GPCore: A Gaussian Process Approach for Inferring Ice Core Chronologies

Tom Andersson

Department of Engineering
University of Cambridge

This report is submitted for the degree of
Master of Engineering

Pembroke College July 2019

Acknowledgements

I would like to thank Rich Turner and Will Tebbutt for their excellent supervision of the project and for making it so enjoyable. Thank you to Scott Hosking and Holly Winton for providing supervision on the climate science aspects of this project and for introducing the problem of ice core chronology inference to us. Thank you also to Liz Thomas for kindly providing the dataset of British Antarctic Survey ice core measurements, without which progress would have been very difficult.

Technical abstract

This report details the development of a flexible, non-parametric Bayesian approach for fitting high-resolution chronologies in ice cores. We call the algorithm *GPCore* and use Gaussian processes (GPs) to model ice core data in a Bayesian statistics framework. As input, GPCore requires a dataset of ice core measurements containing depth-series of proxies, which are chemicals that relate to the local climate history of the ice core drilling site. The proxies must be sampled at a sufficiently high rate so that their seasonal dependencies can be resolved within each annual layer. The dataset must also include an initial guess of the ice core's annual layer boundaries from manual or automated layer counting, which are only used for initialisation of GPCore's optimisation procedure. GPCore attempts to maximise the posterior probability of the chronology given the input data, extracting and combining information about the ice core's timescale contained within each proxy series. The output of GPCore is a vector of age predictions associated with the locations of each proxy sample, the uncertainties of which can be estimated using the Laplace approximation or a sampling method with the unnormalised posterior. GPCore is a work-in-progress, but in early-stage trials has demonstrated an ability to refine and correct the depth-time mapping of the input layer count samples and provide a high-resolution chronology with rigorous prediction uncertainties.

In this work, we primarily contrast the performance of GPCore with *StratiCounter*, a state-of-the-art layer counting algorithm for ice core chronology inference, which we use as a benchmark when analysing GPCore. StratiCounter attempts to infer an ice core's depth-time mapping by finding a probable set of layer boundary locations in the annually-periodic proxy data. In black-box experiments with StratiCounter, we found its predictions to be prone to undesirable artefacts. We pose the problem in a new paradigm, where we view StratiCounter's learning of the annual layer boundaries as only indirectly learning yearly-spaced samples from the depth-time mapping. In an attempt to improve robustness, in GPCore we choose to directly infer finely-spaced samples from the depth-time mapping, with estimates at each proxy measurement location.

The contributions of this work comprise three main parts: 1) a novel statistical model for the task of inferring the depth-time mapping of ice cores, 2) a bespoke optimisation procedure to solve the challenges associated with this inference task and 3) well-commented

Python code to implement the algorithm. We hope to eventually make the code open-source for the benefit of researchers developing or using the algorithm in the future.

The report herein is structured as follows. In Chapter 1, we start by providing context and motivation for this study, introducing the problem that has been tackled, and conducting a literature review of existing Bayesian chronology inference methods. In Chapter 2, we describe the theoretical background of StratiCounter and GPCore, and we outline GPCore's statistical model and optimisation procedure. We proceed to evaluate the performance of GPCore, justifying its design choices and comparing it with StratiCounter in Chapter 3. Despite the early stages of the algorithm we have developed, GPCore is generally able to outperform StratiCounter according to the performance metric and simulated datasets utilised in this study. GPCore also shows promise in inferring the chronology of a real ice core, while StratiCounter produces several errors. In Chapter 4 we conclude by outlining more work that should be explored to verify and advance these findings. Future work may investigate an 'unsupervised' mode of GPCore when no manual layer count data is available, as well as improving the algorithm's efficiency.

Table of contents

Nomenclature

Superscripts

B	Backward pass
c	Chunk index
F	Forward pass
FB	Combined forward-backward pass
FT	Fine-tuning phase
k	Forward-backward pass index

Other Symbols

$\boldsymbol{d}$	Depths at which the proxies are sampled from the ice core
$\mathcal{D}_{\text{core}}$	An ice core dataset
$\boldsymbol{d}_m$	Input layer count depths
$\mathcal{D}_{\text{sim}}^{(0)}$	Toy ice core dataset 0 (ground truth layer counts)
$\mathcal{D}_{\text{sim}}^{(1)}$	Toy ice core dataset 1 (missing layer counts)
$\mathcal{D}_{\text{sim}}^{(2)}$	Toy ice core dataset 2 (6 layers of missing proxy data)
$\mathcal{D}_{\text{sim}}^{(3)}$	Toy ice core dataset 3 (3 layers of missing proxy data and layer counts)
EQ	Exponentiated quadratic GP
Lin	Linear GP
N	Number of samples in each of the proxy depth series
N_y	Number of proxies used for the ice core chronology inference
Per	Periodic GP

$\boldsymbol{t}$	Chronology vector of proxy time points
$\boldsymbol{\tau}_\delta$	GPCore optimisation variable: parameterised differenced chronology vector
$\boldsymbol{t}_\delta$	Differenced chronology vector
$\boldsymbol{t}_{GPC}$	GPCore's output chronology vector of ice core age predictions
$\boldsymbol{t}_{gt}$	Ground truth proxy time points
θ_{GPC}	GPCore's algorithm parameters
$\boldsymbol{t}_m$	Estimated age associated with the input layer counts
Y	Set of all proxy series
$\boldsymbol{y}_i$	The ith proxy series

Acronyms / Abbreviations

BAS	British Antarctic Survey
CAP	Chunk annealing phase
FTP	Fine-tuning phase
GP	Gaussian process
HMM	Hidden Markov model
HSMM	Hidden semi-Markov model
i.i.d.	Independent and identically distributed
ML	Machine Learning
NLML	Negative log-marginal likelihood
OC	Overlapping chunks
r.v.	Random variable
SC	StratiCounter
w.r.t.	With respect to

Chapter 1

Introduction

Interdisciplinary work between the machine learning (ML) and environmental science research communities has tremendous potential to trigger advances in both fields. The number and size of observational and simulated datasets in environmental science are growing faster than new laws can be postulated to model properties of the data. ML provides a solution to this problem by allowing patterns in the data to emerge without the need for hard-coded physical models, revealing unexpected phenomena and thereby fuelling scientific discovery. ML principles can also be used to produce statistical augmentations of climate models, offering more rigorous estimates of prediction uncertainty than classical ensemble methods. Furthermore, ML approaches like regression can elegantly solve data processing problems involving multiple sources of interrelated, misaligned and noisy data series, as are often found in the geosciences. Karpatne et al. (2018) outlines some of the numerous ways in which ML can be applied in the geosciences. Despite ML being ripe for application in climate science, integration between these two research communities has historically been lacking (Sweeney et al., 2018), with the exception of the emerging field of *climate informatics.* This project exists within climate informatics and is part of a collaboration between the Computational and Biological Learning Lab and the British Antarctic Survey (BAS).

In this chapter, we will introduce the object of study in this report: ice core chronology inference. After detailing the problem tackled in this study, we continue by showing ice core data provided by BAS and give a literature review of existing Bayesian ice core chronology inference methods. The subsequent chapters will focus on GPCore – the algorithm we have developed to perform high-resolution chronology inference.

1.1 Ice Cores and the Problem of Ice Core Chronology Inference

Ice sheets are subject to the incremental deposition of snow onto their surfaces, which becomes compacted into ice at a depth of 50-120m. If the rate of snow accumulation is sufficiently high, the ice sheet will exhibit vertical annual layering corresponding to each year of snowfall. These annual layers vary in thickness due to differing amounts of snowfall in each year and become increasingly thin with depth due to horizontal ice flow.

Ice cores are long cylindrical pieces extracted from ice sheets using hollow drills, sometimes several kilometres deep, and can contain ice as old as 800,000 years in existing cores (Jouzel et al., 2007). The chronology or 'timescale' of an ice core refers to how depth down the core maps to the ice's age (i.e. the time at which snow was deposited to eventually form that ice).

Snow exhibits chemical and physical properties that relate to climate conditions at the time of snowfall, such as past temperature or solar activity, which are archived in ice cores. Proxies are measurements of these properties, such as chemical concentrations. Proxy variables are of significant interest in the environmental sciences because they can be used to infer climate conditions in the pre-instrumental period (times in the Earth's past that predate direct human measurements). Proxies also provide information about the ice core's chronology because they often depend on the seasons and thus display annual cycles under favourable conditions.

Ice cores therefore provide a record of past climate and are crucial objects in environmental science. However, one obtains a depth series upon measuring proxy data from an ice core, which is of little scientific value. The problem of ice core chronology inference is that climate scientists are primarily interested in proxy *time* series, which requires one to estimate/infer the ice core's chronology using some dating method. With proxy time series, proxies from different ice cores can be compared on the same axis, revealing a spatio-temporal map of proxy variables. Such a dataset can provide information on the rate, spatial distribution and causality of past climate phenomena.

Ice cores are typically dated using the stratigraphic approach of manually counting annual layers in the proxy depth series (assuming the snow accumulation rates have been high enough to produce annual layers). Manual layer counting is usually performed by searching for summer peaks or winter troughs in the proxy data, thus identifying boundaries between yearly cycles. These yearly-spaced samples from the ice core's chronology can then be linearly interpolated to map from proxy depth series to proxy time series. Manual layer counting can be a tedious process, especially if there are many annual layers, involving subjectivity and lacking well-quantified uncertainty estimates. One extreme example is the Greenland Ice Core Chronology 2005 (GICC05) project (Svensson et al., 2008), where over 60,000 annual layers were counted by multiple researchers. In total, an estimated

5-10 person-years were spent counting layers for GICC05 (Winstrup, 2016). Additionally, a correspondence with the Niels Bohr Institute given in Wheatley (2015) reveals that roughly one person-month was required to date a 3,000-layer core. These two cases suggest that around 10-100 annual layers can be identified per person-day. There is clearly a need to automate this process. Furthermore, using a Bayesian probabilistic formulation for the chronology inference algorithm provides the benefit of principled uncertainty estimates.

1.2 Ice Core Datasets

Obtaining the chronology of an ice core from the proxy depth series, whether through manual layer counting or an automated method, can be made complicated by a number of phenomena, for example: insufficient temporal resolution to reveal clear annual cycles in the proxy depth series; the absence of snowfall over an extended period; extreme precipitation events (Turner et al., 2019); the loss of layers by wind reworking; and irregularities in the climate over a particular year. Thus proxy data will rarely appear purely as a warped sinusoid and chronology inference can be a challenging problem.

The ice core data used in this project was provided by BAS and comprises three ice cores drilled from West Antarctica, each extending ∼130m in depth and with nine proxies measured at 5cm intervals. As these datasets are yet unpublished, we refer to the ice cores as Ice Core 1, Ice Core 2, and Ice Core 3.

Fig. 1.1 plots the nine proxy variables from an 10-metre section of Ice Core 3, with manually counted layer boundaries shown as shaded strips. The proxies were pre-processed by log-transforming them to improve their symmetry, followed by subtracting their mean and normalising by their standard deviation. Ice Core 3 has the largest annual layer thicknesses of the three cores and thus its proxies are sampled with the greatest temporal resolution. Two of the nine proxies shown (MSA and δ^{18}O) are clean enough to display clear annual cycles, while the others are much noisier. However, we stress at this point that such noisy proxy series still provide some information on the ice core's annual layers, although they may simply be ignored during manual layer counting.

