This does not make use of any proposal distribution. Instead, it uses a set of W "walkers", each of which is a vector of the model parameters. These explore the parameter space simultaneously, with the position of each walker being updated at each iteration by the position of the others. Let $\theta_w(n)$ be the position of walker w in the N dimensional parameter space at iteration n. To update its position, we select one of the other walkers at random, call this x. The proposed step of walker w for the next iteration is

$$Y = \theta_x + z(\theta_w - \theta_x) \tag{40}$$

where z is a (positive) random variable drawn from the distribution $g(z)$. This is known as the stretch move, because w is stretched along the line joining it to v, either towards it or away from it (unless $z = 1$, in which case it does not move).

$$g(z) \;=\; \begin{cases} \dfrac{1}{\sqrt{z}} & \text{if } \; z \in [1/a, a] \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

where $a > 1$. I follow the advice of Foreman-Mackey et al. (2013) and set $a = 2$. Drawing from $g(z)$ is actually trivial because it has an analytic cumultative density function, so we can use inverse transform sampling. A random sample from z is given by

$$z = \frac{1}{a}\left(1 + (a-1)U\right)^2 \tag{42}$$

where U is the uniform distribution over $(0, 1)$. Having calculated the proposed move, Y, it is accepted with probability

$$q \;=\; \min\left(1, z^{N-1}\frac{P(Y)}{P(\theta_w(n))}\right) \; . \tag{43}$$

where P is the function we are sampling. i.e. the posterior PDF. Note that on account of the z^{N-1} factor, proposals with a higher posterior PDF are not always automatically accepted. If we have transformed the

variables, then we must take into account the Jacobian by multiplying $P(Y)/P(\theta_w(n))$ by $J(Y)/J(\theta_w(n))$ (see section 5.2).

If the move is accepted, the walker is updated immediately. Thus when selecting walker x above, we select from a mixture of walkers at iteration n and walkers already updated to iteration $n+1$. This asynchronous updating is actually required in order to maintain detailed balance.[7] If the move is not accepted the walker remains at its previous position. At the end of the iteration the position of all walkers (whether updated or not) are copied to the chain of samples of the PDF.

The positions all the walkers at all iterations is then our sampling of the function. Generally we will want to exclude the early samples as a burn-in period. In ctsmod we initialize the walkers by drawing from a (diagonal) multivariate Gaussian with small standard deviations.

The walkers cover an infinite range, so the same parameter transformations for the parameters as described in section 5.2 are use here. (The parameter transformations are actually defined in ctsmod as methods of the time series model functions themselves, so can be used by any sampling algorithm.) However, it should be noted that emcee may not prevent the (transformed) parameters extending to very large (positive or negative) values, which can cause problems: see section E.2.

6 Application procedure

Given a data set and a time series model we wish to evaluate, the procedure for applying the model is as follows: (1) define the hyperparameters of the prior parameter PDFs, as well as the standard deviations of the MCMC sampling PDF and its initial values; (2) select a partitioning of the data (normally we will use LOO-CV, so the choice is unique); (3) for each partition of the data, use MCMC to sample the posterior PDF, retaining the value of the likelihood at each parameter sample. Average these likelihoods to get the partition likelihood (equation 16); (4) sum the logarithms of the partition likelihoods to get the K-fold CV log likelihood (normally the LOO-CV log likelihood) (equation 17). Note that each partition provides a posterior PDF, which we could plot and summarize. In order to calculate the evidence for a model (equation 9), then after step (1) we sample the prior PDF and use equation 24. Section D outlines the structure and use of the code `ctsmod` which implements this work.

In section C.3 I define the *no-model*, the model which assumes that the data are just Gaussian variations – with standard deviation given by the error bars – about the mean of the data. As this model has no parameters, its likelihood, L^{NM}, is equal to its LOO-CV likelihood and its evidence. This is therefore a convenient baseline against which to compare all other models, so we may report LOO-CV likelihood/evidence for models relative to this, i.e. $\log L_{\mathrm{LOO-CV}} - \log L^{\mathrm{NM}}$ and $E - \log L^{\mathrm{NM}}$.

Once we have calculated the K-fold CV likelihoods (or evidences) for a number of models, we need to compare them. It is somewhat arbitrary how large the difference in the log likelihoods must be before we identify it as "significant". Clearly very small differences are not significant, as small changes in the priors or the data (within the assumed uncertainties) would produce "acceptable" changes in the likelihoods. We may choose only to consider the difference between models as noteworthy if their log (base 10) likelihoods differ by more than 1.

[7] Note that Algorithm 2 in Foreman-Mackey (2013) is actually wrong in this respect, as it claims that walker x is only selected from the sample at iteration n (David Hogg, private communcation, 22 March 2013).

A Deterministic models

A.1 Linear model (`TSMod1=FuncLinear`)

A linear model is normally represented as $z = a_0 + mt$, in which a_0 is the intercept with the $t = 0$ axis. However, for an arbitrary time scale this axis may lie far from the data, making it difficult to specify priors on this parameter. A more convenient parametrization is

$$z = m(t - t_0) + z_0 \tag{44}$$

because (t_0, z_0) is the "middle" of the data. Priors on this (plus initialization for MCMC) are more obvious, and the price paid – an additional parameter – is probably worth it. Assuming t can be positive or negative, a Gaussian prior is appropriate for t_0 (and this is adopted in `ctsmod`). The same applies to z, although if it must be constrained to be positive, a Gamma prior is appropriate for z_0. (Both options are available for this parameter in `ctsmod`.)

The gradient, m, of the model can be considered as $m = \tan(A)$, where A is the angle of the slope from the t axis. The gradient – rather than A – is adopted as the model parameter as it seems more useful to get results (posteriors) over m than over A. Nonetheless, a convenient prior is one which is uniform in A (from $-\pi/2$ to $+\pi/2$), not least because this has no free parameters, yet covers all possible gradients.[8] This prior is

$$P(A) = \frac{1}{\pi} \tag{45}$$

The corresponding prior in m, $P(m)$ is given by

$$\begin{aligned} P(m) &= P(A)\frac{dA}{dm} \\ &= P(A)\cos^2(A) \\ &= \frac{1}{\pi}\frac{1}{1+m^2} \end{aligned} \tag{46}$$

which is the Cauchy distribution. We may want to constrain m to be positive (or negative), in which case we just set $P(m)$ to be zero for negative (positive) values of m, and double the above expression (as the Cauchy distribution is symmetric). This is provided for in `ctsmod` by setting the hyperparameter `sign` of this prior to "pos" or "neg" (setting it to anything else applies no constraint).

A separate issue is how to represent the slope parameter in the MCMC sampling, which uses a Gaussian sampler. Sampling linearly in $A = \arctan(m)$ is convenient because it automatically imposes periodic boundary conditions: If the current value of A is 89.5°, for example, and the step is $+2^\circ$ taking us to 91.5°, this is equivalent to -88.5°, as $\tan(91.5^\circ) = \tan(-88.5^\circ)$. If we instead sampled in m, we would never be able to achieve very steep functions (close to $A = \pm 90^\circ$), because the step sizes in m would have to become arbitrarily large.

A fixed step size in A is probably appropriate if the $|m|$ is not too large. But as this gets larger – as A tends towards 90° – a fixed step size in A coresponds to increasingly larger changes in m, even infinite ones (as we can cross $A = 90^\circ$). This is inconvenient because we loose sensitivity to relevant gradient changes. Trying to adapt the step sizes is probably hopeless, so in general we should rescale our data such that the magnitudes of the expected gradients are not too large, say less than 50 or so. Of course, if we rescaled in this way we would not require the ability to cross the $A = \pm 90^\circ$ line, in which case sampling in m becomes

[8] A uniform prior over m would be improper, and so unusable in the evidence calculation.

feasible again. But sampling in A has the slight advantage of being less sensitive to the exact scale of the two variables, so this is used in `ctsmod`.

When applying this model to data with no intrinsic slope, it will tend to return a PDF centered around zero slope. This follows because the prior – equation 46 – has its maximum at $m = 0$. As this same prior is uniform in A, which has no preferred direction, this may seem paradoxical. But it makes sense when we remember that we are inferring the probability *density* function over m, and not over A.

A.2 Quasi-sinusoidal model (`TSMod1=FuncQuasiSinusoid`)

This model is like the sinusoidal model but has an additional time-variable phase term, $\Psi(t)$, shown in Table 2, e.g. $\frac{a}{2}\cos[2\pi(\nu t + \phi + \Psi[t])]$ for `FuncQuasiSinusoidZero`. There are many ways in which one could parametrize this. Here I define a fixed number $(N_f + 1)$ of "anchor points" at different times, t_x, generally chosen to span the observed time series. Associated with each of the points is a phase value, f_x. The time-variable phase function is then given by fitting a cubic spline to these points. The times of the anchor points are fixed, so the parameters of this variable-phase term are therefore just $\{f_x\}$. The first anchor point, $x = 0$, f_0 is fixed to zero. (That's because the overall phase shift is provided by ϕ.) In the current implementation $N_f = 4$. This corresponds to five anchor points, the first being $f_0 = 0$, so the model has four parameters. ($N_f = 4$ is hard-wired into the code, because for reasons I will not go into here, each parameter f_x is represented explicitly and not in a vector format). The times of the anchor point must be specified by the user in the setup file when `TSMod1` is initialized using the `fptimes` parameters, e.g.

```
TSMod1 <- new("FuncQuasiSinusoid", fptimes=seq(from=0, to=100, length.out=5))
```

which fixes the anchor points to be $(0, 25, 50, 75, 100)$. Usually we will space the anchor points uniformly. Note that although $f_0 = 0$, we must still provide its anchor time as the first element of `fptimes`. An example of $\Psi(t)$ fit to four specific choices of $\{f_x\}$, as well as the resulting function for `TSMod1` (for some values of the a, ν, ϕ) is shown in Fig. 4.

The four parameters $\{f_x\}$ are assigned uniform priors over the range $(-1, +1)$. For the MCMC samping they are transformed onto the infinite range using the 2arctanh function (see section 5.2; a step sizes (in `sampleSD`) of 0.1 is recommended.) Because a continuous function ($\Psi(t)$) is fit to $\{f_x\}$, it is important that we do not use the periodic boundary condition in this transformation. If we did, then if one of the f_x were near to the boundary (say at $+1$), then a very small change which took it over the boundary to the other side (to -1) would result in a radical change in the fitted function $\Psi(t)$, and thus destroy the smooth dependence of the model on its parameters.

Arguably it would be sufficient (preferable?) to constrain the $\{f_x\}$ to the range $(-1/2, +1/2)$ rather than $(-1, +1)$, as the former covers the full range of the cosine function, and having double the range could result in degeneracies in the MCMC estimate of the posterior.

Simulated data can be generated from this model using the function `sim.asfuncquasisinusoid` in `ctsmod_genfunctions.R`.

