Ice Core Dating using Probabilistic Programming

Aditya Ravuri
University of Cambridge
ar847@cam.ac.uk

Tom R. Andersson
British Antarctic Survey
tomand@bas.ac.uk

Ieva Kazlauskaite
University of Cambridge
British Antarctic Survey
ik394@cam.ac.uk

Will Tebbutt
University of Cambridge
wct23@cam.ac.uk

Richard E. Turner
University of Cambridge
ret26@cam.ac.uk

J. Scott Hosking
British Antarctic Survey
The Alan Turing Institute
jask@bas.ac.uk

Neil D. Lawrence
University of Cambridge
ndl21@cam.ac.uk

Markus Kaiser
University of Cambridge
British Antarctic Survey
mk2092@cam.ac.uk

Abstract

Ice cores record crucial information about past climate. However, before ice core data can have scientific value, the chronology must be inferred by estimating the age as a function of depth. Under certain conditions, chemicals locked in the ice display quasi-periodic cycles that delineate annual layers. Manually counting these noisy seasonal patterns to infer the chronology can be an imperfect and time-consuming process, and does not capture uncertainty in a principled fashion. In addition, several ice cores may be collected from a region, introducing an aspect of spatial correlation between them. We present an exploration of the use of probabilistic models for automatic dating of ice cores, using probabilistic programming to showcase its use for prototyping, automatic inference and maintainability, and demonstrate common failure modes of these tools.

1 Introduction

Chemicals in the atmosphere are deposited onto ice sheets through precipitation, with further deposition burying and eventually compacting the snow into solid ice, recording the chemical composition of the atmosphere. These chemicals provide evidence for the climate of the past and are known as *proxy variables*. Annual cycles can be present in the data if 1) the abundance of that proxy varies seasonally, 2) the precipitation rate at the ice core site is large enough, and 3) the depth is not so great that the annual layers have been excessively compressed. Given these conditions, annual layer thickness is dictated by the amount of annual precipitation (a random component) and compression of the ice with increasing depth (a systematic component). A section of the data that we used, the Jurassic ice core from Emanuelsson et al. [2022], is shown in Figure 1.

Constructing an ice core's timescale manually through layer counting can be an arduous process, lasting from days to years of person-time (Winstrup 2016). Manual counting also has the disadvantage of poorly-quantified uncertainty in the depth to time mapping, based on heuristics from expert disagreement or uncertain layers. Probabilistic inference leads to a more principled treatment of such uncertainties. A probabilistic approach also enables prior knowledge to be incorporated during ice core timescale inference. For example, some observations for time at certain depths may be available

2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems.

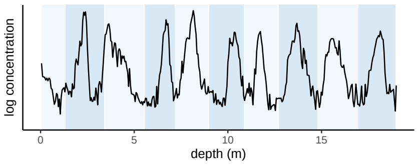

Figure 1: Section of the data, showing annual seasonality in MSA (a proxy exhibiting a strong seasonal pattern) plotted against depth. Manually counted annual layers are overlaid as shaded bars.

due to known volcanic events depositing a layer of ash in the ice (these depth-time observations are known as tie-points).

Our contribution is to present a case study in developing models for the ice core dating problem using a Probabilistic Programming Language (PPL), mainly Stan [Stan Development Team, 2022] alongside torchsde [Li et al., 2020], and TensorFlow Probability [Dillon et al., 2017]. Our work highlights such tools to practitioners at the intersection of probabilistic machine learning and climate science. We aim to show the promise of PPLs in how they can enable the composition of assumptions, easy experimentation, model extension and maintainability. A core promise of widely used PPLs is that they automate and abstract away inference details, thus ensuring that models are written at the right level of abstraction without the need to write and maintain complex inference algorithms. We also show current limitations of these tools, such as inference only being computationally feasible over a limited set of models that may be suitable for a task, and inference methods being limited in practice depending on the specific functional requirements set out for each PPL.