Fig. 1.2 shows the set of manual layer counts for Ice Core 1, Ice Core 2 and Ice Core 3. We point out that the unit of time adopted in this project is 'years before 2019' (yb2019), measuring the age associated with the ice core samples in years before 12.00am on the 1st of January, 2019. From the full set of layer counts plotted in Fig. 1.2a, it is clear that the mean layer thickness varies between ice cores depending on the rate of accumulation at each site. As a result, Ice Core 1 reaches over 400 years into the past before 2019, while Ice Core 3 only goes back 161 years. One can also see the gradual decrease in layer thicknesses due to the onset of ice flow at ∼80m, but this effect would be clearer in longer cores. Fig. 1.2b shows a zoomed-in view with only the top 20m layer counts plotted. While the layer count samples are uniformly spaced 1 year apart along the time axis, the layer thicknesses vary between annual layers due to changes in the amount of snowfall experienced per year.

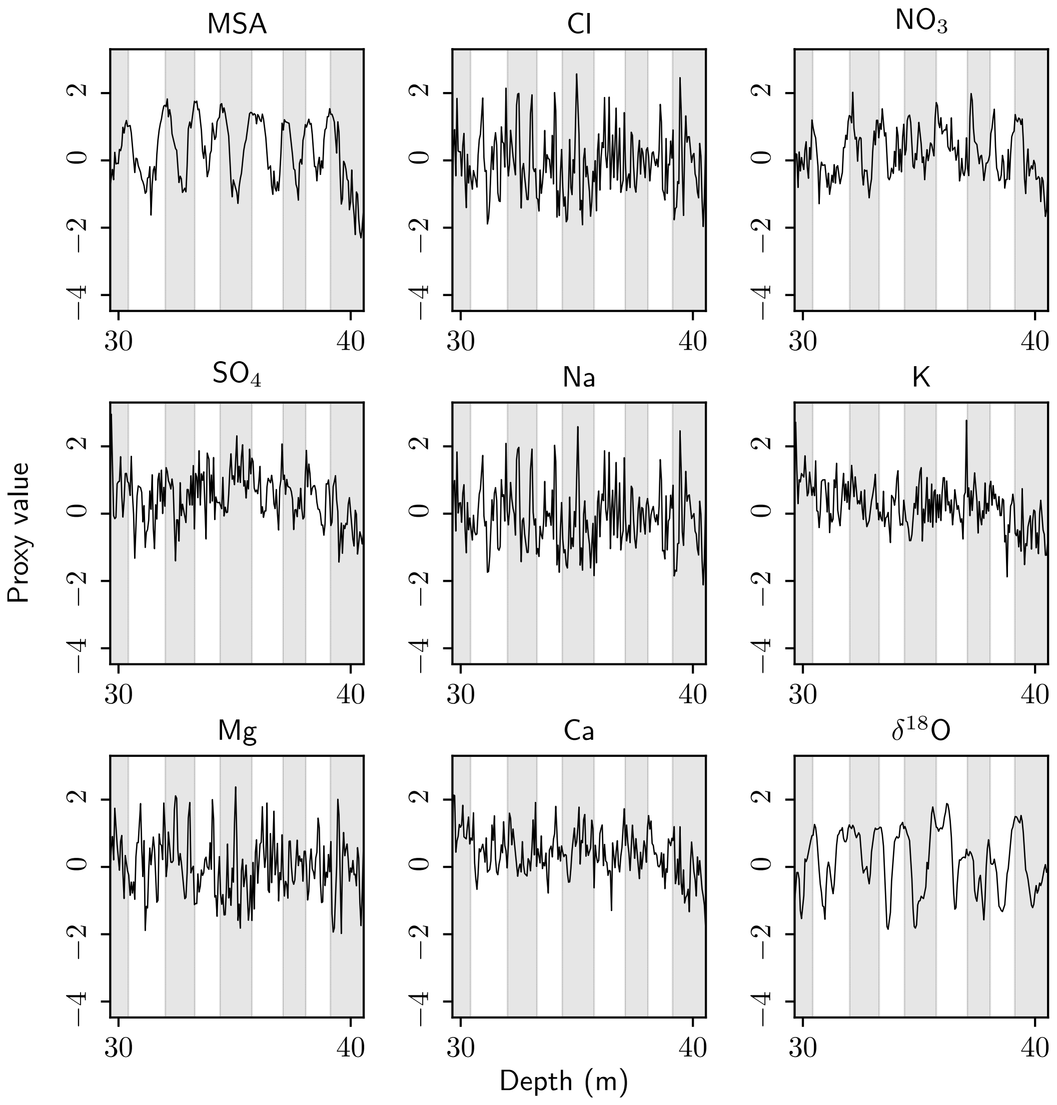
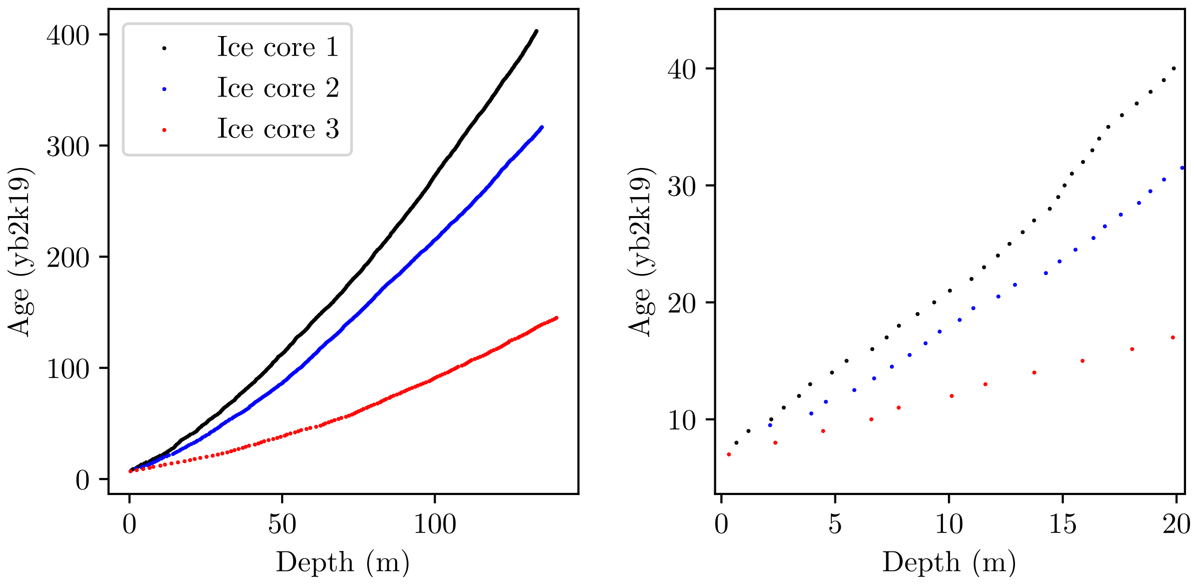


Fig. 1.1 10-metre section of the 9 proxies measured from BAS Ice Core 3, with stripes representing manually counted annual layer boundaries. Note that the proxies have been log-transformed and normalised by their mean and variance.



(a) All layer count samples. (b) Top 20m only.

Fig. 1.2 Manual layer count samples for the three BAS ice cores.

1.3 Past Work

A number of ice core chronology inference methods have been developed over the previous two decades. Here we provide a brief outline of some of these methods, restricting our attention to the few algorithms that are based on Bayesian statistics and thus provide principled uncertainty estimates. To our knowledge, this leaves only StratiCounter (Winstrup, 2016; Winstrup et al., 2012), IceChrono1 (Parrenin et al., 2015), Wheatley et al. (2012) and Wheatley (2015). We give some of the modelling choices made in these studies, but note that any notation used is self-contained in this section.

1.3.1 StratiCounter

Layer detection methods are generally used to construct chronologies when the snow accumulation rate at the ice core site has been sufficiently high to produce annual layers. StratiCounter was the first layer counting algorithm to use a Bayesian formulation and provide a rigorous treatment of uncertainty. We give a more detailed overview of the StratiCounter statistical model and optimisation procedure in Section 2.1. The primary limitation of StratiCounter is that it requires an initial set of manually-identified layer boundaries before it can run, thus depending upon extensive prior study of the data and only outputting a new set of layer boundary estimates (with uncertainty). Furthermore, it is prone to undesirable artefacts in its inferred layer boundaries, as will be demonstrated in this report.

Due to time constraints, in later chapters we only contrast GPCore with StratiCounter, which we use as a benchmark when investigating the performance of GPCore. Future work should compare GPCore with other freely-available chronology estimation algorithms.

1.3.2 IceChrono1

If the annual layers are too thin and the proxies are sampled super-annually, one must generally resort to glaciological modelling the sedimentation process (such as snow accumulation, snow densification, and ice flow) in order to indirectly infer the chronology. In addition, events like volcanic eruptions with known times of occurrence may be identified in a proxy signal and used as tie points for the chronology. IceChrono1, an upgraded version of the Bayesian tool *Datice* (Lemieux-Dudon et al., 2010), uses a combination of these two approaches, employing sedimentation modelling and so-called 'chronological observations' (for example, layer counting information can be provided as dated intervals if it exists). Furthermore, a common chronology is inferred across multiple cores, using inter-ice core matching of dated events to produce coherent timescales between cores. The probabilistic model used in IceChrono1 comprises Gaussian priors and likelihoods, allowing the chronology to be inferred using the least-squares method. Optimisation is performed with the LM algorithm by estimating the Jacobian of the objective function using finite differences.

One limitation of IceChrono1 is the need to define covariance matrices for the Gaussian priors over glaciological functions (the snow accumulation rate, snow densification, and layer thinning functions), relying on prior sedimentation modelling using classical approaches. One must also define covariance matrices for the chronological observations of the Gaussian likelihood terms, thus requiring observation uncertainty estimates as input. As well as depending on expert knowledge, the choices for these covariance matrices will involve an element of subjectivity and ambiguity. For example, in Parrenin et al. (2015), prior covariance matrices with an across-diagonal linear decrease in amplitude are applied to the Antarctic ice core chronology 2012 (AICC2012) dataset. The authors suggest using a combination of models and proxy data to deduce appropriate priors.

IceChrono1 rightfully treats the chronologies for air and ice measurements differently. In an ice sheet, the firn (upper 50-120m section) contains permeable snow, which air can pass through freely. Air is only trapped at the 'lock-in depth' (LID) when the snow is compacted to ice. Thus the chronology of air measurements (such as CO_2) lags behind that of proxies measured from the ice. We do not consider this phenomenon in this study as we only work with inferring the age of ice as opposed to the trapped air in ice cores.

One of the primary appeals of IceChrono1 is its extensive integration of dated chronological markers with the sedimentation models. Multiple ice core timescales are synchronised through inter-core matching of events in the proxy data, allowing for dating information to be transferred between cores and resulting in synchronised chronologies. This approach can theoretically be applied between hemispheres. For example, an event dated using layer counting for the NGRIP ice core in Greenland could be chronologically matched to the same event identified in an Antarctic ice core.