B Fully stochastic processes

The signal component of the time series model is the PDF $P(z_j|t_j, \theta, M)$. For a physical process which has a well-defined time-variable signal on top of which there is some randomness, section 2.3 showed

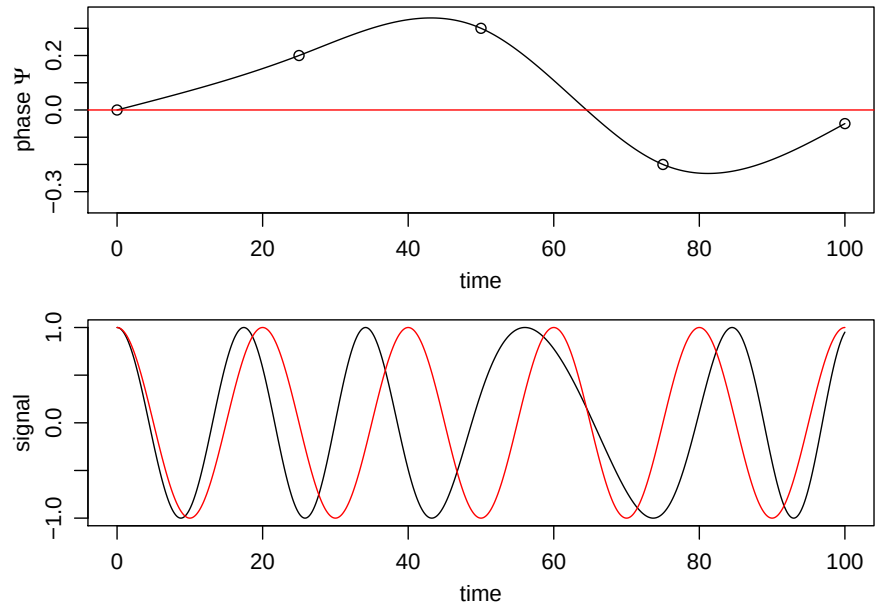

Figure 4: The time-variable phase model `FuncQuasiSinusoidZero`*. The top-panel shows (black curve) the time-variable phase term,* $\Psi(t)$*, with the anchor points* $t = (0, 25, 50, 75, 100)$ *with corresponding phase values as* $f = (0, +0.2, +0.3, -0.2, -0.05)$ *(open circles). The red line is* $\Psi = 0$*. The lower panel shows (black curve) the resulting quasi-sinusoidal variation with* $a = 2$*,* $b = 0$*,* $\nu = 0.05$*,* $\phi = 0$*. The red curve in the lower panel is the strictly periodic function, i.e. with* $\Psi = 0$*.*

a convenient way of expressing this as two independent subcomponents: one which describes the time-dependence of the mean of the PDF and the other which describes the PDF itself and its time-independent parameters (e.g. its variance).

A fully stochastic process, in contrast, is one in which all of the parameters of the PDF in general have a time dependence. Given a functional form for the time dependence of these parameters, we can in principle just introduce this into the parameter θ and calculate the likelihood as before. A simple fully stochastic process is one with a constant mean and variance, a white noise process. This is achieved by setting TSMod1 to a uniform model ($\eta = b$ in equation 4) with TSMod2 a Gaussian.

However, incorporating a stochastic process which has memory, such as a Markov process, is more complicated. Here I show how to introduce a particular but widely used stochastic process, the Ornstein–Uhlenbeck process.

B.1 The Ornstein–Uhlenbeck process

The Ornstein–Uhlenbeck (OU) process (Uhlenbeck & Ornstein 1930) is a stochastic process which describes the evolution of a scalar random variable, z with time, t (for $t > 0$). The equation of motion (Langevin equation), can be written

$$dz(t) = -\frac{1}{\tau} z(t) dt + c^{1/2} \mathcal{N}(t; 0, dt) \tag{47}$$

where τ and c are positive constants, the *relaxation time* (units t) and the *diffusion constant* (units z^2t^{-1}) respectively, dt is an infinitesimally short time interval, $\mathcal{N}(t;0,dt)$ is a Gaussian random variable with zero mean and variance dt, and $dz(t) = z(t+dt) - z(t)$. The OU process is the continuous-time analogue of the discrete-time AR(1) (autoregressive) process, and is sometimes referred to as the CAR(1) process. There are alternative, equivalent forms of this equation of motion.[9] For more details see Gillespie (1996) (and Gillespie 1996b for a nice introduction to continuous Markov processes).

The OU process is stationary, Gaussian and Markov.[10] The PDF of $z(t)$ is Gaussian with mean and variance given by

$$\mu_z = z_0\upsilon \tag{48a}$$

$$V_z = \frac{c\tau}{2}(1-\upsilon^2) \tag{48b}$$

respectively, for any $t > t_0$, where $z_0 = z(t{=}t_0)$ and

$$\upsilon = e^{-(t-t_0)/\tau} \tag{49}$$

(see, for example, Gillespie 1996b, section II.D). Given the initial condition z_0 at t_0, we know the PDF of the process at any subsequent time. The relaxation time, τ, determines the time scale over which the mean and variance change. The diffusion constant determines the amplitude of the variance. The OU process $z(t)$ is a mean-reverting process: for $t - t_0 \gg \tau$ the mean tends towards zero and the variance asymptotes to $c\tau/2$.

The time integral of $z(t)$ is just

$$w(t+dt) = w(t) + z(t)dt \tag{50}$$

and is called the integrated OU process. In the context of Brownian motion, $z(t)$ describes the velocity of the particle and $w(t)$ its position. The integrated OU process, $w(t)$, is stationary and Gaussian, but not Markov (because its mean depends also on z_0)[11]. The mean and variance of its PDF is

$$\mu_w = w_0 + z_0\tau(1-\nu) \tag{51a}$$

$$V_w = c\tau^3\left[\frac{t-t_0}{\tau} - 2(1-\nu) + \frac{1}{2}(1-\nu^2)\right] \tag{51b}$$

respectively for any $t > t_0$, where $w_0 = w(t = t_0)$. Given the initial conditions z_0, w_0 at t_0, we can determine the PDF of the integrated process at any subsequent time. Its mean tends towards w_0 for $t - t_0 \gg \tau$, but the variance diverges as $c\tau^2(t-t_0)$. The initial variance is zero, of course.

An example of how the moments of both the OU process and the integrated OU process vary with time is shown in Fig. 5.

What kinds of time series are actually produced by these processes? Gillespie (1996) derives exact updating equations to give the values of $z(t)$ and $w(t)$. These are

$$z(t) = z_0\upsilon + n_1\sqrt{V_z} \tag{52a}$$

$$w(t) = w_0 + z_0\tau(1-\nu) + n_2\left(V_w - \frac{\kappa_{zw}^2}{V_z}\right)^{1/2} + n_1\frac{\kappa_{zw}}{\sqrt{V_z}} \tag{52b}$$

[9] In particular when we use the familiar property of the Gaussian that $\mathcal{N}(t;0,dt) = \mathcal{N}(t;0,1)(dt)^{1/2}$.

[10] Put loosely: *Stationary* means that the joint PDF of a set of events from the process is invariant under translations in time; *Markov* means that the present value of the process depends only on the value at one previous time step (equivalently, the future state variable is independent of the past values conditioned on the present value); *Gaussian* means that the joint PDF of any set of points is a multivariate Gaussian, in particular the PDF of a single point is Gaussian.

[11] But $w(t)$ does form a *bivariate* Markov process together with $z(t)$.

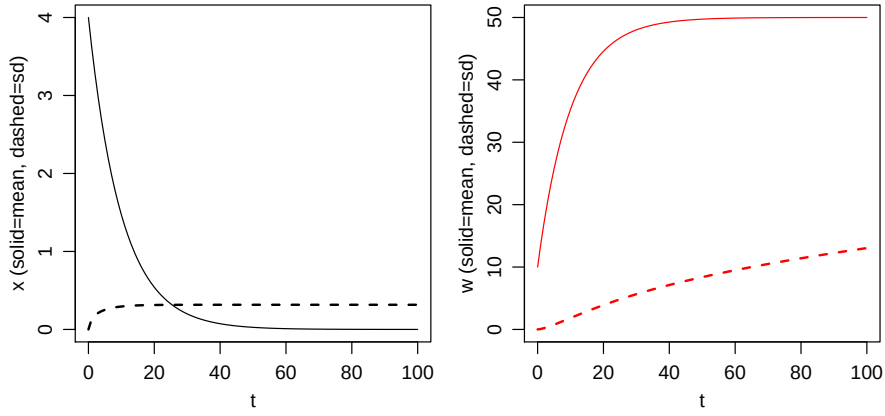

Figure 5: The mean (solid line) and standard deviation ($= \sqrt{V}$) (dashed line) of the Gaussian PDF of the Ornstein–Uhlenbeck process (left) and the time integrated process (right) for parameters $\tau = 10$, $c = 0.02$, $t_0 = 0$, $z_0 = 4$, $w_0 = 10$.

where $\kappa_{zw} = (c\tau^2/2)(1-\nu)^2$ and n_1 and n_2 are statistically independent unit random Gaussian variables. (Note that the first term in equation 52a is just μ_z from equation 48a, and similarly for the other update equation.) The update equation 52a is intuitive: it is just the sum of the mean and a random number drawn from a zero-mean Gaussian with the variance at time t. For a given sequence of time steps, $(t_0, t_1, \ldots)$, we can use these equations to generate an OU process or integrated OU process. Because the time series is stochastic and must be calculated at discrete steps, then even for a fixed random number seed, the generated time series depends on the actual sequence of steps.

Several examples of time series for the parameters shown in Fig. 5 are shown in Fig. 6. Fig. 7 shows the mean and standard deviation of the processes if we now increase the diffusion constant to $c = 20$ (and set $z_0 = w_0 = 0$). Examples time series for these parameter settings are shown in Fig. 8. Note how the larger c produces a dramatic change in the (lack of) smoothness of the curves.

In principle we could model the OU process in `ctsmod` by setting the mean and variance of a Gaussian for TSMod2 (section 2.3 and Table 2) to the functions shown in equation 48 (or equation 51 for the integrated OU process).[12] While not actually wrong, this is very inefficient for the OU process, because it does not take advantage of its Markov property, namely that a measurement of $z(t)$ at any time between t_0 and the current time must provide a better constraint on the process than does z_0. This is a direct consequence of the fact that the variance of the estimation of z increases with time. We will see how to calculate the likelihood for this process properly in section B.2.

B.2 Likelihood of the Ornstein–Uhlenbeck process

A Markov process is one in which we can specify the PDF of the state variable, z_j at time t_j, using $P(z_j|t_j, z_{j-1}, t_{j-1}, \theta, M)$, i.e. there is a dependence on the previous state variable, z_{j-1}. For the OU pro-

[12] Specifically, we would set TSMod2 to be $\frac{1}{\sqrt{2\pi}\sigma_z}\, e^{-(z-\eta[t;\theta_2])^2/2\sigma(t;\theta_2)^2}$ with $\eta[t;\theta_2]$ and $\sigma(t;\theta_2)$ given by equation 48 for the OU process (or by equation 51 for the integrated OU process, i.e. setting $z = w$). TSMod1 would not be used. $\theta_2 = (\tau, c, z_0, t_0)$ for the OU process, and additionally w_0 for the integrated OU process. Note that the initial conditions of the process are parameters of the model. These are implemented as `ProbOUprocNaive` and `ProbIntOUprocNaive` in Table 2. Their use is not recommended.