2 Problem Setup

The proxy variables are measured in ice cores along a depth dimension. We represent the depth series as a set of known random variables $\boldsymbol{\delta} := \{\delta_i\}_{i=1}^{n}, \delta_i \in \mathbb{R}^+$ where n is the number of sampled depths at which proxy readings are available. The depth series $\boldsymbol{\delta}$ is mapped to latent time (age) values associated with each observation, represented by the stochastic process $\mathbf{t} := \{t_{\delta_i}\}_{i=1}^{n}$ indexed by the depth series. Proxy measurements are denoted as $\mathbf{s} := \{s_{\delta_i}\}_{i=1}^{n}$, with $s_{\delta_i} \in \mathbb{R}$. The *ice core dating* process can be stated as an inference problem for time conditioned on the proxy and depth data, $\mathbf{t}|\mathbf{s}, \boldsymbol{\delta}$. Within this work, we assume that proxies depend only on the latent time process, the depth to time mapping is monotonic, and that there is only one proxy available with a clear seasonal signal. This gives rise to the class of models shown in Fig. 4 (a), in the Appendix.

3 Methodology

The setup of the problem gives rise to a class of models that vary in their assumptions and complexity from discrete-index HMMs to continuous-index HMMs to SDE-based models. Some of these models, particularly HMMs, were previously studied by Winstrup [2011, 2016] who explored the the effect of batching, the usage of other observation models and extensions to hidden semi-Markov models (for allowing for priors to be set over lengths of year boundaries) and inference therein.

3.1 A Hidden Markov Model

Model Under the assumption that the depth-sampling is uniform, and that the latent time process has a discrete domain, we can model the latent time process as a Markov chain. If in addition, the proxies are conditionally independent given the time periods they correspond to, the framework in Figure 4 (a) reduces to a Hidden Markov Model shown in Figure 4 (b). We assume that $\boldsymbol{t}$ can occupy states $\forall i : t_{\delta_i} \in \{{}^{k}/_{n_s}\}_{k=1}^{m \cdot n_s}$, where n_s is the number of states within each yearly cycle and m is an arbitrarily large number of years. The transition matrix

$$\mathbb{P}(t_{\delta_i}|t_{\delta_{i-1}}) = \begin{bmatrix} p_{1/n_s} & 1-p_{1/n_s} & 0 & \cdots & 0 \\ 0 & p_{2/n_s} & 1-p_{2/n_s} & \cdots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

is bidiagonal, with p_i denoting the probability that the chain stays in state i given that it was in the same state at the last depth measurement. In other words, as depth increases, the time process either advances or stays the same. This enforces monotonicity of the latent time sequence w.r.t. depth and also allows one to track what year an observation corresponds to (given by the floor of the state, $\lfloor i \rfloor$). The parameters for each state within a year are repeated across years to constrain the model (i.e. $p_{c+k/n_s} = p_{k/n_s}$ for $c \in \mathbb{N}$). The following observation model is used as part of the HMM,

$$\forall i : s_{\delta_i} | t_{\delta_i} \sim \mathcal{N}(a \cos(2\pi t_{\delta_i}) + b, \sigma^2).$$

Note that, as the transition matrix parameters are not constant within each annual layer, the model's annual layer shapes can be a warping of the mean cosine function in the observation model, enabling some flexibility in modelling the real proxy cycle shapes.

Inference Using the Stan language, we perform maximum likelihood inference for the parameters given data with time marginalised $(a, b, \sigma, \{p\}_j | \mathbf{s})$, with the posterior over times $\mathbf{t}|\mathbf{s}, a, b, \sigma, \{p\}_j$ estimated using the forward-backward algorithm in Stan. The probabilistic program is shown in Appendix B. Two problems are encountered with standard tooling:

- The runtimes increase quadratically w.r.t. the state space (due to the forward algorithm, which computes the log likelihood with hidden states marginalised). We remedy this by rewriting the forward algorithm exploiting the sparsity of our transition matrix, providing a direct replacement of the native Stan function (shown in Appendix B). In a PPL, users can typically implement their own efficient functions for such likelihood calculations.
- The model is misspecified as the data is expected to be non-stationary (due to compression of the ice core and temporal variation in precipitation). This can pragmatically be remedied by processing the data in batches, allowing the parameters such as a and b to change between different sections of the ice core.