Because of the modelling approach taken and the expensive Jacobian estimation, IceChrono1 is not appropriate for high-resolution chronology inference. In Parrenin et al. (2015), the timescale is typically estimated at 1 kiloyear intervals and linear interpolation is used for finer-scale estimates. This contrasts with the algorithm we have developed, GPCore, which works with high-resolution, sub-annual proxy data. Therefore, IceChrono1 and GPCore are not directly competing algorithms; IceChrono1 is well-suited for low-resolution dating of deep, low accumulation ice cores without stratigraphic layering, while GPCore would be more appropriate for high-resolution dating of high accumulation cores containing ice less than a few kiloyears old. For this reason, we do not provide further comparison with IceChrono1 in this study.

1.3.3 Wheatley

A simple layer counting algorithm for low-noise proxy series is presented in Wheatley et al. (2012). A significant benefit over StratiCounter is that the algorithm does not require a set of initial manual layer counts. First, the proxy signal is de-trended and normalised to produce a standardised signal $\boldsymbol{s}$ that more closely resembles a warped periodic function. The principle of the algorithm is then to classify samples in $\boldsymbol{s}$ into quarter-cycle 'runs' comprising peak (P), descending (D), trough (T) and ascending (A), or as an issue (χ) which requires manual assessment. The user chooses a cut-off parameter ν, and runs of $\boldsymbol{s}$ above $+\nu$ are labelled as a potential peak P*; runs below $-\nu$ are labelled as a potential trough T* (the star denotes a potential classification). Runs between a P* and a T* or a T* and a P* are labelled as D* or A*, respectively, and χ otherwise. Finally, potential runs are deemed 'certain' runs if they are central to 5 consecutive runs obeying a periodic pattern, (such as A, P, D, T, A), and otherwise are labelled as issues, χ. Probabilities are assigned to potential run patterns within issues (called 'reconstructions') by using a linear Gaussian regression model for log-total run length given mean run depth. The model for possible reconstructions are fit using least-squares on training data from analogous groupings of the certain runs.

This algorithm is very efficient and could significantly speed up the manual layer counting procedure by restricting expert attention only to the regions labelled as issues. Limitations of the approach are that it relies on the noise level of the proxy signal being very low and that it does not consider uncertainty in the runs which are successfully labelled as a quarter-cycle.

To our knowledge, at the time of its publication, Wheatley et al. (2012) presented the highest resolution algorithm for estimating the chronology (four times per annual layer) – all other methods operate over annual layer boundaries (e.g. StratiCounter) or super-annual resolution (e.g. IceChrono1). An even higher resolution method, first outlined in Wheatley et al. (2014), is detailed in the PhD thesis Wheatley (2015). The latent timescale is estimated at each proxy sample location to yield a chronology vector $\boldsymbol{\tau}$ (which we call $\boldsymbol{t}$ in the following chapters). A proxy signal is modelled using an annually-periodic function of time $f(\tau)$ such that $f(\tau) = f(\tau + 1)$. In the thesis, $f(\tau) = \sin(2\pi\tau_i)$ is used because the proxies considered are roughly sinusoidal. The proxy model then takes the form $x_i = \alpha_i \sin(2\pi\tau_i) + \beta_i + \epsilon_i$, with $\epsilon_i \sim$ i.i.d. $\mathcal{N}(0, \sigma^2)$. The unknown parameters of the model are $\boldsymbol{\theta} = \{\boldsymbol{\tau}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$. The amplitude and offset are controlled by α_i and β_i, respectively, and are assigned Gaussian random walk priors: $\alpha_i \sim \mathcal{N}(\alpha_{i-1}, \sigma_\alpha^2)$, $\beta_i \sim \mathcal{N}(\beta_{i-1}, \sigma_\beta^2)$. It is intended that the processes $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ will model correlated noise in the signal resulting from the difference between the model's periodic function $f(\tau)$ and the shape of the underlying proxy's annual cycle. The prior over time is achieved by differencing the chronology vector $\boldsymbol{\tau}$ and assigning Gamma distributions such that $\delta_i = \tau_i - \tau_{i-1} \sim G(\psi, \lambda)$. An MCMC scheme is used to sample from the joint posterior over $\boldsymbol{\tau}, \boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ using Metropolis-Hastings and Gibbs sampling methods.

The research in Wheatley (2015) has not been applied in the ice core community for a number of reasons – partly due to limitations in the approach. One of the main drawbacks is its inability to deal with noisy proxy data. The MCMC method for finding $\boldsymbol{\tau}$ is partially dependent on the simple algorithm presented in Wheatley et al. (2012) being executed first, which itself relies on the signal-to-noise ratio (SNR) being very high to allow for the correct classification of the signal's seasonal runs. The dependence lies in the initialisation of $\boldsymbol{\theta}$ and automatically finding a minimum threshold for α_i in order to prevent the trivial solution of $\alpha_i \to 0 \; \forall i$ and $\boldsymbol{\beta}$ fitting to the signal. Though this dependence is not fundamental, the MCMC algorithm would require modification to operate independently of the Wheatley et al. (2012) algorithm. However, it is further assumed that the SNR is large in the model for the timescale $\boldsymbol{\tau}$. The MCMC algorithm is only demonstrated using well-behaved proxy signals, such as H_2O_2 from the Gomez ice core. This requirement for a clear seasonal signal limits the utility of the MCMC algorithm because dating methods are most needed for long, noisy proxy depth-series. A further drawback is the need to define priors over the hyperparameters ϕ, λ, σ, σ_α and σ_β, which should elicit expert knowledge about the ice core dataset in relation to the operation of the algorithm.

Despite shortcomings of the MCMC algorithm discussed above, it is a strong attempt at performing ice core chronology inference at the highest resolution possible (at every proxy measurement location). A particular strength of Wheatley's work is its characterisation of the posterior over $\boldsymbol{\tau}$ through sampling. This is able to capture the true multimodality of the posterior resulting from aliasing effects of uncertain layers. Future work on the design of GPCore should review Wheatley (2015) more thoroughly to contrast its approach with GPCore.

1.4 Ice Core Chronologies as Depth-Time Mappings and High-Resolution Inference

Fig. 1.3 illustrates the framework with which we pose the problem of ice core chronology inference and how it differs from that of layer counting approaches. A simulated proxy series is plotted – most real proxies will not have such a clear annual cycle. We view an ice core's chronology as a depth-time mapping, which is a monotonically increasing function whose input is depth and output is the ice's age. In this framework, layer counting algorithms indirectly operate over yearly-spaced samples from these depth-time mappings. The illustration plots estimated layer boundaries as red points, with the ground truth depth-time mapping shown as a black line. The observed proxy depth series exhibits warped annual cycles resulting from non-linearities in the depth-time mapping[1]. The task is to infer a posterior distribution for the chronology vector $\boldsymbol{t}$ to yield an estimate of the proxy time series (with uncertainty).

The benefit of layer boundary inference is that the dimension of the chronology vector being inferred is minimised, while still operating over informative tie points in the chronology. However, we hypothesise that directly inferring finely-spaced samples from the depth-time mapping, with high-resolution estimates at each proxy measurement location, could improve robustness at the expense of increased computational complexity. An intuitive example of how robustness could be improved is that making an 'extra layer error' would result in two cycles-worth of timescale samples being 'wrong', as opposed to only a single erroneous sample in a layer counting framework. Furthermore, a mistake in an inferred layer boundary perturbs the solution by 1 year, while errors in a high-resolution chronology estimate could be constrained to accumulate more slowly.

[1]This is another way of phrasing the non-uniformity in layer thickness.

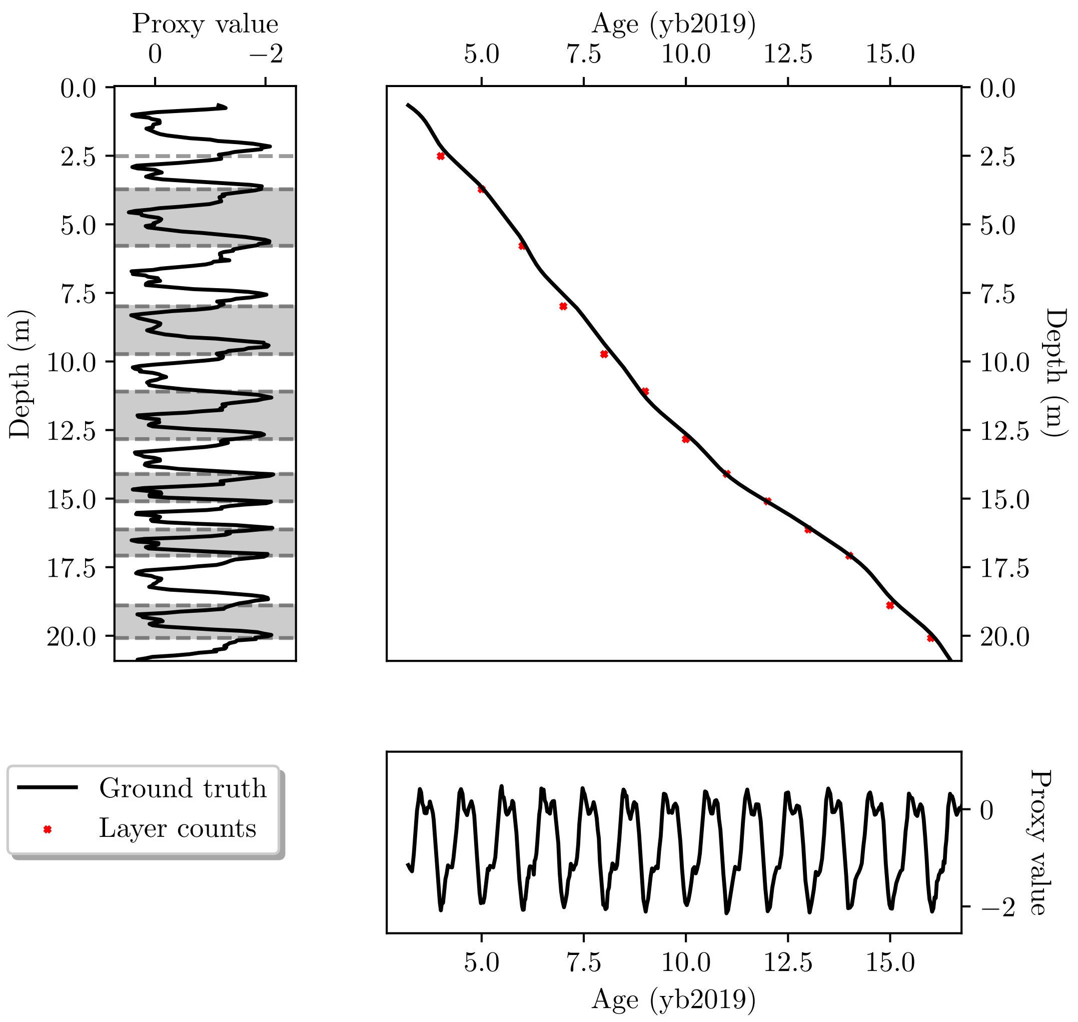


Fig. 1.3 We pose the problem of ice core chronology inference as that of estimating a depth-time mapping at a set of depth locations. A periodic function is used in place of real proxy data for illustration purposes. The ground truth proxy time series is shown in the lower figure, exhibiting yearly cycles. Layer boundaries resulting from manual or automated counting are plotted as red points in the top right figure, with the ground truth depth-time mapping shown as a black line. When the proxy time series is passed back through the depth time mapping, one obtains a depth series with warped periods corresponding to annual layers (shown as alternating light and dark strips).