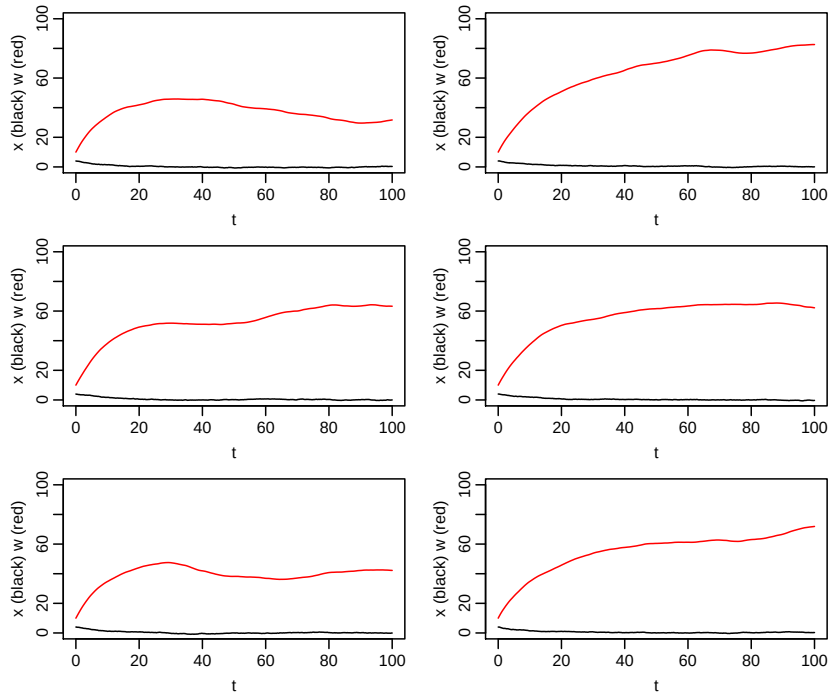

Figure 6: Example time series resulting from the OU process (black) and the integrated OU process (red) for the parameter settings shown in Figure 5 and using $\Delta t = 0.1$.

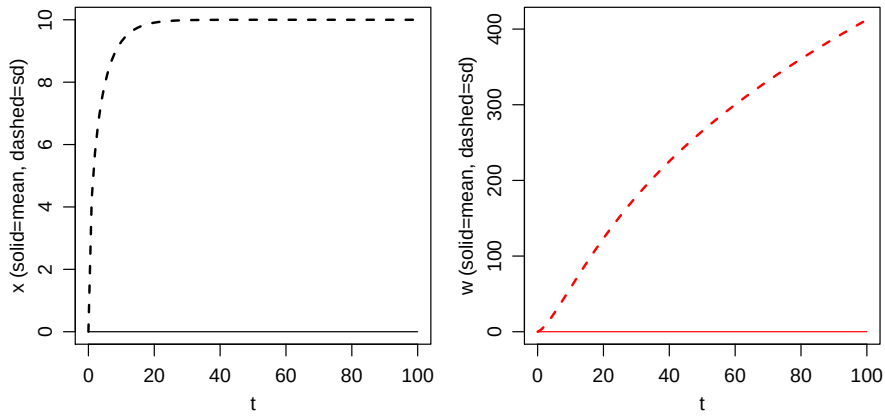

Figure 7: As Figure 5 but now with $c = 20$, $z_0 = 0$, $w_0 = 0$.

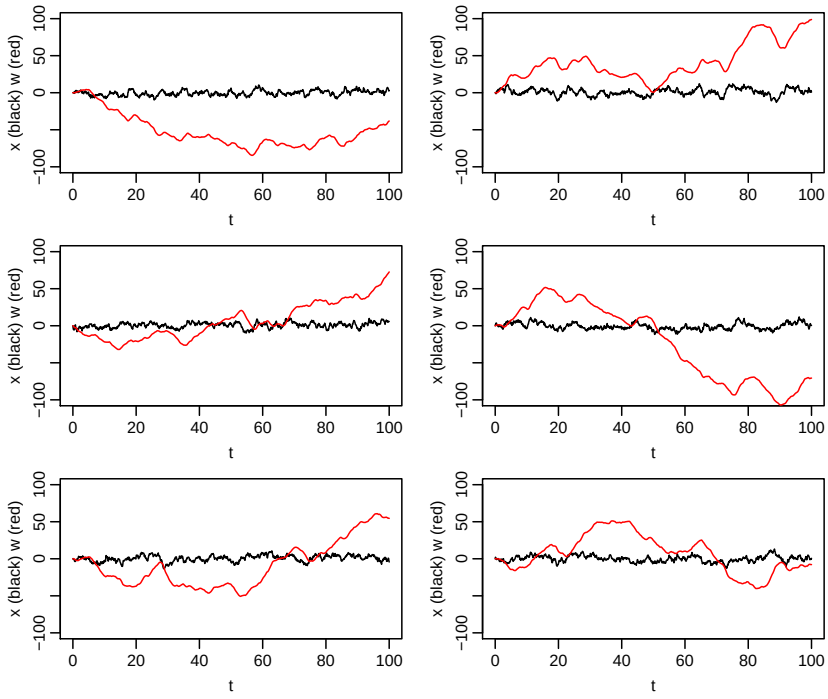

Figure 8: Example time series resulting from the OU process (black) and the integrated OU process (red) for the parameter settings shown in Figure 7 and using $\Delta t = 0.1$.

cess, this PDF is a Gaussian with mean and variance given by equation 48. Clearly, the nearer t_{j-1} is to t_j the better a measurement of z_{j-1} will constrain z_j.

We could therefore write the signal component of the time series model (cf. equation 3) as

$$\begin{aligned} P(z_j|t_j) &= \int_{t_{j-1},z_{j-1}} P(z_j|t_j, z_{j-1}, t_{j-1}) P(z_{j-1}, t_{j-1}|t_j)\, dt_{j-1} dz_{j-1} \\ &= \int_{t_{j-1},z_{j-1}} P(z_j|t_j, z_{j-1}, t_{j-1}) P(z_{j-1}|t_{j-1}) P(t_{j-1})\, dt_{j-1} dz_{j-1} \end{aligned} \tag{53}$$

where conditional independence has been applied in the second line to remove the t_j dependence from the second two terms. Note that everything is implicitly conditioned on M and its parameters θ, but these have been omitted for brevity. The first term under the integral is the PDF for the Markov process we aimed to introduce. The second term is also a PDF for the Markov process but referred to the previous event. We could place that with another 2D integral over (t_{j-2}, z_{j-2}) of exactly the same form as equation 53. We could then continue recursively to achieve a chain of nested 2D integrals going back to the beginning of the time series, and use that in our likelihood calculation. Although this is a plausible and general solution for a Markov process, it is not very appealing.

Fortunately a significant simplification is possible. Let us first neglect the time uncertainties. In that case the event likelihood (equation 7) becomes

$$P(D_j|\sigma_j, \theta, M) = \int_{z_j} P(y_j|z_j, \sigma_{y_j}) P(z_j|t_j, \theta, M) P(t_j|\theta, M)\, dz_j \tag{54}$$

with $t_j = s_j$. $P(y_j|z_j, \sigma_{y_j})$ is the signal part of the measurement model (the time part has dropped out). If this is Gaussian in $y_j - z_j$ (cf. equation 1) and $P(z_j|t_j, \theta, M)$ is Gaussian in z_j, then equation 54 is a

convolution of two Gaussians, which is another Gaussian, multiplied by $P(t_j|\theta, M)$ (which is independent of z_j). A general result is that if f is a Gaussian with mean μ_f and variance V_f, and g is a Gaussian with mean μ_g and variance V_g then

$$\int_{-\infty}^{+\infty} f(y-z)g(z)dz = f \otimes g \tag{55}$$

is a Gaussian with mean $\mu_f + \mu_g$ and variance $V_f + V_g$. For the Gaussian measurement model, f is Gaussian in the argument $y_j - z_j$ with $\mu_f = 0$ and $V_f = \sigma_{y_j}^2$. g is then the time series model.

We now turn specifically to the OU process in order to determine its time series model, $P(z_j|t_j, \theta, M)$. We can derive this from the update equation (equation 52). With the state variable now written as z_j rather than $z(t)$, the update equation is

$$z_j \;=\; z_{j-1}\upsilon \;+\; n_1\sqrt{V_z} \tag{56}$$

with

$$\upsilon \;=\; e^{-(t_j - t_{j-1})/\tau} \tag{57a}$$

$$V_z \;=\; \frac{c\tau}{2}(1-\upsilon^2)\;. \tag{57b}$$

z_j has a Gaussian distribution with mean and variance[13]

$$\mu[z_j] \;=\; \mu[z_{j-1}]\upsilon \tag{58a}$$

$$V[z_j] \;=\; V[z_{j-1}]\upsilon^2 + V_z \tag{58b}$$

respectively (see also Berliner 1996). Specifically, $P(z_j|t_j, \theta, M)$ is a Gaussian with this mean and variance, which are specified by the parameters $\theta = (\mu[z_{j-1}], V[z_{j-1}], \upsilon, \tau, c)$.[14] We will look in a moment at how we estimate $\mu[z_{j-1}]$ and $V[z_{j-1}]$.

We can now write the likelihood, the result of the Gaussian convolution, equations 54 and 55, as

$$\begin{aligned} P(D_j|\sigma_j, \theta, M) \;&=\; P(t_j|\theta, M)\int_{z_j} P(y_j|z_j, \sigma_{y_j})P(z_j|t_j, \theta, M)\,dz_j \\ &=\; P(t_j|\theta, M)\,\frac{1}{\sqrt{2\pi\,V[y_j]}}\,\exp\left(\frac{-(y_j - \mu[y_j])^2}{2V[y_j]}\right) \end{aligned} \tag{59}$$

where the mean and variance of this Gaussian are

$$\mu[y_j] \;=\; 0 \quad + \mu[z_j] \tag{60a}$$

$$V[y_j] \;=\; \sigma_{y_j}^2 + V[z_j] \tag{60b}$$

respectively. Recall that $P(t_j|\theta, M)$ is just the time component of the time series model with $t_j = s_j$. Normally we will use a uniform model (equation 5), so this is just a constant.

To estimate $\mu[z_{j-1}]$ and $V[z_{j-1}]$ we make use of the data, y_{j-1}. For an event t_j, $P(z_j|t_j, \theta, M)$ – which has mean and variance given by equation 58 – is our estimate of the PDF of the state variable at t_j prior to taking into account the measurement y_j. It is therefore the appropriate thing to use to calculate the likelihood of

[13] The variance is just the sum of the variances of the two terms in equation 56. Recall that in general $V(fg) = f^2V(g) + g^2V(f)$, and that $V(\upsilon) = 0$.

[14] If we instead had an actual value of z_{j-1}, then $P(z_j|t_j, \theta, M)$ would be Gaussian with mean $z_{j-1}\upsilon$, variance V_z, and $\theta = (z_{j-1}, \upsilon, \tau, c)$.

y_j, as we have done in equation 59. But in parallel to this we want to use y_j to improve our estimate of z_j, i.e. we want to calculate the posterior PDF of z_j. This is given by Bayes' theorem

$$P(z_j|y_j, t_j) \propto P(y_j|z_j, t_j)P(z_j|t_j) \tag{61}$$

(ignoring the normalization constant $1/P(y_j|t_j)$, and omitting a lot of dependencies). These two terms are again the measurement model (so the dependence on t_j drops out) and the time series model, both of which are Gaussian in z_j. Thus the posterior PDF over z_j is also a Gaussian with mean and variance[15]

$$\mu'[z_j] \;=\; \frac{y_j V[z_j] + \mu[z_j]\sigma_{y_j}^2}{V[z_j] + \sigma_{y_j}^2} \tag{63a}$$

$$V'[z_j] \;=\; \frac{V[z_j]\sigma_{y_j}^2}{V[z_j] + \sigma_{y_j}^2} \tag{63b}$$

respectively, where the prime symbol is used to distinguish these posterior moments from the prior ones from equation 58. It is these quantities which we then use at the *next* event as the estimates of the mean and variance of the state variable. Thus, at iteration (event) j, when we calculate equation 60 and hence the likelihood, we use $\mu'[z_{j-1}]$ and $V'[z_{j-1}]$ as our estimates of $\mu[z_{j-1}]$ and $V[z_{j-1}]$. This is how we introduce a dependence on the previous measurement (the Markov property). We then calculate the mean and variance of the posterior for z_j using equation 63 to use in the next iteration. Thus we have a recurrence relation for the posterior PDF of z_j, at each iteration siphoning off the relevant quantities in order to calculate the event likelihood.