Inference using this implementation takes 2.5 minutes using a single-thread run, without GPU utilization, for the entire ice core (about 2.5k observations), without requiring any special initialization strategies.

3.2 An extension allowing for tie-point specification

To allow for tie-points to be integrated, which constrain the depth-to-time mapping, the observation model is extended to account for non-stationarity in the signal, because accounting for tie-points would require processing data in batches large enough to cover the tie-points. This is done by changing the observation model to allow for parameters a, b to change with each data point along the ice core,

$$\forall i : s_{\delta_i} | t_{\delta_i} \sim \mathcal{N}(a_i \cos(2\pi t_{\delta_i}) + b_i, \sigma^2),$$

with a prior (as part of a hierarchical model) placed over a_i, b_i (thus, the change is "slow"/constrained). A similar hierarchical treatment is given to parameters of the transition matrix, using the prior

$$\forall j \in 1/n_s, ..., m : p_j \sim \text{Beta}(\alpha_{j*n_s \bmod n_s}, \beta_{j*n_s \bmod n_s}).$$

This allows the transition probabilities to change over years (and hence depths). The hierarchical distribution of states at the same point in any yearly cycle however share the same parameters, providing some constraint.

Having extended the model to different observation models per data-point, we specify volcanic tie-points using an alternative observation model using the state directly rather than proxy information. We use a categorical distribution ensuring that the time states must reach the volcanic tie-points,

$$p(s'_{\delta_{\text{tie}}} | t_{\delta_{\text{tie}}}) = \begin{cases} 1/n_s & \text{if } \lfloor t_{\delta_{\text{tie}}} \rfloor = t_{\text{tie}} \\ 0 & \text{otherwise} \end{cases},$$

to enforce that the volcanic ash observed at depth δ_{tie} corresponds to a known year of the eruption t_{tie}.

Maximum likelihood inference for parameters such as a_i, b_i in such models produces subpar results, due to posterior modes of such unidentifiable models lying outside their typical sets, raising a need to integrate out the parameters. As MCMC is computationally infeasible due to the number of

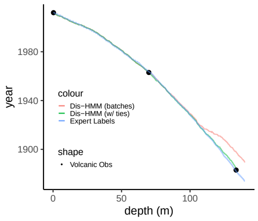

Figure 2: A comparison between the inference results, i.e. sample paths of $\mathbf{t}|\mathbf{s}$, obtained using Stan. These correspond to models presented in sections 3.1 and 3.2, with expert labels overlaid.

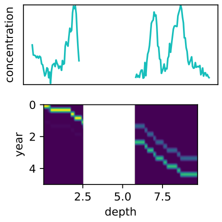

Figure 3: On top, a short data series with a missing section. At the bottom, posterior paths of time inferred using the data above - the missing section induces multimodality in the depth to time posterior.

parameters and data, we use mean-field variational inference (a seamless change in the Stan API). This mode of inference greatly increases runtimes w.r.t. the model in Section 3.1 however, to about a few hours due to the increased number of parameters. Results of inference in the two models presented thus far is shown in Figure 2.

3.3 An extension to continuous-index Hidden Markov Models

Depth measurements may not be regularly spaced, due to missing sections of an ice core as a result of the extraction process, or due to the sampling frequency changing over its depth. In this case, modelling the latent time process as a continuous-index Markov chain leads to the times observed at irregularly spaced points being described by an index-(depth) inhomogeneous Markov chain. Here, the transition matrix associated with a step from δ_{i-1} and δ_i can be computed via the transition rate matrix $\mathbf{Q}$ with the same sparsity structure as Equation 1,

$$\mathbb{P}(t_{\delta_i}|t_{\delta_{i-1}}) = \exp_{\text{matrix}}((\delta_i - \delta_{i-1})\mathbf{Q}). \tag{2}$$

Such models are termed *continuous-time HMMs* [Liu et al., 2015] (although we use the terminology continuous-index to avoid confusion as time is not the index in our application). They allow a better representation of posterior uncertainty arising from missing observations, as seen in Fig. 3. Inference was performed on small datasets by using tensorflow-probability for computing log likelihoods of HMMs in a differentiable manner. High-level pseudocode for MLE/VI when using such tools is shown in Appendix E. Inference involving larger datasets would involve bespoke inference code due to a lack of functionality around efficient computation of the forward algorithm involving matrix exponentials within the PPLs considered.