Chapter 2

The GPCore Algorithm

Now that the problem has been introduced, this chapter will first outline the technical details of a state-of-the-art algorithm for ice core chronology inference, StratiCounter. GPCore's theoretical background will be discussed by introducing the concept of Gaussian Processes. We will then introduce our notation for ice core datasets that we use to infer ice core chronologies and describe the statistical model employed in GPCore. Finally, we give details of GPCore's optimisation procedure for performing inference. For experimental results verifying a number of the design choices discussed here and comparing GPCore's performance with StratiCounter, see Chapter 3.

2.1 Related Work: StratiCounter

Before discussing GPCore's theoretical background, statistical model and optimisation procedure, we will first outline StratiCounter in greater detail than that provided in Section 1.3.1. Although the technical details of StratiCounter are not of principal importance in this study, it is hoped that summarising its approach will allow the reader to obtain some insight into the novelties of GPCore before the differences between GPCore and StratiCounter are explicitly discussed at the end of this chapter.

StratiCounter is an open-source, MATLAB-based script and is described by its author as a layer counting algorithm. The object that StratiCounter performs inference over is the set of annual layer boundaries in the ice core. The probabilistic model employed for this task is a hidden semi-Markov model (HSMM) (Yu, 2010). A HSMM differs from the popular hidden Markov model (HMM) in that each hidden state is allowed to generate a variable number of observations before the hidden state advances. This model can be applied to the layer counting problem by choosing the hidden state to be the layer number that each proxy sample originated from. Assuming that the layer number increases in increments of one as the ice core is descended (i.e. no missing or repeated layers), this results in a simplification to the HSMM whereby the hidden state must deterministically advance in increments of one; the only stochasticity lies in the duration of each hidden layer state.

The layer duration in samples, denoted d, is given a lognormal prior distribution parameterised by μ_d and σ_d^2 such that $p(d) = \ln\mathcal{N}(\mu_d, \sigma_d^2)$, where $\ln\mathcal{N}$ denotes the lognormal distribution. The proxy samples for each layer j are assumed generated from a linear Gaussian model, $\boldsymbol{y}_j = X\boldsymbol{a}_j + \boldsymbol{\varepsilon}_j$, where $\boldsymbol{\varepsilon_j} \sim \mathcal{N}(\mathbf{0}, \Lambda_\varepsilon)$ and Λ_ε is diagonal. The basis matrix X collects the basis vectors such that $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots]$ and $\boldsymbol{a}_j$ contains the weights for the each basis vectors. A multivariate Gaussian prior is placed over the weights $\boldsymbol{a}_j$ with mean vector $\boldsymbol{\psi}$ and covariance matrix Φ. The likelihood of the dataset thus depends on both the set of parameters $\theta_{\text{SC}} = \{\mu_d, \sigma_d^2, \Lambda_\varepsilon, \boldsymbol{\psi}, \Phi\}$ and the hidden layer state assigned to each observation. The ice core dataset is processed in batches – each with multiple iterations – where in each iteration the expectation maximisation (EM) algorithm is used to estimate the maximum-likelihood (ML) set of parameters $\theta_{\text{SC}}^{(\text{ML})}$ (using the previous hidden state estimate) and then the forward-backward algorithm is used to extract the ML layer boundaries using $\theta_{\text{SC}}^{(\text{ML})}$. The layer boundaries are initialised at the input manual layer counts to give an initial estimate of the hidden states.

X is held fixed throughout the algorithm and is computed for each proxy from the mean and first principal component of the annual signals extracted from the initial manual layer counts, which we believe to be a shortcoming in the model because a) it enforces a stationarity assumption about the annual period profiles and b) the model is directly influenced by the input manual layer counts. As we will see, GPCore's Gaussian process model addresses both of these shortcomings.

2.2 Theoretical Background: Gaussian Processes for Regression

Since the statistical model used in GPCore is based on GPs, here we will briefly describe GPs in a general sense before applying them to the problem of ice core chronology inference. We will also state some of the equations associated with GPs that will be used in this study. For a thorough introduction to and analysis of GPs, see Rasmussen and Williams (2006).

GPs are a flexible class of probabilistic model used primarily for supervised learning problems. In supervised learning, the task is to learn a probability distribution over function mappings $f(x) : \mathbb{R} \to \mathbb{R}$ from inputs[1] $x \in \mathbb{R}$ to outputs $f(x) \in \mathbb{R}$. The training data consists of a set of input-observations pairs, $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$, where we assume that

[1]We will restrict our attention here to scalar-valued inputs and outputs, although the theory extends readily to general case of vector-valued inputs and outputs.

the underlying function values $f(x_i)$ are corrupted by additive white Gaussian noise ε_i to yield the observations

$$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim \text{ i.i.d. } \mathcal{N}(0, \sigma_n^2). \tag{2.1}$$

A fundamental machine learning problem is to combine one's prior knowledge about the function mapping $f(x)$ with the new information gained from observing the dataset $\mathcal{D}$. A common approach in such a scenario is to write $f(x)$ as a member of a parametric family f_θ and place a prior distribution $p(\theta)$ over the parameters in order to indirectly specify a distribution over functions.

An alternative *non-parametric* approach is to specify a prior directly over functions and perform inference directly in the function space. In this case, the parameters can be viewed as the function values themselves. GPs are such non-parametric models. If one considers a continuous, univariate function as an infinite-dimensional vector, then a GP can be thought of as infinite-dimensional random vector whose distribution is an infinite-dimensional multivariate Gaussian distribution. GPs are specified completely by a mean *function* $m : \mathbb{R} \to \mathbb{R}$ and a positive semi-definite covariance *function* (or 'kernel') $k : (\mathbb{R}, \mathbb{R}) \to \mathbb{R}$ (as opposed to mean *vector* and covariance *matrix* for finite-dimensional multivariate Gaussians). Loosely speaking, the covariance function measures how similar $f(x)$ and $f(x')$ are expected to be when sample functions are drawn from the GP. In general, $k(x, x')$ will depend on some 'hyperparameters' θ that govern the GP prior. In this report, all GP distributions will be implicitly conditioned on their hyperparameters θ and model structure (i.e. mean and covariance functions) $\mathcal{M}$.

For simplicity, and with little impact on predictive power, we will take the mean function $m(x)$ to be zero in the GP priors used in the GPCore model. In this case

$$m(x) = \mathbb{E}[f(x)] = 0 \tag{2.2}$$
$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] = \mathbb{E}[f(x)f(x')], \tag{2.3}$$

and we will write

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{2.4}$$

At first sight, the infinite-dimensionality appears that it could make inference impossible in practice. However, since we only ever have observations of the process at a finite subset of input points, the GP prior over function values $\boldsymbol{f}$ reduces to a simple multivariate

Gaussian[2] where the covariance matrix is built by sampling from the covariance function at the locations where there is input data, giving

$$\boldsymbol{f}|X \sim \mathcal{N}(\mathbf{0}, K_{X,X}) \tag{2.5}$$

where X is the 'design matrix' collecting all of the observed inputs into a $D \times n$ matrix (though here $D = 1$ and so X is a vector) and $(K_{X,X'})_{m,n} := k(x_m, x'_n)$. Note that this distribution can also be used to draw sample functions from the GP prior. Because of our i.i.d. Gaussian noise assumption, the marginal likelihood (sometimes referred to as the 'model evidence' in other contexts) of the observation vector $\boldsymbol{y}$ is

$$\boldsymbol{y}|X \sim \mathcal{N}(\mathbf{0}, K_{X,X} + \sigma_n^2 I) \tag{2.6}$$

The covariance function $k(x, x')$ of a GP is typically dependent on a set of hyperparameters and a challenge is how to choose the best hyperparameters for a particular problem. The marginal likelihood is of crucial importance because it can be maximised to learn an appropriate set of GP hyperparameters to model a given dataset. The intuition behind this is based on the 'Bayesian Occam's razor' phenomenon, whereby $p(\boldsymbol{y}|X)$ will be maximised by a set of hyperparameters corresponding to a model of intermediate complexity in the parametric family of models. See Section 3.4 in Bishop (2006) and Chapter 28 in MacKay (2003) for further discussion on Bayesian Occam's razor. The resulting optimal solution for the hyperparameters may well have captured some structure in the function mapping and can be used to make refined predictions.

To avoid numerical underflow or overflow resulting from the floating point arithmetic used in computer software, the log-marginal likelihood is maximised in practice[3]. Another caveat is that most optimisation packages implement minimisation by convention, and thus the objective is to minimise negative log-marginal likelihood (NLML), given by

$$\text{NLML} = -\log p(\boldsymbol{y}|X) = \frac{1}{2}\boldsymbol{y}^T\Big(K_{X,X} + \sigma_n^2 I\Big)^{-1}\boldsymbol{y} + \frac{1}{2}\log|K_{X,X} + \sigma_n^2 I| + \frac{n}{2}\log 2\pi. \tag{2.7}$$

Evaluating the NLML costs $\mathcal{O}(N^3)$ due to the matrix inversion and determinant[4], which can become prohibitively expensive as n becomes large. This computational bottleneck is the reason for several of the design decisions in GPCore's optimisation procedure. Although

[2]This result follows from the marginalisation property of Gaussians.

[3]The minima and maxima of the log-marginal likelihood lie at the same points as for the marginal likelihood because log() is a monotonically increasing function.

[4]However, note that to estimate GP hyperparameters the matrix inversion and determinant need only be computed once for a given dataset.

not explored in this project, there are variational inference methods based on a set of M 'inducing points' that reduce the cost to $\mathcal{O}(NM^2)$ (Titsias, 2009).

To make predictions about the function values $\boldsymbol{f}_*$ at unobserved test inputs X_*, first consider the following general identity for a multivariate Gaussian distribution. If

$$p\left(\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{v}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{a}\\ \boldsymbol{b}\end{bmatrix}, \begin{bmatrix}A & C\\ C^T & B\end{bmatrix}\right), \tag{2.8}$$

then

$$p(\boldsymbol{u}|\boldsymbol{v}) = \mathcal{N}(\boldsymbol{a} + CB^{-1}(\boldsymbol{v} - \boldsymbol{b}), A - CB^{-1}C^T). \tag{2.9}$$

The joint distribution over the observations $\boldsymbol{y}$ and the function values $\boldsymbol{f}_*$ (here implicitly conditioned on X and X_*) is given by

$$\begin{bmatrix}\boldsymbol{f}_*\\ \boldsymbol{y}\end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}K_{X_*,X_*} & K_{X_*,X}\\ K_{X,X_*} & K_{X,X} + \sigma_n^2 I\end{bmatrix}\right). \tag{2.10}$$

Applying 2.9 to 2.10 yields the predictive distribution

$$p(\boldsymbol{f}_*|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{f}_*}, \Sigma_{\boldsymbol{f}_*}), \tag{2.11}$$

$$\boldsymbol{\mu}_{\boldsymbol{f}_*} = K_{X_*,X}\Big(K_{X,X} + \sigma_n^2 I\Big)^{-1}\boldsymbol{y}, \tag{2.12}$$

$$\Sigma_{\boldsymbol{f}_*} = K_{X_*,X_*} - K_{X_*,X}\Big(K_{X,X} + \sigma_n^2 I\Big)^{-1}K_{X,X_*}. \tag{2.13}$$

Note that one can produce a predictive mean and error bar plot by setting $\boldsymbol{f}_*$ to be the one-dimensional function value f_* associated with the test input x_*, sweeping x_* along input space and computing the mean μ_* and variance σ_*^2 using 2.12 and 2.13. The resulting distribution of f_* is referred to as the marginal posterior in GP terminology. Like the NLML, obtaining the marginal posterior requires the inversion of an $N \times N$ matrix and so standard GP regression is a costly $\mathcal{O}(N^3)$ procedure.