To initialize the process we must specify initial values $\mu[z_1]$ and $V[z_1]$. We use these in equation 60 to calculate $\mu[y_1]$ and $V[y_1]$ and hence the likelihood for the first event, y_1, from equation 59. We then calculate the posterior moments using equation 63. For the next event, $j = 2$, these posterior moments are assigned to $\mu[z_{j-1}]$ and $V[z_{j-1}]$ in equation 58 and the likelihood calculated. The procedure is iterated through all the events.

The model prediction of the OU process is a Gaussian distribution at each event (at time t_j) with mean and variance given by equation 58. Unlike the memoryless time series models, the OU process requires the measurement of the one previous event in addition to the model parameters in order to predict the next event (this is the Markov property). The relevant model prediction of event j is therefore given by the prior distribution of equation 58 – which has not yet looked at y_j – and not by the posterior distribution of equation 63, which has.

The parameters of the process are $\theta = (\mu[z_1], V[z_1], \tau, c)$ (and implicitly the initial time, t_1). Figure 9 shows an example of a simulated OU process and the model predictions thereof.

As it stands, the long-term mean of this OU process is zero. We can introduce the long-term mean as an additional parameter, b, of the model. Equation 58a then becomes

$$\mu[z_j] \;=\; \mu[z_{j-1}]\upsilon + b(1-\upsilon)\;. \tag{64}$$

The variance is unchanged. (See also Brockwell & Davis 2002, section 10.4.) Introducing this corresponds to solving a different Langevin equation, namely one in which we have the additional term $(b/\tau)dt$ on the right-hand-side of equation 47. Note that b is the long-term mean of the process: it is neither the mean of

[15] The product of two Gaussians f and g with means μ_f and μ_g and variances V_f and V_g is another Gaussian with

$$\text{mean} \quad \frac{\mu_f V_g + \mu_g V_f}{V_f + V_g} \quad \text{and variance} \quad \frac{V_f V_g}{V_f + V_g}\;. \tag{62}$$

Note that the mean is just the inverse-variance weighted average of μ_f and μ_g.

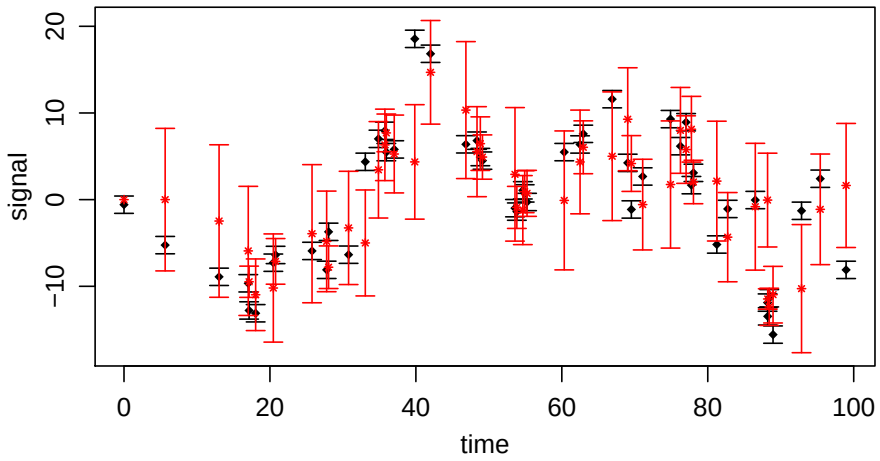

Figure 9: The black points (with a uniform random time distribution) have been simulated from an OU process with parameters $\tau = 10$, $c = 20$ and initial conditions $t_1 = 0$, $z_1 = 0$, to which Gaussian measurement noise with mean zero and unit standard deviation has been added (as indicated by the black error bars). This red points show the predictions of this process using an OU process with parameters $\tau = 10$, $c = 20$, $\mu[z_1] = 0$, $V[z_1] = 0$. The prediction for each event is a Gaussian in the signal with mean and variance given equation 58.

the data set nor of the model-predicted process over the time range of the data. Specifically, $\mu[z_j] \to b$ as $\upsilon \to 0$, which occurs when $\Delta t/\tau \to \infty$. As the data themselves are used to predict the model process, then the mean of the model predictions over the time scale of the data depends on the data as well as on b.

We can now use the likelihood to calculate the evidence and/or to sample the posterior via MCMC. By partitioning the data set we can also use posterior sampling to evaluate the cross-validation likelihood, as described in section 3.2. Note that whatever partitioning we do, when it comes to calculating the partition likelihood for data D_k, we must still use all of the data to predict the full sequence of events. That is, for a given θ_n, we predict the entire sequence of J events using all the data, but then only make use of those event likelihoods which are appropriate. Specifically, to calculate the posterior to use in MCMC sampling (equation 18) we just select the likelihood for the events in D_{-k}, and to calculate the likelihoods in equation 15 we just use the events in D_k. The OU process depends not only on the model parameters but also on the state at the previous time step, so we should not be changing these time steps by removing events when predicting the sequence.

B.3 Literature note

I am not aware of an explicit derivation in the literature either of the above posterior recurrence relation (although Berliner 1996 outlines the same thing) or of the event likelihood for the OU process. Kelly et al. (2009) write down similar equations for the latter (their equations 6–12), but in a significantly rearranged form. Kelly et al. also assume a specific initial value (zero) for the initial state variable, z_0 (their equation 8), whereas I give this a distribution. (In my formulation we can achieve a specific initial value by setting $V[z_1] = 0$.) My expression for the variance (equation 58b) therefore has an additional term compared to theirs (their equation A5, which is V_z in my notation), because they are conditioning on an fixed value of the process at the previous step whereas I assume this itself has a variance, $V[z_{j-1}]$. (See also section 10.4 of Brockwell & Davis 2002.) Closely related formulations of this process – but not the likelihood calculation

– are given in Jones (1986; sections 4 and 5) and Kozlowski et al. (2010; appendix).

B.4 Wiener process

A Wiener process, $z(t)t \geq 0$, can be defined as the process for which

- $z(t)$ is continuous in t for all t;
- the increments, $z(t + \Delta t) - z(t)$, are independent, stationary and have the normal distribution $\mathcal{N}(0, \Delta t)$;
- $z(t) = 0$.

The Wiener process can be considered as a special case of the OU process in which $\tau \to \infty$ (Gillespie 1996b, p. 230). As $\upsilon = e^{-\Delta t/\tau}$, $\upsilon \to 1$ and[16] $\tau(1 - \upsilon^2) \to (\Delta t)^2$. The update equation (equation 56) then becomes

$$z_j \;=\; z_{j-1} \;+\; n_1\sqrt{V_z} \tag{65}$$

where now $V_z = c\Delta t$. The mean and variance of this are

$$\mu[z_j] \;=\; \mu[z_{j-1}] \tag{66a}$$
$$V[z_j] \;=\; V[z_{j-1}] + V_z\;. \tag{66b}$$

These replace the moments in equation 58 in the calcluation of the likelihood of the stochastic process. The rest of the calculation in section B.2 is otherwise unchanged.

The difference with respect to the OU process is that there is no damping, as the relaxation timescale is infinite. The variance of the state variable therefore grows monotonically, and linearly with time. As the expectation value of the state variable is constant, this Wiener process is referred to as being *driftless*. The Wiener process is sometimes taken to describe, in a somewhat idealized form, the position of a particle undergoing Brownian motion.

C Simplifying the event likelihood integration

The two-dimensional integral in equation 7 is in principle straight forward, but because it has to be calculated for every event for every parameter combination in the evidence estimate, it is critical that it is accurate and rapid. As the functions involved may be strongly peaked compared to the range of integration, it is desirable to identify numerical simplifications.

[16] Using a Taylor expansion

$$\begin{aligned}\tau(1 - e^{-2(\Delta t)/\tau}) \;&=\; \tau\left[1 - \left(1 - \frac{2\Delta t}{\tau} + \frac{1}{2!}\left(\frac{2\Delta t}{\tau}\right)^2 - \dots\right)\right]\\ &=\; \tau\left[\frac{2\Delta t}{\tau} - \frac{1}{2!}\left(\frac{2\Delta t}{\tau}\right)^2 + \dots\right]\\ &=\; 2\Delta t \;\text{ as }\; \tau \to \infty\,.\end{aligned}$$

C.1 Integration limits on t and z

In principle the integration extends over the whole two-dimensional real space ($\pm\infty$). In practice we will truncate it according to (a) the measurement model (e.g. negative data may not be permitted) and the range of measured data (taking into account the measurement uncertainties).

C.2 Dropping the stochastic signal component of the time series model (TSMod2 bypass)

If TSMod2 is a Gaussian (equation 3) in which ω is very small compared to the scale of signal variations, then the only contribution to the event likelihood is at the prediction of the signal by TSMod1. For given θ_1, the magnitude of the event likelihood is then dictated only by the measurement model, i.e. how close the measured y_j is to the prediction z_j. In the limit $\omega \to 0$, the signal part of the time series model (equation 2) becomes $P(z_j|t_j, \theta_1, \theta_2, M) = \delta(z_j - \eta[t_j; \theta_1])$. This gives us a purely deterministic signal in the time seres model; we "bypass" TSMod2. The event likelihood integration (equation 7) then becomes a 1D integration[17]

$$P(D_j|\sigma_j, \theta, M) \;=\; \int_{t_j} \underbrace{P(D_j|t_j, z_j = \eta[t_j;\theta_1], \sigma_j)}_{\text{Measurement model}} \; \underbrace{P(t_j|\theta_3, M)}_{\text{Time series model}} \; dt_j \;\; . \tag{67}$$

C.3 Small uncertainties on the measured times

If the uncertainty on the measured time, σ_{s_j}, is very small compared to the time scale over which the time series model varies, then the integral over t_j in the event likelihood for that event will have a significant contribution only for times t_j close s_j. This must hold for any sensible measurement model or definition of uncertainties. The time part of the measurement model can then be approximated by the delta function $\delta(t_j - s_j)$, and the integration over t_j is just unity. If the signal part of the measurement model is Gaussian, the event likelihood equation (equation 7) becomes

$$P(D_j|\sigma_{y_j}, \theta, M) \;=\; \int_{z_j} \underbrace{\frac{1}{\sqrt{2\pi}\sigma_{y_j}} e^{-(y_j - z_j)^2/2\sigma_{y_j}^2}}_{\text{Measurement model}} \; \underbrace{P(t_j = s_j, z_j|\theta, M)}_{\text{Time series model}} \; dz_j \;\; . \tag{68}$$

When TSMod2 is the Gaussian model this becomes

$$P(D_j|\sigma_{y_j}, \theta, M) \;=\; \int_{z_j} \underbrace{\frac{1}{\sqrt{2\pi}\sigma_{y_j}} e^{-(y_j - z_j)^2/2\sigma_{y_j}^2}}_{\text{Measurement model}} \; \underbrace{\frac{1}{\sqrt{2\pi}\omega} e^{-(z_j - \eta[s_j;\theta_1])^2/2\omega^2} \; P(s_j|\theta_3, M)}_{\text{Time series model}} \; dz_j \;. \tag{69}$$