3.4 An exploration of Stochastic Differential Equations for ice core dating

Assuming instead that we are working with a continuous index and a continuous state space for $\mathbf{t}$, the class of models under consideration naturally extends to state space/ SDE models. Given an index δ, a prior over a stochastic process t_δ can be formulated as,

$$\begin{bmatrix} d\mathbf{z}_\delta \\ dt_\delta \end{bmatrix} = \begin{bmatrix} \mu(\mathbf{z}_\delta, \delta) \\ -\exp(\mu'(\mathbf{z}_\delta, t_\delta, \delta)) \end{bmatrix} d\delta + \begin{bmatrix} \Sigma \\ \epsilon \end{bmatrix} d\mathbf{W}_\delta,$$

with $\epsilon \to 0$ enforcing monotonicity of sample paths and where $\mathbf{z}_\delta$ is a latent process that, for example, can have a GP prior represented as an SDE [Särkkä and Solin, 2019]. Such a prior is similar to the one used in Ustyuzhaninov et al. [2020]. Inference in this class of models can be performed by specifying a variational SDE using the same diffusion but with a different drift [Li et al., 2020]. As in the case above, due to customised variational inference not being supported in Stan, we use functionality from torchsde (differentiable solvers) that enable computation of the objective.

Inference in this class of models was particularly difficult, mirroring the findings of GPCore [Andersson, 2019], where a GP prior is placed on $\mathbf{t}|\boldsymbol{\delta}$, and where maximum likelihood inference is performed for $\mathbf{t}|\mathbf{s}, \mathbf{d}$, in a constrained manner to ensure monotonicity in the depth-time sample paths. The inference is very dependent on good initialisation; using existing estimates of the chronology in the case of GPCore, and using inverse Lomb-Scargle spectrograms for $\mathbf{z}_\delta$ in our SDE models, following ideas from Gay et al. [2014]. However, we had greater difficulties due to local minima than in GPCore. More discussion on this topic can be found in Appendix F.

4 Conclusion

In this paper we presented a taxonomy of models for ice core dating, showing how simplifying assumptions lead naturally to HMMs and how these models, and corresponding probabilistic programs can be extended. We exemplified the shortcomings of PPLs as the models are extended towards SDEs and discussed difficulties with inference in such models. Future work would involve extension of the models presented to utilize multiple proxies, account for spatial correlation between ice cores, and an exploration to determine how to aid inference in latent SDE models for such modelling tasks.

Data and Code

Code to reproduce key results in this paper can be found at https://github.com/infprobscix/icecores.

Acknowledgements

AR is supported by a studentship from the Accelerate Programme for Scientific Discovery. IK was funded by a Biometrika Fellowship awarded by the Biometrika Trust. The authors would also like to thank Liz Thomas and Dieter Tetzner from the British Antarctic Survey for providing the datasets we used, for their insight and helpful discussions on ice core dating task.

References

Tom Andersson. GPCore: A Gaussian process approach for inferring ice core chronologies. Master's thesis, University of Cambridge, 2019.

David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. Tensorflow distributions, 2017.

B. Daniel Emanuelsson, Elizabeth R. Thomas, Dieter R. Tetzner, Jack D. Humby, and Diana O. Vladimirova. Ice core chronologies from the antarctic peninsula: The palmer, jurassic, and rendezvous age-scales. *Geosciences*, 12(2), 2022.

M. Gay, M. De Angelis, and J.-L. Lacoume. Dating a tropical ice core by time–frequency analysis of ion concentration depth profiles. *Climate of the Past*, 10(5), 2014.

James Hensman, Nicolas Durrande, and Arno Solin. Variational fourier features for gaussian processes, 2016.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations, 2020.