GPs are flexible probabilistic models because there exists a range of covariance functions that allow different prior assumptions about the function to be encoded. We will only state the following four covariance functions that are used in the GPCore probabilistic model:

Exponentiated quadratic (EQ): $k_{\text{EQ}}(x, x') = \sigma_f^2 \exp(-\frac{1}{2}(x - x')^2/l^2)$,

Periodic (Per): $k_{\text{Per}}(x, x') = \sigma_f^2 \exp(-\frac{2\sin^2(\pi|x-x'|/p)}{l^2})$,

Linear (Lin): $k_{\text{Lin}}(x, x') = \sigma_b^2 + \sigma_v^2 x x'$,

Noise (Delta): $k_{\text{Delta}}(x, x') = \sigma_n^2 \delta(x, x')$,

where $\delta(x, x')$ is the delta function such that $\delta(x, x') = 1$ if $x = x'$ and $\delta(x, x') = 0$ otherwise.

As discussed above, each of these covariance functions depend on a number of different hyperparameters. For example, the EQ length scale l dictates how rapidly the function values can decorrelate and thus whether the function is slowly or rapidly varying. For the periodic GP, the parameter p determines the frequency of oscillations of samples from the model, while l encodes how much the function values decorrelate between periods. Samples from a Lin GP are straight lines, while samples from a Delta GP are simply i.i.d. Gaussian noise. See Chapter 2 of Duvenaud (2014) for further discussion on GP kernels.

We will conclude this section by pointing out that new covariance functions can be constructed by the multiplication and addition of other covariance functions and the positive semi-definite property will be preserved (see Section 4.2.4 of Rasmussen and Williams (2006) for a proof of this result). The samples that result from such combined kernels are often what one would expect. For example, samples from a GP with a $k_{\text{Per}} \times k_{\text{EQ}}$ kernel look like periodic functions with periods that decorrelate over input space.

2.3 Ice Core Datasets for Chronology Inference

Here we will define the notation that we adopt for the ice core dataset and random variables treated in this report.

- $\boldsymbol{y}_i$, the ith proxy series,
- N_y, the number of proxies used for the ice core chronology inference,
- N, the number of samples in each of the proxy depth series,
- Y, the set of all proxy series, $Y = \{\boldsymbol{y}_i\}_{i=1}^{N_y}$,
- $\boldsymbol{d}$, the depth grid at which the proxies are sampled from the ice core,
- $\boldsymbol{t}$, the random variable for the *unknown* ‘proxy time points’ associated with the proxy samples (also referred to as the ‘chronology vector’),
- $\boldsymbol{t}_{gt}$, the ground truth proxy time points,
- $\boldsymbol{d}_m$, the depths at which input layer counts are sampled,
- $\boldsymbol{t}_m$, the estimated age associated with the input layer count depths,

- $\mathcal{D}_{\text{core}}$, the dataset collecting all the GPCore input ice core measurements, $\mathcal{D}_{\text{core}} = \{Y, \boldsymbol{d}, \boldsymbol{d}_m, \boldsymbol{t}_m\}$,
- $\boldsymbol{t}_{GPC}$, GPCore's output chronology vector of ice core age predictions.

In this work we assume a somewhat simplified scenario whereby all proxy series from the core are sampled along the same depth grid. However, we note that the GPCore model can readily be extended to the general case where each proxy depth series is sampled at different depths along the ice core.

2.4 The GPCore Statistical Model

In contrast with StratiCounter's approach of hidden layer states, we pose the inference task as to estimate $\boldsymbol{t}$ given $\mathcal{D}_{\text{core}}$. While StratiCounter's ML solution for the annual layers can be linearly-interpolated to indirectly estimate $\boldsymbol{t}$, we choose to learn $\boldsymbol{t}$ directly. In the paradigm with which we frame the problem, the object of interest is the *depth-time mapping.* We consider the layer boundaries simply as yearly-spaced samples from the depth-time mapping.

As input, GPCore requires $\mathcal{D}_{\text{core}} = \{Y, \boldsymbol{d}, \boldsymbol{d}_m, \boldsymbol{t}_m\}$, as well as the age associated with the first proxy sample (possibly equal to the time of drilling), t_0. Although the subscript m in $\{d_m, t_m\}$ denotes 'manual' for the input manual layer counts, these could arise from an automated layer detection procedure like the Wheatley et al. (2012) algorithm. The $\{d_m, t_m\}$ data is only used for initialisation of GPCore's optimisation procedure, as will be discussed in this chapter. GPCore's output is a monotonically increasing chronology vector $\boldsymbol{t}_{GPC}$, with estimates of the ice's age at each input depth location in $\boldsymbol{d}$.

GPCore's graphical model is shown in Fig. 2.1, where the cells represent random variables and a shaded cell implies that the variable is observed rather than unknown. This graphical model encodes our conditional independence (CI) assumptions that each proxy value is independent of the depth it was sampled from and all other proxies given the time associated with the proxy measurement, or $y_i \perp d, \{y_j\}_{j \neq i} | t$. While there may indeed be causal links between the different proxy variables and the proxy values may depend on depth (via a section of the ice core having more noise, for example), these CI assumptions are necessary simplifications for the inference to be tractable given the limited available data. The stochastic procedure assumed to generate the dataset is that firstly each of the functions mappings $f_{t|d} : \mathbb{R} \to \mathbb{R}$ and $\{f_{y_i|t} : \mathbb{R} \to \mathbb{R}, \quad i = 1, \ldots, N_y\}$ are drawn from some set of distributions over functions. Then, $\boldsymbol{t}$ is sampled noise-free from $f_{t|d}$ at the depth inputs $\boldsymbol{d}$ and each of the $\boldsymbol{y}_i$ are sampled (with Gaussian noise) from the $f_{y_i|t}$.

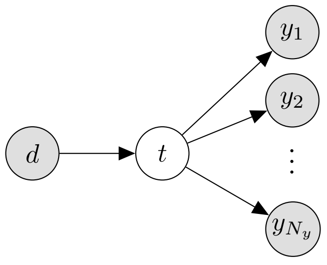


Fig. 2.1 GPCore's graphical model.

We place zero-mean GP priors over each of the functions, such that

$$f_{t|d}(d) \sim \mathcal{GP}(0, k_{t|d}(d, d')), \tag{2.14}$$
$$f_{y_i|t}(t) \sim \mathcal{GP}(0, k_{y_i|t}(t, t')), \quad i = 1, \ldots, N_y. \tag{2.15}$$

At the time of writing, we embed our prior assumptions about the phenomena that generated $\mathcal{D}_{\text{core}}$ in the following covariance function structures employed in GPCore

$$k_{t|d} = k_{\text{Lin}} + k_{\text{EQ}}^{(1)} + k_{\text{EQ}}^{(2)}, \tag{2.16}$$
$$k_{y_i|t} = k_{\text{Per}} \times k_{\text{EQ}}^{(3)} + k_{\text{Delta}}, \quad i = 1, \ldots, N_y. \tag{2.17}$$

Note that the covariance matrices for each of the $y_i|t$ GPs need not be tied together as in 2.17 and we do so merely for notational simplicity. Each of the kernel components encode our assumptions and model ice core phenomena as follows:

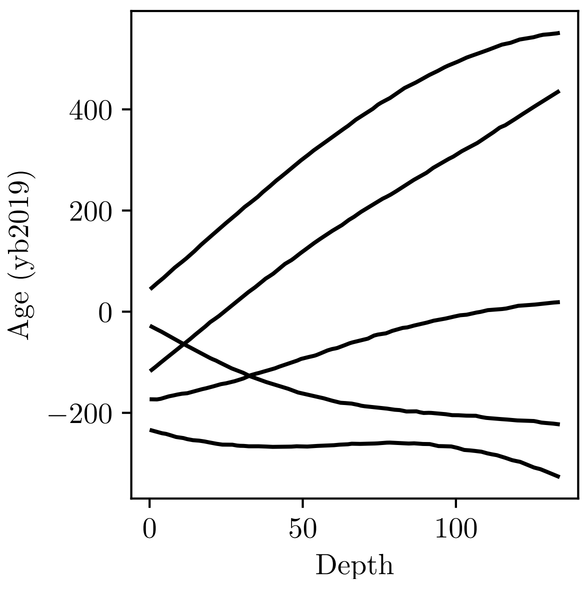
- k_{Lin}: Models the mean layer thickness at a particular ice core site.
- $k_{\text{EQ}}^{(1)}$ (long length scale): Models large-scale deviation from the mean layer thickness. For example, the ice sheet is subject to some degree of ice flow, resulting in the thinning of annual layers.
- $k_{\text{EQ}}^{(2)}$ (short length scale): Models short-scale deviation from the mean layer thickness. The amount of precipitation at the site of ice core drilling varies gradually due to sub- and super- annual climate processes, leading to some local warping in the depth-time mapping.
- $k_{\text{Per}} \times k_{\text{EQ}}^{(3)}$: The proxy variables are periodic with a period of one year, but decorrelate over time due to the changing climate. We assume that the proxy signals have been detrended so that a zero-mean assumption is appropriate.
- k_{Delta}: The proxy measurements are subject to random noise.

To obtain an intuition for the appearance of depth-time mappings and proxy time series that would have high probability under these GP priors, in Fig. 2.2 we plot a set of samples from both the $t|d$ and the $y_i|t$ GP priors for a particular setting of their hyperparameters. Fig. 2.2a illustrates that the $t|d$ GP prior can generate non-monotonically increasing depth-time mappings. This might appear to invalidate the model. However, the optimisation procedure is constrained to only search for monotonic mappings, as will be explained shortly.

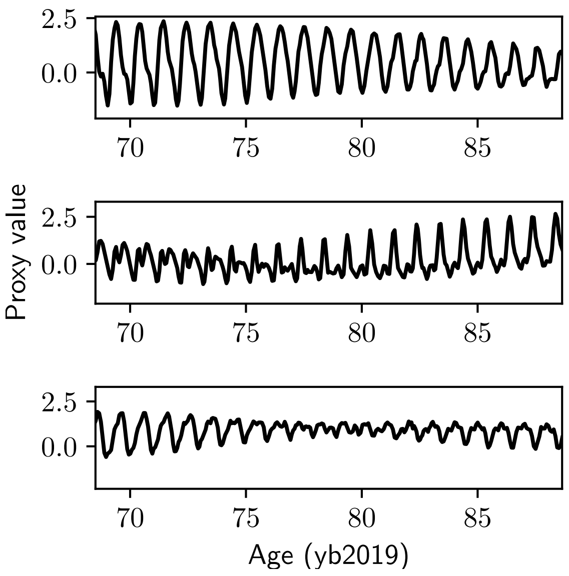
One issue with the model for $y_i|t$ is that it does not encode fixed phase in the periodic proxy time series because the Per×EQ covariance structure allows for decorrelation in the periods. This has the effect of GPCore's inferred proxy time series being able to drift out of phase with the seasons – i.e. a peak in the summer months is able to drift to the winter months without a penalty in the NLML. Fig. 2.2b highlights this issue; the periods in the middle sample are able to shift from a double peak in the middle of the year to a single peak towards the end of the year. This behaviour would not be observed in real proxy data – the value should always go 'up' in the summer months and 'down' in the winter months, for example. Future work with GPCore will investigate embedding fixed periodic structure in the proxy model, one option being

$$y_i(t) = f_{\mathrm{EQ}(\sigma)}(t) \times f(t) + f_{\mathrm{EQ}(\mu)}(t), \tag{2.18}$$

where $f(t)$ is a fixed periodic function of time such as $\sin(2\pi t)$. The amplitude and offset of the proxy signal are controlled by $f_{\mathrm{EQ}(\sigma)}(t)$ and $f_{\mathrm{EQ}(\mu)}(t)$ – both assigned EQ GP priors. This would present a non-parametric approach to the proxy model used in Wheatley (2015), as outlined in Section 1.3.3.