Note that this can be written as

$$P(D_j|\sigma_{y_j}, \theta, M) \;=\; P(s_j|\theta_3, M) \int_{z_j} f(y_j - z_j) g(z_j) dz_j \;. \tag{70}$$

This is just a convolution of two Gaussian functions, f and g, which is another Gaussian with mean equal to the sum of the means of f and g and variance equal to the sum of the variances of f and g. The event

[17] As we now have no stochastic element in either t or z, you may wonder why this integral is over t rather than z, i.e. why there is an asymmetry. The point is that we need to integrate along the path of the (deterministic) function $z_j = \eta[t_j; \theta_1]$. As this only requires one parameter, we only have a one-dimensional integral. Whether we parametrize this with t_j or z_j is unimportant, but having written the function as $z_j = \eta[t_j; \theta_1]$ rather than $t_j = \eta'[z_j; \theta_1]$, t_j is the more natural choice.

likelihood is therefore

$$P(D_j|\sigma_{y_j},\theta,M) \;=\; P(s_j|\theta_3,M)\;\frac{1}{\sqrt{2\pi(\sigma_{y_j}^2+\omega^2)}}\,e^{-(y_j-\eta[s_j;\theta_1])^2/2(\sigma_{y_j}^2+\omega^2)} \tag{71}$$

i.e. involves no integration. Note that the time part of the time series model, $P(t_j = s_j|\theta_3, M)$, is simply evaluated at the measured time, s_j. One particular application of this is to calculate the event likelihood for the *no-model*, the model in which there is no stochastic component, so that the expected value at all times is just the mean of the signal. This is obtained by setting $\omega = 0$ and $\eta = \overline{y_j}$ in equation 71. The total likelihood for the no-model, $L^{\rm NM}$, is the product of these event likelihoods (equation 8). This is a useful baseline model against which to compare the likelihood of other models. As this model has no adjustable parameters, this likelihood is equal to both the evidence and the K-fold CV likelihood.

C.4 Both TSMod2 bypass and negligible time uncertainties

If, in addition to a purely deterministic time series model, we also have negligible uncertainties on time, then the time part of the measurement model in equation 67 is a delta function, $\delta(t_j - s_j)$. The likelihood then involves no integration. If the signal part of the measurement model is a Gaussian, the likelihood is

$$P(D_j|\sigma_{y_j},\theta,M) \;=\; P(s_j|\theta_3,M)\;\frac{1}{\sqrt{2\pi}\sigma_{y_j}}\,e^{-(y_j-\eta[s_j;\theta_1])^2/2\sigma_{y_j}^2}\quad. \tag{72}$$

We also reach this result if we set $\omega = 0$ in equation 71.

C.5 Zero uncertainties on measured time and signal

In the limit of the uncertainties in both s_i and y_j in the measurement model becoming zero, then $P(D_j|t_j, z_j, \sigma_j)$ becomes a delta function at the measured values. In that case the 2D event likelihood integration, equation 7, reduces to a direct evaluation of the time series model at the measured values, i.e. $P(t_j = s_j, z_j = y_j|\theta, M)$ (no integration). This only makes sense if there is some stochastic component, so does not make sense with the TSMod2 bypass.

C.6 Unknown uncertainties.

If the measurement uncertainties are unknown, then we can redefine the evidence as a quantity which is marginalized over these, i.e.

$$\begin{aligned} P(D|M) \;&=\; \int_\sigma P(D,\sigma|M)\;d\sigma \\ &=\; \int_\sigma P(D|\sigma,M)P(\sigma)\;d\sigma\quad. \end{aligned} \tag{73}$$

In other words, we marginalize the previous evidence over the prior for the uncertainties. If we do this for both time and signal for all events, then this is a $2J$-dimensional integration. This is not currently implemented in the `ctsmod` code.

Table 4: Summary of `ctsmod` *files. Only those in the first block are part of the code execution.*

`calc_evidence.R`	samples from the prior to calculate the evidence
`ctsmod_genfunctions.R`	general functions
`ctsmod_likefunctions.R`	likelihood calculation functions
`ctsmod_models.R`	time series and measurement model definition
`kfoldCV.R`	sets up and calls the multiple partition sampling (to calculate the K-fold CV likelihood)
`mcmc.R`	The MCMC algorithms
`run_ctsmod.R`	root code to source. Does parameter/data consistency checks and calculates end results
`sample_posterior.R`	sets up and calls the posterior sampling
`setup_ctsmod.R`	configuration file
`analyse.R`	examples of post-code analyses (uses `utilities.R`)
`install_ctsmod_packages.R`	run to install dependent packages
`utilities.R`	set of utility functions for analysing results

D Overview of the code, its outputs, and analysis functions

D.1 Overview

The model is implemented in the R code `ctsmod`, with the code distributed across several files. The files are summarized in Table 4. For a given model we can do one or more of the following

1. calculate the evidence – i.e. sample the prior (using the simple Monte Carlo method; section 5.1) and calculate the corresponding likelihoods. This is selected by setting `calcEvidence=TRUE`, and is done by function `calc.evidence()`

2. calculate the posterior-averaged likelihood (posterior sampling), defined in section 3.2. This is selected by setting `samplePosterior=TRUE`, and is done by function `sample.posterior()`, which uses one of the MCMC methods (see section 5). These MCMC methods are general functions not specific to `ctsmod`. The function sampled – the posterior PDF – is calculated by the function `calc.post.mcmc()` in `sample_posterior.R`. We use the resulting likelihoods to calculate the posterior-averaged likelihood.[18] The full set of MCMC samples is preserved and can be used in post-processing to plot the posterior PDFs. If parallel tempering is done, all chains are also returned (see below), and these are used to calculate the evidence via thermodynamic integration.

3. calculate the K-fold cross validation likelihood (posterior sampling), defined in section 3.2. This is selected by setting `Npart>1`, and is done by function `kfold.cv()`. This partitions the data into `Npart` sets, calls `sample.posterior()` for each partition, and then calculates the likelihood on the corresponding complementary data sets. For each partition the event numbers, the MCMC samples, and the partition likelihood is recoded. These can be used in post-processing to plot the posterior PDFs. If `Npart` $\geq$ `Nevents`, then we force `Npart=Nevents`, which is leave-one-out CV.

[18] Code – currently commented out – is also present in `run_ctsmod.R` to calculate the DIC, but tests so far have not shown this to be very useful.

See the notes in the file of the function definition for the format of what is returned.

The 2D integration for a single event likelihood calculation is done using `cuhre` in the `R2Cuba` library (Hahn 2005), and the 1D integration is done with `integrate` in the `stats` library. (The full 2D integration is rather slow, and most use of the code to date has been with the "0D" shortcut achieved by assuming the time uncertainties to be negligible, as explained in section C.)

The code makes use of the object oriented programming features of R by defining the different time series and measurement models as S4 classes. Inheritance is used to define more compound models, and function overloading is used to evaluate the priors and functions and sample the priors via generic calls.

The code contains a number of comments and explanations, in particular in the header of the executed file `run_ctsmod.R`.

D.2 Running the code

The code can be run from the linux bash shell in batch mode using

```
R < run_ctsmod.R --no-save --args setup_ctsmod.R &> outfile &
```

This will redirect all output, including errors, to `outfile`. The user-defined inputs and configurations are defined in the setup file `setup_ctsmod.R` (or any other file name), the name of which is assigned to the variable `setup`. R can also be run within the R session simply by sourceing the file `run_ctsmod.R` (which will also default to `setup=''setup_ctsmod.R''` unless this has been defined otherwise by the user).

On completion the entire R session is saved to an R object. This contains everything needed for subsequent analysis. Functions for doing so are in the file `utilities.R`. Examples of their use are provided in `analyse.R`, and some are explained below.

The code will either read data files, or will simulate data using one of several functions implemented in `ctsmod_genfunctions.R` (see section D.6). This is controlled by the variable `obssel` in the setup file and must take one of the strings defined by the variable `obstypes` (listed in the same file). (The translation between `obssel` and which function is called is done in `run_ctsmod.R`). For those strings which refer to simulation functions, the behaviour of these functions is controlled by modifying the function directly. Otherwise, data is read from a file in standard ASCII format: one event per row, with four columns: time (s), standard deviation of time (s.sd), signal (y), standard deviation of signal (y.sd). The file must also have a one line header with named columns (i.e. four strings in that row), although this row is not used.

To run on the grid engine on our multi-node and core machine `karun`, we create the following script file, called `qsubscript.sh`.

```
!/bin/bash
#$ -V
#$ -o /home/calj/ctsmod/
#$ -e /home/calj/ctsmod/
#$ -cwd
#$ -pe make 5
R < run_ctsmod.R --no-save --args setup_ctsmod.R
```

This is submitted to the grid engine from the master `karun` node (only) in the bash shell using
`qsub qsubscript.sh`
and is allocated a number, let's call it `NNNNN`. The `-V` passes all enviroment variables and `-cwd` makes

the directory from which the script was submitted the current working directory (which in the case above needs to the directory where `run_ctsmod.R` and `setup_ctsmod.R` situated, although we could add directory paths to these names). The option `-pe make 5` reserves 5 cores for the job. This needs to be at least as large as the number of cores requested by the program. Running the script produces two output files, `qsubscript.sh.o<NNNNN>` and `qsubscript.sh.e<NNNNN>` in the directories specified following the `-o` and `-e` entries in the script file, respectively. The first is the standard output, the second the standard error, from the job execution.

D.3 Using the various models

This is a brief guide to using the various models and methods of calculating the likelihood within the `ctsmod` code. The generic syntax is
`TSModX <- new("name_of_model")`
Note that when calculating the K-fold CV likelihood, the likelihood will never even be calculated if the prior PDF is zero. (This is determined in `calc.post.mcmc()` in tt sample_posterior.R.)

1: Two measurement models are currently implemented. `TwoGaussian` is Gaussian in both time and signal. `TimeUniformSignalGaussian` is uniform in time and Gaussian in signal.

2: The "standard" use of `ctsmod` is to use TSMod2 = `ProbGaussian` with one of the TSMod1 models shown in Table 2 to describe its mean, and TSMod3 = `ProbUniform`. Which parameters to adjust are set with `thetaXFlag` (X=1,2,3) and the fixed values of any parameters are set with `thetaXFixed`. The likelihood is the product of event likelihoods, each being calculated by the function `calc.event.like`, which is called (the event likelihoods combined) by the function `calc.like.isolated`. This normally results in a 2D integration, the integrand of this is provided by the function `calc.event.like.integrand`. The following points note the exceptions (explained in section C).

3: `FuncUniform` has two alternative priors for its parameter b, either a Gaussian or a Gamma, for variables which can be positive/negative or non-negative respectively. They are selected by setting `priorform="Gaussian"` or `priorform="Gamma"` in the `TSMod1` definition, e.g.
`TSMod1 <- new("FuncUniform", priorform="Gamma")`
Gaussian is the default.

4: `FuncLinear` has the same two alternative priors for its parameter z_0 as does `FuncUniform` for b. Indeed, `FuncLinear` is implemented using the `FuncUniform` class. The slope (gradient) can be constrained by setting the hyperparameter `sign` to be "pos" or "neg". If this is done, then the MCMC sampler could some up with a sample which has zero prior PDF, meaning this sample will be rejected. For more details on `FuncLinear`, see section A.1.

5: To bypass TSMod2 we set TSMod2 = `BypassTSMod2`. This results in a 1D integration (over t) for the event likelihood (equation 67), the integrand for which is provided by the function `calc.event.like.integrand.2`. (If the time uncertainties are also negligible, then the next point applies and a "0D" integration is done instead.)