Yu-Ying Liu, Shuang Li, Fuxin Li, Le Song, and James M. Rehg. Efficient learning of continuous-time hidden markov models for disease progression. *Advances in Neural Information Processing Systems*, 28:3599–3607, 2015.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. 2012.

Stan Development Team. Stan modeling language users guide and reference manual 2.29, 2022. URL `https://mc-stan.org`.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. 2019.

Ivan Ustyuzhaninov, Ieva Kazlauskaite, Carl Henrik Ek, and Neill Campbell. Monotonic Gaussian process flows. In *International Conference on Artificial Intelligence and Statistics*, pages 3057–3067. PMLR, 2020.

Mai Winstrup. *An Automated Method for Annual Layer Counting in Ice Cores*. PhD thesis, University of Copenhagen, 2011.

Mai Winstrup. A hidden markov model approach to infer timescales for high-resolution climate archives. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 4053–4060. AAAI Press, 2016.

A Appendix: Supplementary Figures

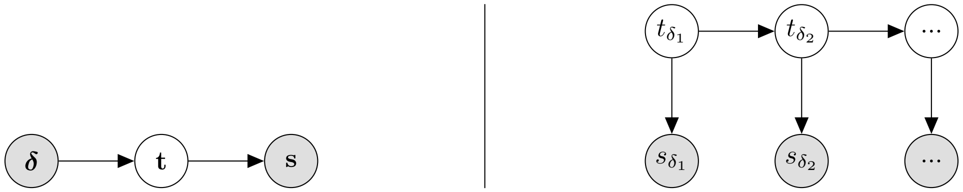

Figure 4: (a.) On the left, a graph that summarises the class of models under consideration. (b.) On the right, a graph representing a hidden Markov model obtained when assumptions of the form in Section 3.1 are made.

B Appendix: Discrete-index HMMs in Stan

Probabilistic programs in Stan are composed as blocks corresponding to,

- data and transformed data: where users can input / specify random variables for which observations have been made and other fixed values,
- parameters and transformed parameters: where users specify random variables on which inference is performed,
- model: where users specify distributional assumptions on and between the random variables defined,
- generated quantities: where downstream analysis can be computed as a function of posterior draws of the parameters conditioned on data.

Extensive documentation on example models in the Stan language, on functions and the language can be found here

We define the data blocks in Stan as follows, for all use cases.

```
data {
    int n;  // num data
    int s;  // num states per year
    int num_years;
    vector[n] depth;  // depth data
    vector[n] y;  // proxy data
    vector[s * num_years] initial_probs;
}
transformed data {
    int n_st = s * num_years;  // total number of states
    vector[s] year_fractions;
    year_fractions = cumulative_sum(rep_vector(1.0/s, s));
    simplex[n_st] rho = initial_probs + 1e-10;
    rho = rho/sum(rho);
}
```

A simple HMM in Stan, corresponding to the model defined in Section 3.1, would consist of parameters,

```
parameters {
    vector<lower=0, upper=1>[s] p_diag;
    real<lower=-3, upper=3> mu;
    real<lower=0, upper=1> sigma;
    real<lower=0, upper=2> scale;
}
```

on which inference will occur. Following the stan syntax on how HMMs are defined, we define `log_omega` corresponding to our observation model, and other variables as follows,

```
transformed parameters {
    vector[s] cosine_term = cos(2.0*pi()*(year_fractions + 0.5));
    matrix[n_st, n] log_omega;

    for(i in 1:num_years){
        for(j in 1:s) {
            for(k in 1:n) {
                log_omega[(i-1)*s + j, k] =
                      normal_lpdf(y[k] | mu + cosine_term[j]*scale, sigma);
            }
        }
    }
}
```

Note that we create these variables in the transformed parameters section instead of the model section (which would be more Stan like, as it would be easier to read the observation model in the model section) to be able to access these variables in the generated quantities block without having to redefine them. It's an unusual case syntactical case as that the parameters governing the likelihood of the HMM with hidden states marginalised is a function of the observation model likelihood. Nevertheless, the observation model is specified in the line,

```
normal_lpdf(y[k] | mu + cosine_term[j]*scale, sigma);
```

The model then simply specifies the log posterior, which is a sum of priors over our parameters (implicitly assumed to be uniform as they're not specified) and the likelihood of

$$\mathrm{s}|\mu, \sigma, \text{scale and transition matrix diagonal}$$

which is specified as,

```
model {
    matrix[n_st, n_st] p_full = diag_trans_to_full(tile(p_diag, num_years));
    target += hmm_marginal(log_omega, p_full, rho);
}
```

where `target` corresponds to the log posterior.