(a) Samples from $t|d$.



(b) Samples from $y_i|t$.

Fig. 2.2 Example samples from the GP priors used in GPCore.

The σ_f hyperparameter for $k_{\mathrm{EQ}}^{(2)}$ can be increased to make the depth-time mappings less locally smooth. This would allow the model to capture sharper changes in the depth-time mapping, for example due to extreme precipitation events. However, we warn that we have found that, in practice, such added flexibility in the model is liable to produce non-smooth artefacts in the inferred depth-time mapping, as demonstrated in the result of a simulated test[5] (where ground truth is available) shown in Fig. 2.3. This emphasises the need to use a noise-free $t|d$ model to constrain the search to smooth chronologies.

[5]Details of how the simulations were set up are given in Section 3.1.

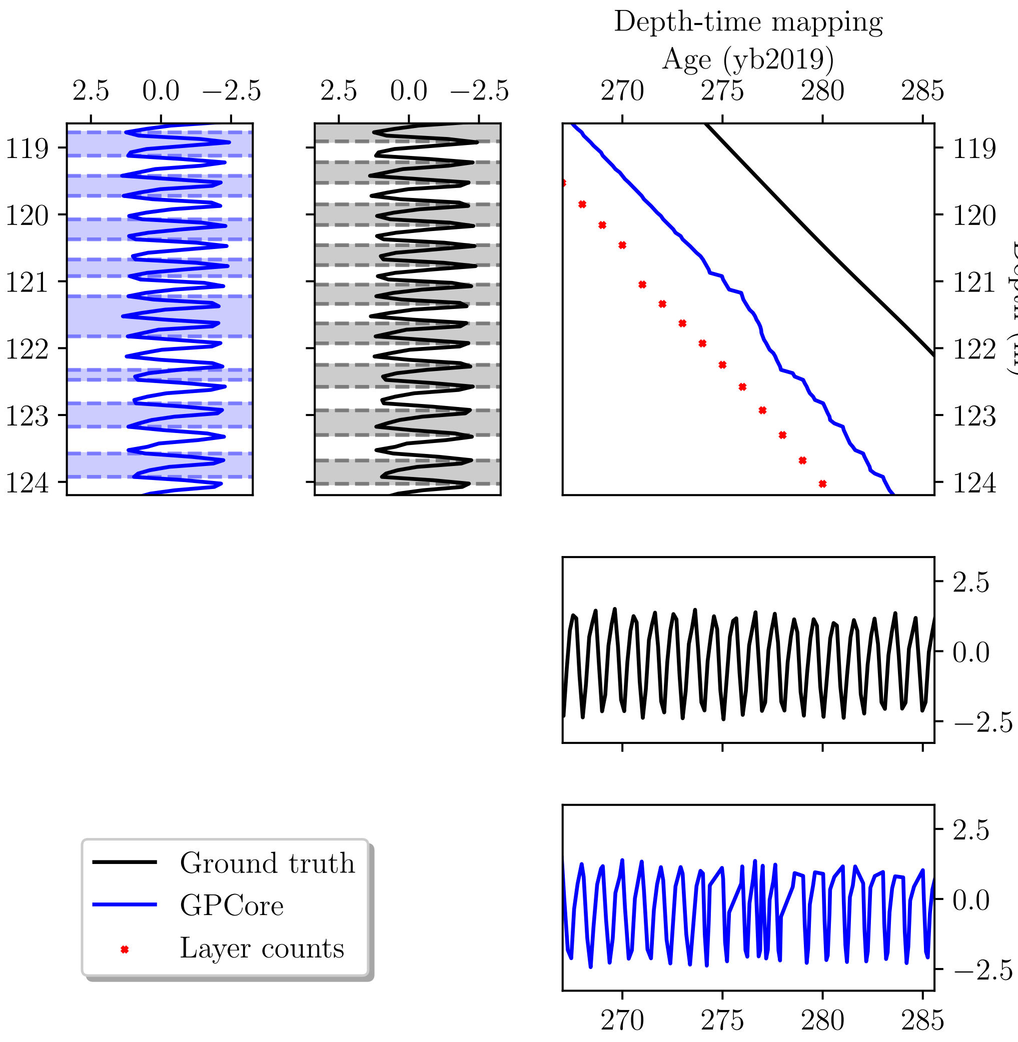


Fig. 2.3 Non-smooth artefacts in GPCore's inferred depth-time mapping in a simulated test when the $t|d$ GP prior is not sufficiently smoooth. The inferred and ground truth layer boundaries are shown in the top left in blue and black, respectively. The top right plot shows the inferred and ground truth depth-time mapping in blue and black respectively, and the simulated input manual layer counts are shown as red dots. The inferred and ground truth proxy time series are shown in the bottom two plots.

When sampled at the observed data locations, the model for the dataset $\mathcal{D}_{\text{core}}$ becomes

$$p(\boldsymbol{t}|\boldsymbol{d}) = \mathcal{N}(\boldsymbol{t}; \boldsymbol{0}, k_{t|d}(\boldsymbol{d}, \boldsymbol{d})), \tag{2.19}$$

$$p(\boldsymbol{y}_i|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{y}_i; \boldsymbol{0}, k_{y_i|t}(\boldsymbol{t}, \boldsymbol{t})) \tag{2.20}$$

Where the notation $k_{t|d}(\boldsymbol{d}, \boldsymbol{d}))$ denotes the covariance matrix formed when $k_{t|d}$ is sampled at $\boldsymbol{d}$. We then pose the inference task as to obtain the maximum-a-posteriori (MAP) estimate $\boldsymbol{t}^*$ for the proxy time points $\boldsymbol{t}$ by maximising the posterior over the time values $p(\boldsymbol{t}|\mathcal{D}_{\text{core}})$ – which we call the *chronology posterior* – to obtain an objective function in terms of the marginal likelihoods of the GP components

$$
\begin{aligned}
\boldsymbol{t}^* &= \underset{\boldsymbol{t}}{\operatorname{argmax}} \; p(\boldsymbol{t}|\mathcal{D}_{\text{core}}), && (2.21)\\
&= \underset{\boldsymbol{t}}{\operatorname{argmax}} \; \log p(\boldsymbol{t}|\mathcal{D}_{\text{core}}), && (2.22)\\
&\overset{(a)}{=} \underset{\boldsymbol{t}}{\operatorname{argmax}} \; \log p(\boldsymbol{t}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_y}, \boldsymbol{d}), && (2.23)\\
&\overset{(b)}{=} \underset{\boldsymbol{t}}{\operatorname{argmax}} \; \log p(\boldsymbol{t}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_y}|\boldsymbol{d}), && (2.24)\\
&\overset{(c)}{=} \underset{\boldsymbol{t}}{\operatorname{argmax}} \; \log p(\boldsymbol{t}|\boldsymbol{d}) p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_y}|\boldsymbol{t}, \boldsymbol{d}), && (2.25)\\
&\overset{(d)}{=} \underset{\boldsymbol{t}}{\operatorname{argmax}} \; \log p(\boldsymbol{t}|\boldsymbol{d}) \prod_{i=1}^{N_y} p(\boldsymbol{y}_i|\boldsymbol{t}), && (2.26)\\
&\overset{(e)}{=} \underset{\boldsymbol{t}}{\operatorname{argmin}} \left\{ -2\log p(\boldsymbol{t}|\boldsymbol{d}) - 2\sum_{i=1}^{N_y} \log p(\boldsymbol{y}_i|\boldsymbol{t}) \right\}, && (2.27)\\
&\overset{(f)}{=} \underset{\boldsymbol{t}}{\operatorname{argmin}} \; \left\{ \boldsymbol{t}^T k_{t|d}(\boldsymbol{d}, \boldsymbol{d})^{-1} \boldsymbol{t} + \log |k_{t|d}(\boldsymbol{d}, \boldsymbol{d})| + \sum_{i=1}^{N_y} \left(\boldsymbol{y}_i^T k_{y_i|t}(\boldsymbol{t}, \boldsymbol{t})^{-1} \boldsymbol{y}_i + \log |k_{y_i|t}(\boldsymbol{t}, \boldsymbol{t})| \right) \right\}, && (2.28)\\
&= \underset{\boldsymbol{t}}{\operatorname{argmin}} \; \mathcal{L}(\boldsymbol{t}), && (2.29)
\end{aligned}
$$

where $\mathcal{L}(\boldsymbol{t})$ can be exponentiated to give the unnormalised chronology posterior. The reasoning behind the algebraic steps above are: (a) in GPCore, the manual counts $(\boldsymbol{d}_m, \boldsymbol{t}_m)$ in $\mathcal{D}_{\text{core}}$ are not included in the statistical model and so are dropped here; (b) follows from Bayes' rule and the fact that 2.23 and 2.24 differ only by a constant factor that is independent of $\boldsymbol{t}$; (c) using Bayes' rule to break up the joint over time and proxies; (d) using Bayes' rule again with our CI assumption of $y_i \perp d, \{y_j\}_{j \neq i} | t$; (e) switching from maximisation to minimisation and scaling the objective by 2 to cancel with the factor of $\frac{1}{2}$ in the NLML from 2.7 and (f) using the result from 2.7. Note that in 2.28 we do not include the observation noise term $\sigma_n^2 I$ in the covariance matrices because we choose to treat the observation noise internally within covariance functions using k_{Delta} components (which has the same effect as adding a diagonal matrix).

The objective function $\mathcal{L}(\boldsymbol{t})$ that we seek to minimise to obtain the MAP estimate is

$$\mathcal{L}(\boldsymbol{t}) = \boldsymbol{t}^T k_{t|d}(\boldsymbol{d}, \boldsymbol{d})^{-1}\boldsymbol{t} + \log|k_{t|d}(\boldsymbol{d}, \boldsymbol{d})| + \sum_{i=1}^{N_y}\left(\boldsymbol{y}_i^T k_{y_i|t}(\boldsymbol{t}, \boldsymbol{t})^{-1}\boldsymbol{y}_i + \log|k_{y_i|t}(\boldsymbol{t}, \boldsymbol{t})|\right), \quad (2.30)$$

Intuitively, the MAP optimisation can be thought of as shifting the proxy time values until the proxy looks sufficiently periodic in time with period 1, while enforcing a smooth depth-time mapping. In this project, we use the Python package *stheno* to perform inference in GPs (Bruinsma and Tebbutt, 2018).

Note that although $\mathcal{L}(\boldsymbol{t})$ arises from the posterior over proxy time points and comprises a sum of individual NLMLs, we will refer to it simply as 'the NLML'.