6: If the uncertainties in the measured times are small compared to the time scale variations in TSMod1, the event likelihood can be calculated analytically without integration (section C). This will be used if $\sigma_{s_j} = 0$ or it can be forced by setting `negtsd` = `TRUE`. This currently also requires that have TSMod2 set to `ProbGaussian` or `BypassTSMod2` and that a measurement model is used in which the signal component is Gaussian (equation 71). The integrand for this is provided by the function `calc.event.like.negtsd`.

7: If the uncertainties in both the measured time and the measured signal are zero (indicated in the code by

having $\sigma_{s_j} = \sigma_{y_j} = 0$ explicitly for that event), then the likelihood calculation again reduces to an analytical calculation without integration (section C). (As this only makes sense when we have a stochastic component, this will not be triggered if TSMod2 = `BypassTSMod2`.) Note that this option works also with the fully stochastic models. The integrand for this is provided by the function `calc.event.like.zerobothsd`.

8: A simple stochastic model is to model the data as showing random Gaussian variations of constant known mean and constant unknown variance. This is achieved by setting
TSMod1 = `FuncUniform`
with its only parameter fixed to the mean of the data

```
theta1Flag = c(offset=FALSE)
theta1Fixed = list(offset=0)
```

(here the mean is assumed to be zero). With

```
TSMod2 = ProbGaussian
theta2Flag = c(sd=TRUE)
```

the evidence is calculated by marginalizing over the variance, the only free parameter.

9: To use fully stochastic time series models (section B), we choose the appropriate model for TSMod2. This automatically forces TSMod1 = `BypassTSMod1`, which is used internally.

10: To model the OU process as described in section B.2, we set TSMod2 = `ProbOUprocess`. This assumes that the time uncertainties are negligible (if given they are ignored) and that the measurement noise model is Gaussian (this overrides any other settings). As the likelihood is calculated without (explicit) integration, the integration limits are ignored. The same is done when modelling the Wiener process using TSMod2 = `WienerOUprocess`. In both cases the likelihood is calculated via the function `calc.like.FullyStochasticprocess`.

Some other possible scenarios are not yet implemented, such as negligible uncertainties only in the signal.

D.4 Priors, fixed parameters and initial values

Every parameter, θ, of every time series model (`TSModX`), has an associated prior PDF. The parameters of this prior PDF are its hyperparameters, $\{\alpha\}$. These are specified in Table 2. For some parameters the hyperparameters are fixed and not settable by the user, e.g. for the phase parameter, ϕ, in `FuncSinusoid`. Otherwise, values for the hyperparameters should be set according to the domain knowledge of the problem.

We can also fix some of the parameters. The values are specified using the `thetaXFixed` variables (`X`=1, 2, 3) in the setup file. These must be set to respect the prior PDF for that parameter. That is, if you fix the parameter to a value which has zero prior, the posterior PDF will of course always be zero (in fact, the likelihood is never even calculated for zero prior parameters).

The initial values of the parameters used in the MCMC samplers, `thetaXInit`, should be set as close to the expected values as possible. But read the advice and warnings in section E.2.

D.5 Choice of sampling and explanation of the main output from ctsmod

ctsmod will sample either form the prior or from the posterior (or both).

To sample the prior, set `calcEvidence==TRUE` and select the number of samples using `Nevsamp`. The result of this is a 2D matrix, `evCalc`, with `Nevsamp` rows and $2 + N_\theta$ columns containing the log likelihood, log prior, and the parameter samples. The rest of this subsection is concerned with posterior

sampling.

The code gives three options for sampling the posterior by MCMC:

- Metropolis. Selected by setting `Nchain=1` and `Nwalker=0`.
- Parallel tempering. Selected by setting `Nchain>1` (the number of chains) and `Nwalker=0`.
- emcee. Selecting by setting `Nchain=0` and `Nwalker>1` (the number of walkers).

All three can be used with a single partition (direct use of the function `sample.posterior()`) or multiple partitions (use of the function `kfold.cv()`). In all cases the variables are transformed as necessary from their natural range to an infinite range, as described in section 5.2. After the MCMC is complete, the variables are transformed back, so the user will only ever see parameters in their natural range.

For all MCMC methods, the function `sample.posterior()` returns a list with two elements, the `postSamp` and `postSampAll`, which contain in some form the result of the MCMC. These are extracted and made available as the arrays `postSamp` and `postSampAll` at the top level of ctsmod for further analysis. `postSamp` is a 2D array, with one row per sample, and with $2 + N_\theta$ columns containing the log likelihood, log prior, and the parameter samples. Exactly what is in the rows of this array and what is in `postSampAll` depends on the MCMC method and is described in the subsections below.

The function `kfold.cv()` returns a list with four elements, which is called `kfoldCV` and made available at the top level of ctsmod for further analysis:

- `multiPostSamp`, which is a 3D array, of which `multiplotSamp[,,k]` is `postSamp` for each partition `k`
- `logLikePart`, a vector containing the log likelihood for each partition
- `partInd`, which is a list, each element of which is a vector of the events in that partition
- `postSampNames`, a character list of the names of the names of the parameters (actually the columns in `postSamp`.

Note that `postSampAll` for each partition is not retained.

Metropolis

A Gaussian covariance matrix is used for the proposal distribution (the "sampling matrix"). The standard deviations – the leading diagonal of this matrix – are specified in the setup file with `sampleSD`. *Note that this is in the units of the transformed variables*, which is actually convenient as for the log transformation this makes them correspond to multiplicative scales. If `sampleCor` is set to be 0 then the sampling matrix is diagonal, otherwise a common covariance between all parameters of that value is assumed. Sampling is initialized using `thetaInit`.

`postSamp` is a 2D matrix containing, for each sample (row), the log prior, log likelihood, and parameter values.
`postSampAll` is `NULL`.

Parallel tempering

The sampling matrix and initialization is set up as just described for the Metropolis algorithm (it is the same for all chains). An equal spacing of the β parameters from 0 to 1 is used.
`postSampAll` is a 3D array, containing the posterior samples (and log prior and log likelihood), for all chains in the order (sample, chain, parameter). The burn-in samples are included.
`postSamp` contains just the cold chain ($\beta = 1$), i.e. samples from the posterior PDF, and excludes the burn-in.

emcee

The walkers are initialized by drawing from a multivariate Gaussian with mean `thetaInit` and covariance matrix constructed in the same way as the covariance matrix for the Metropolis sampling is defined (i.e. using `sampleSD` and `sampleCor`).[19]
`postSampAll` is a 3D array, containing the posterior samples (and log prior and log likelihood), for all walkers in the order (iteration, walker, parameter). The burn-in samples are included.
`postSamp` is just a convenient reformation of this 3D array into a 2D array by combining iteration and walker into a single dimension, and also removes the burn-in. It is projected such that the rows list all walkers for the first iteration, then all for the second iteration, etc.

D.6 Simulated data

The file `ctsmod_genfunctions.R` contains various functions for simulating data from different functions. These are called directly by ctsmod (see section D.2). These functions, named `sim.*` should be self explanatory, and should be edited directly to achieve the desired simulated data.

One of the functions, `sim.quasisinusoid`, generates data from a sinusoidal function with an extra phase component which varies smoothly in time. This is achieved by generating a time series of Gaussian random deviates (zero mean, unit variance) then smoothing this using a locally fitted polynomial, achieved with the `locpoly` function in package `KernSmooth`. This gives the variable phase, $\Psi(t)$ The quasi sinusoidal model is

$$z = b + \frac{a}{2}\big(\cos[2\pi(\nu t + \Psi(t) + \phi)] + I\big) \tag{74}$$

where $I = 0$ if the function is to be centered around zero signal, or $I = 1$ if the minimum value if to be zero. An example of the variable phase and the resulting model is shown in Fig. 10.

Note that this is not the same model as `FuncQuasiSinusoid`, a `TSMod1` model described in section A.2, for which data can be simulated using the function `sim.asfuncquasisinusoid`.

D.7 Analysis and plotting utilities

The file `utilities.R` contains various functions for analysing the results of ctsmod, in particular plotting the results and calculating statistics. This mostly use the outputs `evCalc`, `postSamp`, `postSampAll`

[19] Note that this can produced values for the phase parameter in `FuncSinusoid` outside of the range 0–1, which would give logPosterior of `-Inf` and thus trigger the initialization check error in `emcee()`. Thus values of `thetaInit` and `sampleSD` must be chosen carefully to avoid this, e.g. 0.5 and 0.01 respectively.

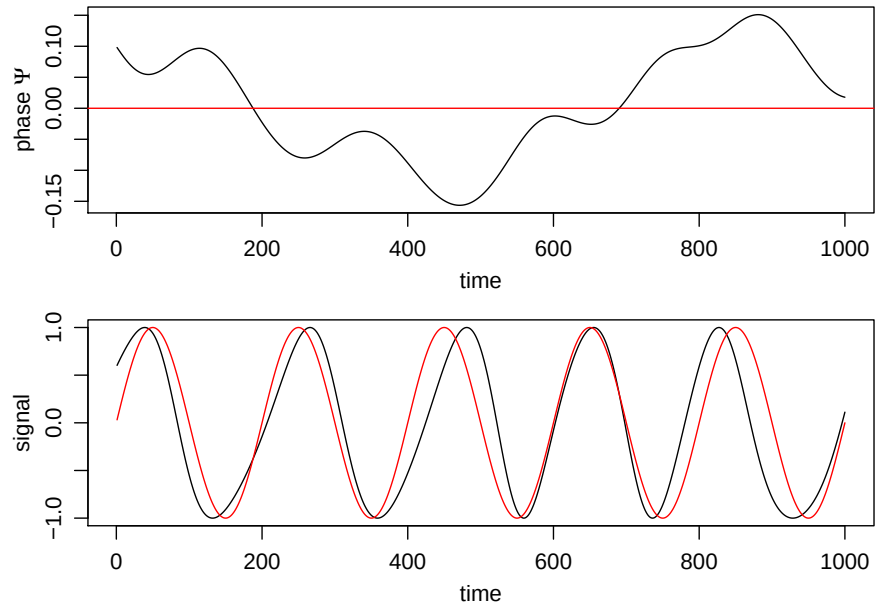

Figure 10: Example of a (noise-free) simulated data set generated with quasi-sinusoidal model (equation 74) with $a = 2$, $b = 0$, $\nu = 1/200$, $\phi = 0$ and $I = 0$. The red curve in the lower panel is the strictly periodic function, i.e. the same model but with $\Psi = 0$.

and/or `kfoldCV`. The functions are documented and should be fairly self-explanatory. Where appropriate, the functions permit specification of a number of parameters:

- removal of samples, `Nrej`. If this is a scalar, then remove this from the beginning of the set of traces. This is generally used to apply a post-doc burn-in. If it is a vector (typically specied using a range, e.g. `8e3:1e4`, then remove these samples. This we might use to remove final samples, to check convergence.[20]
- a thinning factor `thin`[21] and `Nthinblock`.[22]
- the factor h (`hfac`) described in section E.1.
- plotting control, with for example `yzero`, `xlim`, `p`, `pbound`, `prange`, `Ntrace`, `Ndense`, `plotsize`, `mfrow`.
- file name, `fname`.

The main functions in `utilities.R` are as follows. The functions are documented in the code. The file `analyse.R` shows examples of how the various functions can be used.

[20] Recall the way in which the walkers are stored in `postSamp` if emcee has been used: all walkers for the first iteration, then all for the second iteration, etc. Thus in order to remove `N` samples from each walker post-hoc, set `Nrej = N * Nwalker`.