As we're interested in the posterior probabilities (for analysis and to initialize the inital state probabilities in an iterative manner if this model is used on sequential batches of data) and sample paths, we generate the following quantities,

```
generated quantities {
    matrix[n_st, n_st] p_full = diag_trans_to_full(tile(p_diag, num_years));
    matrix[n_st, n] posterior = hmm_hidden_state_prob(log_omega, p_full, rho);
    array[n] int sampled_states = hmm_latent_rng(log_omega, p_full, rho);
}
```

for post-hoc analysis.

As the forward algorithm here, i.e. the function `hmm_marginal` is inefficient for our use case, we can define a more efficient function for bidiagonal transition matrices as follows. A discussion on the computation of the forward algorithm can be found in Barber [2012], Murphy [2012].

```
functions {
    real hmm_marginal_banded(matrix log_omega,
                             vector Gamma_diag,
                             vector rho) {
```

```
        int K = dims(log_omega)[1];
        int N = dims(log_omega)[2];

        vector[K] log_alpha;
        vector[K] inner_sum;
        vector[2] inner_vec;
        vector[K] log_Gamma_diag = log(Gamma_diag);
        vector[K] log_1mGamma_diag = log1m(Gamma_diag);

        int min_i; int max_i;

        log_alpha = log_omega[, 1] + log(rho);

        if (N > 1) {
            for (n in 2:N) {
                for (i in 1:K) {
                    if (i == 1) {
                        inner_sum[i] = log_alpha[i] + log_Gamma_diag[i];
                    } else {
                        inner_vec[1] = log_alpha[i - 1] + log_1mGamma_diag[i - 1];
                        inner_vec[2] = log_alpha[i] + log_Gamma_diag[i];
                        inner_sum[i] = log_sum_exp(inner_vec);
                    }
                }
                log_alpha = log_omega[, n] + inner_sum;
            }
        }

        return log_sum_exp(log_alpha);
    }
}
```

Then, the model changes to,

```
model {
    target += hmm_marginal_banded(log_omega, tile(p_diag, num_years), rho);
}
```

Other convenience functions we define are as follows,

```
functions {
    matrix diag_trans_to_full(vector trans_mat_diag) {
        int n = size(trans_mat_diag);
        matrix[n, n] full_mat = diag_matrix(trans_mat_diag);

        for (i in 1:(n - 1)) {
            full_mat[i, i + 1] = 1 - trans_mat_diag[i];
        }
        full_mat[n, n] = 1.0 - 1e-6;
        full_mat[n, 1] = 1e-6;

        return full_mat;
    }
    vector tile(vector x, int r) {
        int n = size(x);
        vector[n * r] result;
        for (i in 1:r) {
            result[((i - 1)*n + 1):(i*n)] = x;
        }
        return result;
    }
}
```

C Appendix: discrete-index HMMs in Stan, hierarchical case

To change the model to allow for changing parameters as described in 3.2, we simply change the following aspects of the Stan model.

The parameters are extended such that there is one for every data point, with further parameters corresponding to hierarchical distribution parameters.

```
parameters {
    vector<lower=0, upper=1>[n_st] p_diag;
    real<lower=-3, upper=3> mu[n];
    real<lower=0, upper=1> sigma[n];
    real<lower=0, upper=2> scale[n];

    vector<lower=0, upper=10>[s] p_a;
    vector<lower=0, upper=10>[s] p_b;
    real<lower=-3, upper=3> mu_m;
    real<lower=0, upper=3> mu_s;
    real<lower=0, upper=3> sg_s;
    real<lower=0, upper=4> cs_s;
}
```