2.4.1 An Alternative Statistical Model for GPcore

The reader might have noticed that the generative model shown in Fig. 2.1 suggests that time is generated from depth, which is obviously against the arrow of causality. Although this is not strictly a flaw in GPCore's model, we also considered the alternative graphical model shown in Fig. 2.4 that does abide by $t \to d$ causality.

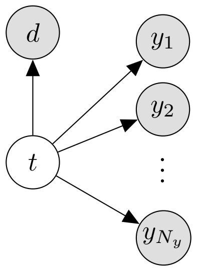


Fig. 2.4 An alternative statistical model that abides by time → depth causality.

This model is more challenging to work with for inferring $\boldsymbol{t}$, because the MAP objective becomes

$$\boldsymbol{t}^* = \underset{\boldsymbol{t}}{\operatorname{argmax}}\ p(\boldsymbol{t}, \mathcal{D}_{\text{core}}), \quad (2.31)$$

$$= \underset{\boldsymbol{t}}{\operatorname{argmax}}\ \log\left(p(\boldsymbol{t})p(\boldsymbol{d}|\boldsymbol{t})\prod_{i=1}^{N_y} p(\boldsymbol{y}_i|\boldsymbol{t})\right), \quad (2.32)$$

which requires a prior to be placed over the time values $p(\boldsymbol{t})$. One option that has been considered is an HMM with lognormal transition probabilities, such that

$$p(\boldsymbol{t}) = p(t_1)\prod_{j=2}^{N} p(t_j|t_{j-1}), \tag{2.33}$$

$$= p(t_1)\prod_{j=2}^{N} \ln \mathcal{N}(t_j - t_{j-1}; \mu_t, \sigma_t^2), \tag{2.34}$$

The advantage of such a prior is that if μ_t and σ_t^2 happen to agree with the ground truth time vector $\boldsymbol{t}_{gt}$, then the $p(\boldsymbol{t})$ term can be used to guide the MAP optimisation towards the ground truth time points. In the next chapter we will explore how this phenomenon can be exploited in a non-statistical manner to constrain the optimisation during simulated tests of GPCore where $\boldsymbol{t}_{gt}$ is known. However, in the case of real ice core chronology inference problems where $\boldsymbol{t}_{gt}$ is not known, selection of an appropriate prior is a challenge and is even more difficult if the proxies are sampled non-uniformly in depth. Hence the previous statistical model whose graphical model is shown in Fig. 2.1 was selected as the GPCore model.

2.5 The GPCore Optimisation Procedure

The optimisation problem in 2.30 turns out to be extremely challenging. This is partly because the optimisation variable $\boldsymbol{t}$ is the *input* for the $y_i|t$ component and so performing gradient descent on $\mathcal{L}(\boldsymbol{t})$ costs $\mathcal{O}(N^3)$ due to the matrix inversion/gradient calculation. Thus the optimisation problem has a significant computational cost for long ice cores with more than a few thousand proxy samples. This section will explore some of the heuristic design choices that have been taken to develop a tractable optimisation procedure that addresses the challenges associated with the problem.

Before outlining the structure of GPCore's optimisation procedure (as of the end of this project), it should be stressed that the algorithm's performance depends on a number of parameters, denoted θ_{GPC}, whose most appropriate values for a given ice core chronology inference problem will depend on the characteristics of the dataset and prior knowledge about the ice core. This section is not intended to provide definitive answers for what values θ_{GPC} should take, but offer some guidance on how to choose them based on the experiments conducted thus far. The overall design arose from a series of iterative refinements based on performance evaluation experiments with simulated ice core datasets, which will be detailed in the next chapter. However, some of the design choices are yet to be rigorously tested and GPCore is by no means at the stage of a final product.

Implementation Details

We perform gradient descent with the L-BFGS-B algorithm (Liu and Nocedal, 1989) and use the Python package *autograd* to implement automatic differentiation (Maclaurin et al., 2015) when computing gradients, which achieves a vast increase in speed compared with Python's *scipy.minimize* function.

Parameterised Time Differences as the Optimisation Variable

To constrain the search to $\boldsymbol{t}$ solutions that increase monotonically with depth, we instead optimise over time *differences* $\boldsymbol{t}_\delta := [t_2 - t_1, t_3 - t_2, \ldots, t_N - t_{N-1}]$ rather than raw time values $\boldsymbol{t}$. In the transformation from $\boldsymbol{t}$ to $\boldsymbol{t}_\delta$, we store t_1 to allow for a one-to-one mapping between $\boldsymbol{t}$ and $(t_1, \boldsymbol{t}_\delta)$, and assume that the value for the first proxy time sample t_1 is observed from the time of drilling of the ice core. t_1 is thus required as an additional input to GPCore as well as $\mathcal{D}_{\text{core}}$. Enforcing monotonicity in the inferred depth-time mapping is thus a matter of constraining the time differences to be positive. To achieve this, we re-parameterise $\boldsymbol{t}_\delta$ using a log-transformation to obtain the actual optimisation variable, $\boldsymbol{\tau}_\delta$, whose elements relate to those of $\boldsymbol{t}_\delta$ via

$$t_{\delta,j} = \log(1 + e^{\tau_{\delta,j}}), \tag{2.35}$$

$$\tau_{\delta,j} = \log(e^{t_{\delta,j}} - 1), \tag{2.36}$$

and the transformation is taken into account when computing gradients w.r.t. $\boldsymbol{t}_\delta$. The L-BFGS-B box constraint bounds are set to (-30,30) for each of the $\tau_{\delta,j}$ elements to guard against numerical underflow or overflow caused by $e^{\tau_{\delta,j}}$ being very large or very small. The -30 lower bound on $\tau_{\delta,j}$ is of particular importance, as it was observed in simulated trials that GPCore sometimes attempts to make the time differences between samples very small. This lower bound corresponds to restricting time differences to be greater than 3 microseconds and so $\boldsymbol{t}_{gt}$ will almost certainly lie within the feasible region created by this box constraint.

Chunking and Initialisation

When we first started to tackle the problem in the early stages of the project, we naïvely performed inference over the entire ~2,600-dimensional ice core dataset that we were using. The optimisation with this approach was very slow. Furthermore, we initialised the whole time vector $\boldsymbol{t}$ linearly based on the number of counted layers in the core. This is a poor approach because the gradual thinning of layers over ice cores means that linear estimates of the depth-time mapping are only valid over short depth ranges. In order for GPCore to fit to the ground truth depth-time mapping, it is of crucial importance that the initialisation is sufficiently close to ground truth to prevent the optimisation from falling into local minima, which motivates the question of how to treat the initialisation of $\boldsymbol{t}_\delta$.

As discussed, optimising directly over the $(N-1)$-dimensional vector $\boldsymbol{t}_\delta$ becomes intractable when N is large. In order to reduce the dimension to a manageable size, $\mathcal{D}_{\text{core}}$ is split into subsections, which can naturally be thought to correspond physically to *chunks* of the ice core and are henceforth referred to simply as 'chunks'. We denote these subsets $\mathcal{D}^{(c)}_{\text{core}}$ and use c to index the chunk. Rather than the high-dimensional objective function associated with the whole dataset, the lower-dimensional objective function used for optimisation is that associated with $\mathcal{D}^{(c)}_{\text{core}}$. We denote the chunk objective function as $\mathcal{L}_c(\boldsymbol{t}^{(c)})$, where $\boldsymbol{t}^{(c)}$ are the subset of time points associated with the proxy samples contained within the chunk. The chunk's first time point, $t_1^{(c)}$, is obtained from the previous chunk's solution.

The choice for the number of samples used for each chunk, N_c, involves two types of trade-off that have been observed in trials. Firstly, as the chunk size decreases, the algorithm duration required to produce an estimate for $\boldsymbol{t}$ first decreases due to the lower-dimensionality of the chunk objective functions, but eventually starts to increase due to the need to perform more optimisation runs. Thus there will be an optimal choice for N_c in terms of speed. Secondly, N_c needs to be large enough to contain a sufficient number of annual proxy cycles to ensure that $\mathcal{D}^{(c)}_{\text{core}}$ holds enough information about the depth-time mapping over the span of the chunk, but not so large that $\mathcal{L}_c(\boldsymbol{t}_c)$ starts to exhibit more local minima and stops being well-behaved. Hence N_c dictates the two competing effects of the algorithm's speed and its ability to perform inference, each with their own trade-off. For the trials run so far, a value of $N_c = 200$ has been used (where each chunk contains 5-10 layers) and has yielded good performance.

Another question is how to initialise the time differences for each chunk, $\boldsymbol{t}^{(c)}_\delta$. The method we adopt is to obtain an initial estimate for the time spanned by each chunk by using the number of input layer counts within the chunk. The number of layers are estimated to decimal accuracy by linearly interpolating between the layers that the top and bottom of the chunk lie within, as opposed to simply counting the number of full layers that lie within the depth range of the chunk. $\boldsymbol{t}^{(c)}_\delta$ is then initialised by linearly interpolating using $\boldsymbol{d}^{(c)}$ such that the sum of chunk time differences equals the number of counted layers in the chunk. This corresponds to an initial linear estimate for the depth-time mapping within each chunk. N_c should therefore not be so large that this linear initialisation is invalid due to the thinning of layers with depth. In general, the number of chunks will not divide the size of the dataset, and so the final chunk is truncated. Some care should be taken that this final chunk is not too small.

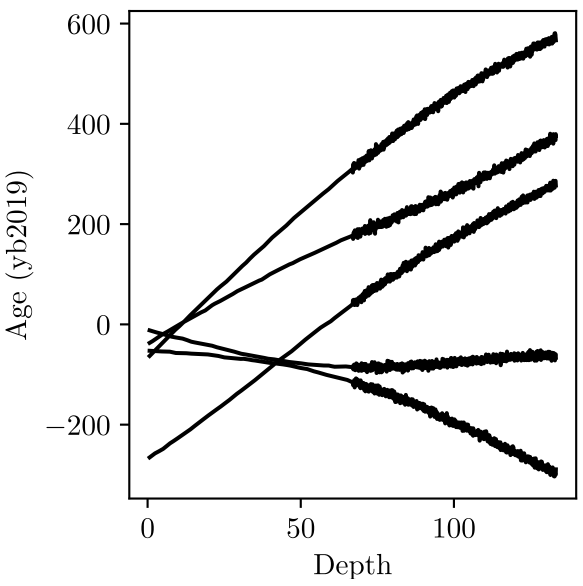
We stress that the input layer boundaries are only used for initialisation in the procedure outlined above; they do not form a part of the statistical model, unlike in StratiCounter.