[21] A potential problem with MCMC is that successive samples are correlated. This can result in incorrect inference from the resulting chain, e.g. the PDF being narrower than it really is. This problem is mitigated by using *thinning*, which means preserving only every N^{th} step in the chain.

[22] Thinning is applied to `postSamp` in blocks of this value. This is specifically for emcee: setting `Nthinblock = Nwalker` applies the thinning to each walker separately.

- `plot.data`. Plot the data `obsdata`
- `calc.post.stats`. Calculate statistics (mean, sd, mode) from the function samples[†]
- `calc.post.stats.kfoldcv`. Applies `calc.post.stats` to each partition
- `recalc.like`. Caculate the likelihood, possibly with thinning and `hfac`
- `recalc.kfoldcv.like`. Applies `recalc.like` to each partition
- `find.peak.sol`. Find the parameters at the mode (perhaps over a restricted parameter range)[†]
- `find.peak.sol.kfoldcv`. Applies `find.peak.sol` to each partition
- `calc.xic`. Calculate the AIC and BIC by finding the maximum likelihood
- `plot.postSamp1d`. For posterior sampling, plot the samples vs. iteration as well as the posterior PDF (and optionally the prior PDF) for each parameter, and also calculate the mode of each 1D density estimation[†]
- `plot.chain.acf`. Plot the ACF for each partition for each parameter (for Metropolis, or the chold chain in parallel tempering)
- `plot.pt.chains`. For parallel tempering, plot the evolution of each chain for each parameter
- `plot.emcee.walkers`. For emcee, plot the evolution of each walker for each parameter
- `plot.walker.acf`. For emcee, plot the ACF for each walker for each parameter
- `plot.1dpost`. For prior sampling (evidence calculation), plot the prior and posterior PDFs for each parameter
- `plot.2dpost`. For prior sampling, plot the prior and posterior PDF for two specified parameters as a 2D distribution
- `oplot.model.data`. Plot the data together with the `TSMod1` model at specified parameters
- `oplot.OUprocess.data`. Plot the data together with the predicted values from an OU or Wiener process at specified parameters

[†]`calc.post.stats` and `find.peak.sol` find the global mode of the posterior, the MAP estimate. In contrast, `plot.postSamp1d` calculates 1D density estimates for each parameter, which involves a smoothing and sampling of the density, and then calculates the mode of each of these. This may well give different results. If so, then the latter is probably more reliable.

D.8 Parallel processing

The code makes use of parallel processing in two different ways to parallelize the calculation of the evidence and the K-fold CV likelihood.

For the evidence calculation, it makes use of the `foreach` and `doMC` packages to parallelize the Monte Carlo sampling of the prior. `Nevsamp` is the total number of draws from the prior at which we calculate the likelihood. These are grouped into blocks of size `Nparallel` draws/calculations which are processed in parallel (i.e. the number of blocks is `Nevsamp %/% Nparallel`, where `%/%` indicates integer division).

`Ncores` specifies the number of CPU cores to use. If this is set to 0, then the program attempts to allocate all available cores (but sometimes may not).

The code makes use of the `mcparallel` function in the `parallel` package in order to calculate in parallel the sampling of the K partitions in the K-fold CV likelihood calculation. It seems not possible to specify the number of cores with `mcparallel`, so the K-fold CV likelihood calculation will make use of all available cores, up to K. `mcparallel` is based on the forking processes, which is not supported by Windows. If an unexpected error occurs in one of the partitions, this should be printed and all the partition processing stopped, after which the program should exit cleanly with an error message. This was introduced to deal with potential problems of extreme values (see section E.2)[23], but does not solve the bug of NULL returns described in section D.10.

If the setup file specifies to do calculate both the K-fold CV sampling and the posterior-averaged likelihood, then they are run in parallel using `mcparallel`. The "early exit on error" mechanism just described is not implemented for this fork, so an (unexpected) error in the full posterior sampling will not cause an early stop (see also section D.10).

It appears that the parallelization is only available on a single node. In particular, the methods are not exploitable by MPI, so some parallel processing controllers (such as mpiexec) cannot be used to make ctsmod run across a cluster of nodes.

Warning! These parallel processing packages are not compatible with GUI output, so `ctsmod` should be run in a terminal, and not in a GUI such the Mac R.app.

D.9 Errors and warnings

There is a reasonable but incomplete level of error trapping within the code. This is mostly concerned with detecting illogical combinations of parameter settings or settings which are inconsistent with the data. In such cases error messages are returned, but sometimes it can be difficult to identify exactly where in the code the (same) error message was triggered. The code is not robust to all possible errors in the setup file or data, so the user should always think carefully about that they are doing and not use silly values. In particular, avoid initializing the parameters to their extreme values (see section E.2).

Most of the error trapping is done with the function `stop`. For some reason, this does not always report the expected error message when using running things in parallel. In Messages can be supressed. If you have `Npart>` 1 and `samplePosterior=TRUE` and get an unexplainable crash, then setting `Npart=0` or `samplePosterior=FALSE` and running again should reveal the error message. (See also section D.10). If you get a crash before the code even starts sampling, it is probably due to inconsistent settings in the setup file (which should be revealed in the way suggested).

Evaluations of the functions (`TSMod, MeasMod`) at inappropriate values are not explictly trapped and could cause an error. It is assumed that the priors of the parameters in the `TSMod` functions have been set to values which effectively (due to finite numerical precision) prevent the parameters straying into inappropriate ranges (see section E.2).

There are some specific things in the code which are known would produce errors, although the behaviour

[23]The K-fold CV could be done using `foreach`, and indeed was so up until v21 of the code. However, if an error occurs in one of the workers, the error message is not displayed in the ususal way (side effects are not preserved). Even if we did return an error from a worker, it is not straight forward to use this to terminate all the workers instantly. These errors can remain unnoticed and mess up subsequent processing. The use of `foreach` in the evidence calculation remains vulnerable to this. `mcparallel`, in contrast, has the facility to check the results immediately upon return of any worker, and this is used to stop the other processes.

is unexpected. These are marked with "WARNING" in the code. This includes 0 divided by 0, resulting in a `NaN` which is later required to be numeric. (Note that $1/0$ resulting in `Inf` is allowed to occur in places in the code and there can be handled routinely.)

D.10 Bugs

emcee with FuncQuasiSinusoid

The use of emcee with the `FuncQuasiSinusoid` model can cause an error after a large number of iterations, I think only when running in parallel. Specifically, `sample.posterior()` sometimes return a NULL, either on the full data set or when called by `kfoldcv()` on one or more data partitions. But closer inspection shows that the final value within `sample.posterior()` which is set to be returned is not NULL. So it appears that the code is somehow "loosing" the results upon return from this function, replacing them with a NULL.[24]

This error was not immediately apparent, because if one of the partitions was returned as NULL, the recombination of them into `kfoldCV$multiPostSamp` ignored it, simply recylcing the results from the first partitions to build the matrix up to the specified size. Thus consequence was that two or more partition results were identical, which produced no error as such, but was noticed via inconsistent results (e.g. a small change in the number of iterations made a large change in the K-fold CV likelihood). A NULL return is now trapped and causes a stop with a message.

This NULL return can also occur when calling `sample.posterior()` for calculating the posterior-averaged likelihood. In that case I use this NULL to trigger a warning, but not a stop. `postSamp` and `postSampAll` will be NULL.

This problem became apparent in v20 and persists in v21 and v22. It is reproducible with exactly the same setup. However, small changes, such as adding or removing a small fraction of iterations or changing the number of walkers, can make the error vanish. I have not seen the error with other time series models or using the other MCMC algorithms with this time series model (although it's also possible the error was not noticed.) I have not seen the error in runs with small numbers of iterations, so perhaps it is related to the large values of the transformed variables which can occur with emcee. During the development process I discovered various problems due to extreme values, but all of these are now trapped (see section E.2). As `sample.posterior()` is generally called in parallel – and given the behaviour reported in the footnote earlier – the error must surely be related to the parallelization. Note that we get this error both with `foreach` (in v21) and with `mcparallel` (in v22).

A work-around this bug is to change the number of iterations, walkers or partitions. The error may then not occur.

[24] In one particular case it is the first partition of ten which gives the NULL return. The MCMC algorithm, called in the middle of `sample.posterior()`, returns without problem: the results were printed to a file. But an attempt to write again to a file at the very end of `sample.posterior()` produced no file. However, if the remaining calculations were performed manually (using the first output file), then no error was encountered. Thus the failure to write the second file might simply be because the buffer was not actually written before the program crashed. Even more curiously, if exactly the same program is run but with an extra line of code in `kfoldcv()` causing just the first or the first two partitions to run, then we get no error!

D.11 A very quick tutorial

Run `ctsmod` using the default setup file. This will apply the linear model, `FuncLinear`, to 20 data points drawn from a straight line with true parameters $m = 1, t_0 = 50, z_0 = 0$ (see section A.1) to which Gaussian noise with $\sigma = 10$ has been added to the y values (and nothing to the x values). This uses the MCMC algorithm emcee with 100 walkers and 200 iterations (which is too small; this is just for illustration) to sample the posterior and calculate the posterior-averaged likelihood. It should run in 1–2 minutes.

To plot the resulting 1D PDFs use

```
plot.postSamp1d(postSamp, priorSamp=NULL, Nrej=0, thin=1, Nthinblock=1, Ntrace=200,
Ndense=2^12, mfrow=c(4,2), plotsize=c(5,7), xlim=list(NULL, NULL, NULL, c(0,50)) )
```

which also prints the mode of the 1D density estimates of the posterior PDF for each parameter. Using these (or any other) parameters you can then plot the data, overplotted with this model

```
oplot.model.data(TSMod1class="FuncLinear", obsdata, paramlist=list(slope=0.9,
tmid=42, offset=-10, sd=5, priorform="Gaussian"))
```

If you want to see the traces of the individual walkers, use

```
plot.emcee.walkers(postSampAll, thetaFlag=thetaFlag, Ntrace=NA, mfcol=c(4,2),
fname="emcee_walkers.pdf")
```

There are several other analysis functions in the file `utilities.R`, summarized in section D.7. They, and the above functions, have various parameters which permit control of the plotting range, thinning etc. (see the documentation in the code).

E Finite numerical precision

E.1 Very small probabilities

The likelihood is typically a very small number, and once small enough (below about 10^{-320} in R) is truncated to zero due to finite numerical precision. `ctsmod` avoids this problem to some degree by calculating the logarithm of the likelihood (equal to the sum of the logarithms of the event likelihoods: see equation 8). But to calculate the evidence (via equation 24) we must exponentiate these values, and this again will frequently run into the numerical precision limitation, giving zero evidence. We can avoid this using a help factor, h. Writing the numerically approximated evidence as E [$\equiv P(D|\sigma, M)$], and the likelihood at parameter sample θ_n as L_n [$\equiv P(D|\sigma, \theta_n, M)$], then we can we can write

$$E = \frac{1}{N} \sum_{n=1}^{n=N} 10^{\log L_n} . \tag{75}$$

If we now add a constant h to every $\log L_n$ term, then provided the dynamic range of all L_n is less than about 2×320 orders of magnitude, we can choose h to ensure that $10^{(h + \log L_n)}$ is not truncated. This will then give us the modified evidence

$$\begin{aligned} E' &= \frac{1}{N} \sum_{n=1}^{n=N} 10^{(h + \log L_n)} \\ &= 10^h E \end{aligned} \tag{76}$$

from which we can trivially calculate E. Equivalently $\log E' = h + \log E$. We use the same principle in the calculation of the K-fold CV likelihood.