The observation model in the transformed parameters section changes to account for the varying parameters,

```
normal_lpdf(y[k] | mu[k] + cosine_term[j]*scale[k], sigma[k]);
```

The model block changes to account for the hierarchical priors,

```
model {
    vector[n_st] p_a_transformed = tile(p_a, num_years);
    vector[n_st] p_b_transformed = tile(p_b, num_years);

    mu ~ normal(mu_m, mu_s);
    sigma ~ exponential(1/sg_s);
    scale ~ exponential(1/cs_s);
    p_diag ~ beta(p_a_transformed, p_b_transformed);

    target += hmm_marginal_banded(log_omega, p_diag, rho);
}
```

We do not change the generated quantities block (and do not code up a more efficient version of the forward backward algorithm as the generated quantities block is only run once at the end of the inference process unlike the forward algorithm, which would be run multiple times during the inference process as the parameters are changed).

We change the inference method to variational inference instead of maximum likelihood due to the need to integrate out the hierarchical parameters during inference (as a maximum likelihood estimate of parameters such as a_i, b_i in such an overparameterised model is perhaps unlikely to lie in the posterior's typical set). This is done by replacing `model.optimize` with `model.variational`. We also increased (doubled with respect to Stan's defaults) the number of gradient samples for the ELBO calculation, which was needed to obtain reasonable outputs.

D Appendix: cts-HMM forward algorithms in Stan

A computationally inefficient example allowing for time varying transition matrices in the forward algorithm is shown below. In this example, the time varying transition matrix is created using Equation (2).

```
functions {
    real hmm_marginal_cts(matrix log_omega,
                          vector Gamma_diag,
                          vector rho,
                          vector ts) {
        int K = dims(log_omega)[1];
        int N = dims(log_omega)[2];
        vector[K] log_alpha;
        vector[K] inner_sum;
        vector[K] inner_vec;
        vector[K] log_Gamma_diag = log(Gamma_diag);
        matrix[K, K] log_trans_mat;
        int min_i; int max_i;
        log_alpha = log_omega[, 1] + log(rho);
        if (N > 1) {
            for (n in 2:N) {
                log_trans_mat = diag_rate_to_full(Gamma_diag);
                log_trans_mat = matrix_exp((ts[n] - ts[n - 1]) * log_trans_mat);
                log_trans_mat = log(log_trans_mat + 1e-10);
                for (i in 1:K) {
                    inner_vec = log_alpha + log_trans_mat[, i];
                    inner_sum[i] = log_sum_exp(inner_vec);
                }
                log_alpha = log_omega[, n] + inner_sum;
            }
        }
        return log_sum_exp(log_alpha);
    }
}
```

E Appendix: other models

We implemented the continuous time Markov chain model in `tensorflow-probability` due to out of the box support for time varying transition matrices. We implemented our SDE models using `torchsde` due to support for differentiable SDE solvers.

The basic algorithm followed in both cases (as VI and maximum likelihood estimation are both optimization problems) is,

```
log_posterior_or_lower_bound(params) $\leftarrow$ function(params, data) ...
parameters $\leftarrow$ ...                     $\triangleright$ variational params or params on which inference is done
optimizer $\leftarrow$ Optimizer(parameters, lr)
while loss not converged do
    loss $\leftarrow$ $-$log_posterior_or_lower_bound(parameters)
    grad $\leftarrow$ loss.grad()
    params $\leftarrow$ optimiser.step(params, grad)
end while
posthoc analysis
```

F Appendix: Results of SDE models

Our SDE models ran into severe difficulties with inference. As an example, we assume the prior,

$$\begin{bmatrix} dz^a_\delta \\ dz^b_\delta \\ dt_\delta \end{bmatrix} = \begin{bmatrix} z^b_\delta \\ -\lambda^2 z^a_\delta - 2\lambda z^b_\delta \\ \alpha * \sigma^+(z^a_\delta) \end{bmatrix} d\delta + \begin{bmatrix} 0 \\ 1 \\ 10^{-2} \end{bmatrix} \odot d\mathbf{W}_\delta,$$

where α and λ are set to be constant, and σ^+ corresponds to the softplus operation. Note that the prior over $\mathbf{z}$ is a Matérn-3/2 Gaussian process. We use the following observation model,

$$s_{\delta_i} | t_{\delta_i} \sim \text{Laplace}(\sin(\pi t_{\delta_i}), 0.05).$$

We tried to fit a variational posterior over t_{δ_i}, using the variational SDE,

$$\begin{bmatrix} dz^a_\delta \\ dz^b_\delta \\ dt_\delta \end{bmatrix} = \begin{bmatrix} f^a_{nn}(\mathbf{z}_\delta; t_\delta) \\ f^b_{nn}(\mathbf{z}_\delta; t_\delta) \\ \alpha * \sigma^+(f^c_{nn}(\mathbf{z}_\delta; t_\delta)) \end{bmatrix} d\delta + \begin{bmatrix} 0 \\ 1 \\ 10^{-2} \end{bmatrix} \odot d\mathbf{W}_\delta, \tag{3}$$

where $\mathbf{f}_{nn}$ was parameterized using a neural network, to some data simulated from the prior. The expected mean of the observation model for a few samples from the variational posterior are shown below in Figure 5.

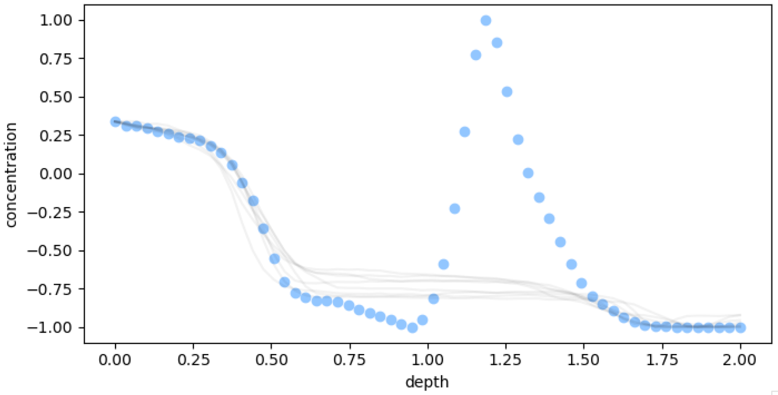

Figure 5: Results of our latent SDE models on synthetic data, showing a poor local maximum in the ELBO reached by VI.

We also tried to use sparse Gaussian processes for placing priors and variational approximations on $\mathbf{z}_\delta$, utilizing random Fourier features [Hensman et al., 2016] to sample functions from these GPs to work seamlessly with torchsde, finding no improvement in results. Pre-fitting a neural-network based SDE prior (using spectra derived from the data) however results in a better fit to the ice core data (illustrated below in Figure 6), however, results are still very (impractically) sensitive to initialisation.

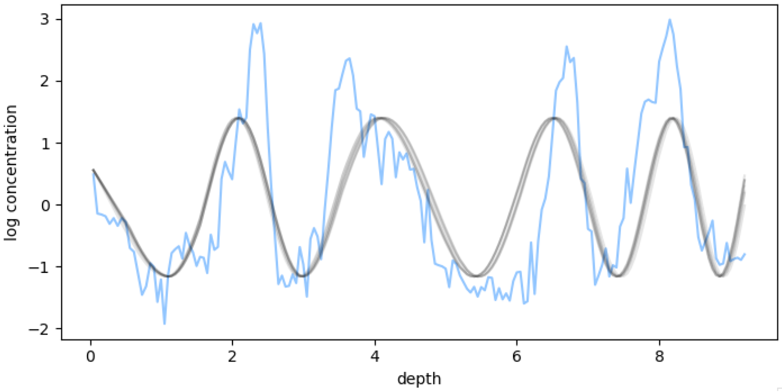

Figure 6: Results of our latent SDE model using spectra to initialize the prior, showing a relatively good fit to the ice core data. The blue line shows ground truth data, while the grey lines show the mean of the observation model using different samples of the posterior over time.

Future work can also involve exploration of Kalman filtering algorithms for this problem, as our SDE priors are valid models that may be assumed for state estimation in extended Kalman filtering.