A reasonably good choice of h is $h = -\text{median}[\log L_n]$, although this does not guarantee avoidance of numerical problems. The term h is called `hfac` in the `ctsmod` code. It is used in the main code (with this suggested value) to calculate the evidence and K-fold CV likelihood, and can also be specified in some functions in `utilities.R` to recalculate these.

E.2 Extreme parameters

Extreme values of the transformed parameters

The MCMC algorithms sample the parameter space by retaining or rejecting a proposed sample based on the value of the log posterior probability of the proposed sample compared to the present one. This sampling is done over an infinite space, so parameters which actually have a finite range (such as frequency) are normally inverse transformed after the sampling (see section 5.2). Due to the finite precision of the computer, some of these transformations can cause numerical errors at the extreme range of the parameters. To avoid this, some parameters are truncated and/or warnings are given (see section D.9. But more generally, it is possible that the MCMC gives rise to a parameter value which is so extreme that the function evaluation gives `NaN`. This arises in the cosine function in `FuncSinusoid` (and its derivatives) when the frequency is very large, for example. Rather than trying to trap such all such extreme values explicitly (and at some pre-defined threshold), I assume that the prior PDF has been designed to give a value numerically identical (in terms of machine precision) to zero in such cases. If `logPrior = -Inf`, then I avoid evaluating the function at all, and set `logLike = -Inf`. This is acceptable, because (a) the likelihood is only calculated[25] when the prior is, and (b) `logPrior = -Inf` will anyway result in the MCMC proposal being rejected, so there is no point in calculating the likelihood anyway.

Problem with emcee

The emcee algorithm (section 5.4) is not entirely robust to use of transformed parameters. Inspection of the parameter proposal equation (equation 40) shows that the (transformed) parameters can become very large (postive or negative). Sometimes such extreme parameters will be rejected by the prior. For example, if a very large positive frequency in `FuncSinusoid` is proposed, its (gamma) prior will be negligibly small, eventually numerically identical to zero, in which case the proposal will be rejected by any of the MCMC algorithms. However, a very large negative value of the (transformed) frequency may have a non-zero prior probability density (e.g. with shape=1 in the gamma prior), as this just corresponds to a very small value of the frequency. But if this is selected to update another walker, then equation 40 shows that this other walker will be assigned a very large *positive* proposed frequency. That would be rejected by the prior, but is still not desirable (as it will lower the acceptance rate). More critical, though, is that equation 40 can produce a numerical error, because if θ_w has a reasonable value, but $\theta_x = -10^{+320}$ for example, then

$$Y = -10^{+320} + z(\theta_w + 10^{+320}) \simeq -10^{+320} + 10^{+320} = \texttt{NaN} \tag{77}$$

as $|z\theta_w| \ll 10^{+320}$. This value causes a crash when we try to evaluate the prior or the function (this has been observed in practice). This is an unavoidable result of using emcee with transformed parameters which are permitted to have very large (positive or negative) values. (The problem is that transformed values are not explicitly suppressed by the prior.) In practice it should not occur often[26], but to be safe the implementation of emcee checks for non-finite proposed parameters, and rejects them (with a warning) if they occur.

[25] Calculation of the DIC does not involve the prior, but this is not part of the MCMC, and it (`calc.dic()`) has anyway been removed from `run_ctsmod` from v19.

[26] Yet it did in v19 when the circular transformation was being used with phase, which resulted in sinamp and freq as well as phase growing to very large negative values of the transformed parameters for some walkers

Parameter initialization and priors

Generally speaking, one should not initialize the parameters of the models to their extreme values. This will generally result in the log prior PDF being minus infinity and the sampling would never get going. For the same reason, one should not initialize the parameters outside of their defined range (see Table 2). While this is obvious for most parameters (such as frequency), it may be less so for some others. In particular, do not initialize the parameters `phase` or `fpX` (`X`$= 1, 2, 3, 4$) of `FuncSinusoid` and `FuncQuasiSinusoid` to their extreme values, which are $0, 1$ for `phase` and $-1, +1$ for `fpX`.

Likewise, fixed parameters must not be fixed to values which give zero prior PDF.

The user is advised to adopt values of the hyperparameters of the priors to values which supress anything unrealistic. The gamma prior on frequency, for example, naturally supresses any very large frequencies, but with shape=1, arbitrarily small frequencies are possible. While this should not cause any code crash, it may produce arbitrarily small and therefore useless frequencies (as such a model is just a constant), possibly resulting in pointless sampling (or a low acceptance rate). A larger value of the shape parameter is therefore recommended.

F Other methods for calculating the evidence

Section 5.1 outlined the methods used in `ctsmod` to calculate the evidence. Some other methods are outlined here.

Sample the posterior (harmonic mean of likelihoods). Taking the idea of importance sampling, it can be shown (e.g. Kass & Raftery 1995, section 4.3) that given a set of samples $\{\theta_n\}$ drawn from the posterior, the evidence can be approximated as

$$P(D|\sigma, M) \approx \left(\frac{1}{N} \sum_{n=1}^{n=N} P(D|\sigma, \theta_n, M)^{-1} \right)^{-1} \tag{78}$$

the harmonic mean of the likelihood at these samples. However, this method is widely criticized in the literature because it is very unstable, and only converges to the true evidence in most cases with an impractically large number of samples. It has been shown to give highly inaccurate results, and is not recommended.

Note that while sampling from prior is generally easy, sampling from the posterior is generally hard, and MCMC methods are usually employed for this.

Laplace's method. If we believe the posterior to be dominated by a single peak (the posterior mode), then a Taylor expansion of the (log) unnormalized posterior up to the quadratic term gives an approximation for this which is just a Gaussian with a mean equal to the posterior mode, and covariance equal to the inverse of the Hessian matrix of the second derivatives (e.g. Kass & Raftery 1995, section 4.1). The integral is analytic and requires evaluation of the likelihood only at the posterior mode (although of course this mode must be located by an optimization algorithm, which will involve many evaluations of the likelihood). This approximation may be valid in the case of informative data (large samples).

Chib's method. In principle the evidence may be evaluated from a few evaluations of the posterior, prior and likelihood at well-chosen values of the parameters. The posterior is estimated via Gibbs sampling. See Chib (1995) or Friel & Pettitt (2008) for an overview and other references.

Savage-Dickey density ratio. If we have nested models (i.e. where one model is just a special case of the other model), and the priors are separable, then the Bayes factor for this pair of models can be calculated much more simply. The simpler model is just the more complex model with one or more parameters fixed to certain values. The SDDR involves calculating the marginal (normalized) posterior of the more complex model at these fixed values. Trotta (2007) summarizes the method and gives a cosmological application.

Annealed importance sampling. This method is based on importance sampling and the use of Markov chains. The importance weights found in the process can be used to estimate the evidence. See Neal (2001).

Reversible Jump MCMC. The goal of this method is to estimate not only the evidence for a single model, but also the posterior probabilities of the models. The method was introduced by Green (1995).

Nested sampling. This method, introduced by Skilling (2004), attempts to overcome some issues with thermodynamic integration. It calculates not only the evidence but also posterior samples. There are several articles in the astronomical literature which describe and use this, or variations thereof, one of which is Feroz & Hobson (2008).

G References

Bailer-Jones C.A.L., 2009, *The evidence for and against astronomical impacts on climate change and mass extinctions: A review*, International Journal of Astrobiology 8, 213

Bailer-Jones C.A.L., 2011, *Bayesian time series analysis of terrestrial impact cratering*, MNRAS 416, 1163 Link to article. Erratum: MNRAS 418, 2111 (2011)

Bailer-Jones C.A.L., 2012, *A Bayesian method for the analysis of deterministic and stochastic time series*, A&A 2012, in press Link to article

Berliner L.M., 1996, *Hierarchical Bayesian time series analysis*, in Hanson K. & Silver R.N. (eds), in Fundamental Theories of Physics, Vol. 79, Maximum Entropy & Bayesian Methods: Proceedings of the 15th International Workshop, Santa Fe, NM, 1995

Brockwell P.J., Davis R.A., 2002, *Introduction to time series and forecasting*, Springer, 2nd edition,

Chatfield C., 1996, *The analysis of time series*, 5th ed., Chapman & Hall, London

Chib S., 1995, *Marginal likelihood from the Gibbs output*, Journal of the American Statistical Association, 90, 1313

Christensen R., 2005, The American Statistician, 59, 121

Earl D.J., Deem M.W., 2005, *Parallel tempering: Theory, applications, and new perspectives*, Phys. Chem. Chem. Phys., 7, 3910

Feroz F., Hobson M.P., 2008, *Multimodal nested sampling: an efficient and robust alternative to Markov Chain Monte Carlo methods for astronomical data analyses*, MNRAS 384, 449

Foreman-Mackey D., et al. 2013, *emcee: The MCMC Hammer*, arXiv:1202.3665v3

Friel N., Pettitt A.N., 2008, *Marginal Likelihood estimation via power posteriors*, Journal of the Royal Statistical Society B, 70, 589

Gillepsie D.T., 1996, *Exact numerical simulation of the Ornstein-Uhlenbeck process and its integral*, Phys. Rev. E, 54, 2084

Gillepsie D.T., 1996b, *The mathematics of Brownian motion and Johnson noise*, Am. J. Phys. 64, 225

Goodman J., Weare J., 2010, *Ensemble samplers with affine invariance*, Comm. App. Math. and Comp. Sci., 5, 65

Gregory P.C., 2005, *A bayesian analysis of extrasolar planet data for HD 73526*, ApJ, 631, 1198

Green P. J., 1995, *Reversible jump Markov chain Monte Carlo computation and Bayesian model determination*, Biometrika 82, 711

Hahn T., 2006, *Cuba: a library for multidimensional numerical integration*, arXiv:hep-ph/0404043v2

Jaynes E.T., 2003, *Probability theory. The logic of science*, Cambridge University Press

Jones R.H., 1986, *Time series regression with unequally spaced data*, J. Applied Probability, 23, 89

Kadane J.B., Lazar N.A., 2004, *Methods and criteria for model selection*, Journal of the American Statistical Association, 99, 465

Kass R., Raftery A., 1995, *Bayes factors*, Journal of the American Statistical Association, 90, 773

Kelly B.C., et al., 2009, *Are the variations in quasar optical flux drived by thermal fluctuations?*, ApJ 698, 895

Kozlowski S., et al., 2010, *Quantifying quasar variability as part of a general approach to classifying continuously varying sources*, ApJ 708, 927

Lartillot N., Philippe H., 2006, *Computing Bayes factors using thermodynamic integration*, Systematic Biology 55, 195

Neal R.M., 2001, *Annealed importance sampling*, Statistics and Computing 11, 125

Skilling J., 2004, in Fisches R., Preuss R., von Toussaint U. (eds), AIP Conf. Proc. Vol. 735, Bayesian Inference and Maximum Entropy Methods in Science and Engineering, p. 395

Spiegelhalter D.J., Best N.G., Carlin B.P., van der Linde A., 2002, *Bayesian measures of model complexity and fit*, J. R. Statist. Soc. B, 64, 583

Trotta, R., 2007, *Applications of Bayesian model selection to cosmological parameters*, MNRAS 378, 72

Uhlenbeck G.E., Ornstein L.S., 1930, *On the theory of the Brownian motion*, Phys. Rev. 36, 823

Vehtari A., Lampinen J., 2002 *Bayesian model assessment and comparison using cross-validation predictive densities*, Neural Computation 14, 2439