Annealing Noise

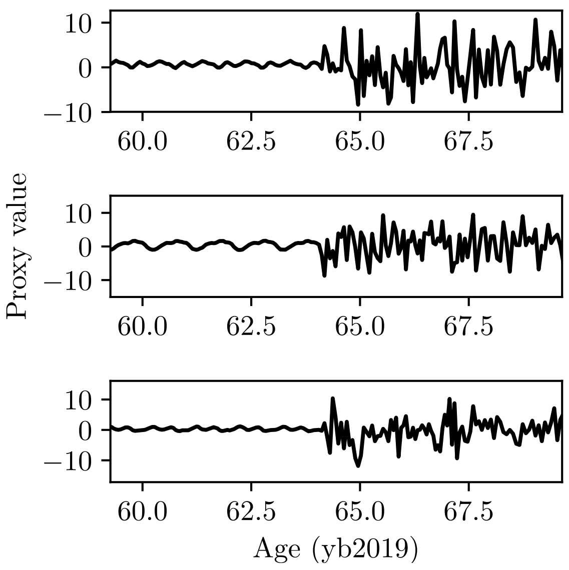
Due to the formulation of optimisation in terms of time differences, varying an earlier element of a chunk's time difference vector $\boldsymbol{t}_{\delta}^{(c)}$ will also vary the raw time values for each of the deeper samples. This can cause the optimisation to become stuck; adjusting a higher layer can cause the proxy cycles below to come out of phase with the annually-periodic cycle. This aliasing effect introduces local minima in the objective function and results from the periodic component of the $y_i|t$ models. Thus there is a need to fit the time points for the top of the chunk *before* those towards the bottom of the chunk. We achieve this by using a heteroscedastic noise model in each of the GP priors with a noise discontinuity that shifts along the chunk as the optimisation progresses, which we call the 'annealing phase' of each chunk's optimisation. With this model, elements of $\boldsymbol{t}^{(c)}$ and $\boldsymbol{y}^{(c)}$ under the noise discontinuity are subject to a step increase in their noise level. This is illustrated in Fig. 2.5 by drawing samples from the modified models.

The annealing noise has the effect of making the objective function less sensitive to the exact values of the $t_j^{(c)}$ and $y_{i,j}^{(c)}$ samples under the noise discontinuity, smoothing the objective function with respect to the samples subjected to annealing noise and allowing $\boldsymbol{t}^{(c)}$ to be fit to a growing fraction of the chunk as the annealing noise sweeps across. The chunk's objective function with respect to the later samples is revealed as the annealing noise discontinuity shifts.

Since the annealing noise has no statistical grounding and is only for guiding the optimisation, we do not run to convergence for each position of the annealing noise but instead use a finite user-defined number of L-BFGS-B samples before the discontinuity shifts again. Once the annealing noise has shifted beyond the end of the chunk, a 'homoscedastic phase' begins with no annealing noise, which the user can choose to run to convergence. The hope is that by the point that the homoscedastic phase is reached, $\boldsymbol{t}^{(c)}$ will correspond to proxy samples that are spaced with their cycles lasting roughly one year each, and their values need only be fine-tuned during the homoscedastic phase.



(a) Samples from $t|d$.



(b) Samples from $y_i|t$.

Fig. 2.5 Samples from the GPs with a heteroscedastic noise model applied as part of the annealing phase for each chunk's optimisation.

Forward-Backward Passes With Overlapping Chunks

We employ forward-backward (FB) passes to allow for information about the ice core's depth-time mapping to flow both up and down the ice core. The backward pass is performed by flipping the ice core data and processing the ice core from bottom to top. We then average the $\boldsymbol{t}_\delta^{(F)}$ and $\boldsymbol{t}_\delta^{(B)}$ to yield a combined estimate, $\boldsymbol{t}^{(FB)}$.

We also use overlapping chunks (OC) with an overlap of 50% between chunks to allow information to flow between neighbouring chunks during the individual forward and backward passes. $\boldsymbol{t}_\delta^{(c)}$ and $\boldsymbol{t}_\delta^{(c+1)}$ are combined by linear interlacing: the overlapping portion of chunk c is elementwise-multiplied with a vector whose elements drop linearly from 1 to 0, whilst the overlapping portion of chunk $c+1$ is elementwise-multiplied with a vector whose elements grow linearly from 0 to 1. These modified vectors are summed to obtain a new solution for the overlapping region.

Combining Multiple Passes

GPCore performs multiple FB passes. Using the superscript k to denote the kth pass, we initialise $\boldsymbol{t}_\delta^{(F,c,k)}$ at $\boldsymbol{t}_\delta^{(F,c,k-1)}$ for the forward pass and $\boldsymbol{t}_\delta^{(B,c,k)}$ at $\boldsymbol{t}_\delta^{(B,c,k-1)}$ for the backward pass. By performing multiple passes, each involving the interlacing of overlapping chunks and averaging of the forward and backward solutions, information about the depth-time mapping is allowed to traverse greater distances through the ice core. Put a different way, as k increases, every time difference $t_{\delta,j}^{(FB,c,k)}$ is influenced by a growing number of inferred time differences above and below. This allows the high-dimensional objective function to be explored whilst restricting the view of each optimisation to a low-dimensional chunk of $\mathcal{D}_{\text{core}}$.

Fine-Tuning Phase

We have also implemented the option for a fine-tuning (FT) phase, where optimisation is performed globally using the entire high-dimensional dataset. This requires that N is not prohibitively large with respect to the computational power available. The time differences for the FT phase, $\boldsymbol{t}_{\delta}^{(FT)}$, are initialised at the solution from the end of the series of FB passes. Due to the high computational cost associated with this global optimisation, each L-BFGS-B iteration takes significantly longer than that of the chunk optimisations. On a high-performance machine that we used for this project, each L-BFGS-B iteration took roughly 45 seconds for an ice core with 2600 proxy samples.

Thus the optimisation schedule is split into a chunk annealing phase (CAP) and a fine-tuning phase (FTP), but note that the FTP involves its own annealing noise pass for the aliasing reasons discussed previously. The solution from the CAP can be considered as a strong initial estimate for the global MAP optimisation performed in the FTP. However, we have found that a FT pass does not provide significant improvements to the MAP solution for $\boldsymbol{t}$ in trials with toy ice cores, as will be discussed in the next chapter.

Time Penalty/Extra-Smooth Hyperparameters

During the iterative refinement GPCore's design, there was a stage where GPCore would regularly produce depth-time mappings with highly non-smooth artefacts, an example of which is provided in Fig. 2.3. This undesirable property has been almost entirely mitigated through the inclusion of a feature that constrains the time differences during optimisation. One approach is to zero out or attenuate the $k_{\text{EQ}}^{(2)}$ component in the $t|d$ GP prior. This will assign low probabilities to depth-time mappings with non-smooth local artefacts. However, this can also reduce the ability of GPCore to fit to the local warping features of the depth-time mapping.

Another approach has been the use of a lognormal penalty on the time differences of exactly the same form as the $p(\boldsymbol{t})$ prior given in 2.34 (but without a probabilistic interpretation), which we call the 'time penalty' (TP). In simulations where ground truth is known, the lognormal parameters μ_t, σ_t^2 have been fit using maximum-likelihood on the ground truth time differences for each chunk, $\boldsymbol{t}_{\delta,gt}^{(c)}$, and has resulted in very strong performance.

Of course, in general, $\boldsymbol{t}_{gt}$ is not known and so the lognormal parameters have to be estimated in some other way. However, the use of ground truth provides an upper bound on the achievable performance using a TP, which shows that there is promise in this feature. It has been found that a value for the mean parameter μ_t can be estimated from the number of counted layers in the ice core, and this gave a similar performance to when μ_t was based on $\boldsymbol{t}_{\delta,gt}^{(c)}$ for each chunk c. Over longer cores experiencing significant layer thinning, it would be more appropriate to have μ_t vary with depth. An open problem remains as to how to choose the variance of the TP, and is non-trivial. Future work should explore the performance with other forms of TP.

We stress that the TP is only valid if the depth-time mapping is sampled sufficiently uniformly in depth. If there's a section of missing proxies that are not filled in with query points, the time penalty could prevent the model from inferring the large time difference over the missing section.

2.6 GPCore Uncertainty Estimates

GPCore is a Bayesian method that indirectly optimises over the chronology posterior $p(\boldsymbol{t}|Y, \boldsymbol{d})$. Though we do not provide chronology prediction uncertainty plots in this report, some work has been done in this regard. The NLML objective function $\mathcal{L}(\boldsymbol{t})$ that GPCore optimises over is simply a scaled and logged form of the chronology posterior $p(\boldsymbol{t}|Y, \boldsymbol{d})$, as derived between 2.21 and 2.29. Thus the posterior can readily be modelled by a Gaussian using the Laplace approximation. This is achieved by computing the Hessian H of $\mathcal{L}(\boldsymbol{t})$ w.r.t. $\boldsymbol{t}$ at $\boldsymbol{t}_{GPC}$, then inverting it to obtain the Gaussian approximation's covariance matrix Σ (the diagonal elements of which provide the marginal variance estimate for each age prediction). The issue with the Laplace approximation is that it cannot model the true multimodality of $p(\boldsymbol{t}|Y, \boldsymbol{d})$ due to the local minima introduced by uncertain layers. This multimodality is what causes inflating uncertainty with depth down the ice core. We have observed the Laplace approximation w.r.t. raw time values $\boldsymbol{t}$ to fail to capture this inflating uncertainty. Another option is to compute the Laplace approximation w.r.t. the parameterised *differenced* time values τ_δ (i.e. GPCore's optimisation variable), drawing samples from the resulting Gaussian and transforming those samples to raw time values. The resulting samples do not have the property that their mean is equal to $\boldsymbol{t}_{GPC}$ because they are positively-skewed by the transformation in 2.35. However, they are at least pinned to t_0 and their variance inflates with depth (because they result from the sum of an increasing number of random time differences).

Another issue with the Laplace approximation is that computing the Hessian can become prohibitively expensive for ice core datasets with many samples. For example, on a 64GB desktop with 16 CPUs and using *autograd* for efficient differentiation, computing a 2600×2600 Hessian took multiple hours.

Future work should investigate alternative ways to model the complicated posterior $p(\boldsymbol{t}|Y, \boldsymbol{d})$. An option that should capture its multimodality would be a sampling scheme, possibly using a Gaussian proposal distribution found from a Laplace approximation.

Chapter 3

Experimental Evaluation

After introducing the theoretical background and implementation details of GPCore in Chapter 2, we now proceed to evaluate the design choices made and compare the performance of GPCore with StratiCounter. A simulated or 'toy' ice core was generated for the purpose of evaluating the performance of the two algorithms (the procedure for this generation is given in Section 3.1). In these simulations, the ground truth time points $\boldsymbol{t}_{gt}$ are known, and this allows a performance metric to be defined. While there is no single optimal performance measure for the problem of chronology inference, we choose to measure closeness of fit between the estimated and ground truth chronologies with the 'error' $E(\boldsymbol{t}, \boldsymbol{t}_{gt})$ given by the mean absolute difference between $\boldsymbol{t}$ and $\boldsymbol{t}_{gt}$ in years, i.e.

$$E(\boldsymbol{t}, \boldsymbol{t}_{gt}) = \frac{1}{N} \sum_{j=1}^{N} |t_j - t_{gt,j}|. \tag{3.1}$$

A significant portion of this project involved the iterative improvement of GPCore's performance on simulated ice core datasets. We achieved this in a principled fashion by varying the GPCore algorithm parameters θ_{GPC} and measuring the mean and standard deviation of the error $E(\boldsymbol{t}, \boldsymbol{t}_{gt})$ from a loop over a set of simulated proxy series. It would be a mistake, however, to only consider the error with ground truth – the gold standard for comparing the inferred depth-time mapping with ground truth is simply to plot both mappings on (d, t) axes, which is a sanity-check that we frequently conducted throughout the design process.

We also tested the various design features outlined in Section 2.5 – the Python class we have written that implements GPCore allows for each of the features to be switched on or off and the effect on performance measured. The class-based, object-oriented approach in GPCore's implementation permits a different 'GPCore object' to be instantiated for each successive run, guaranteeing the independence between runs.

In this Chapter, we describe how the toy ice core dataset was generate using GPs. We then justify our design choices for GPCore based on tests with this toy ice core. We then compare GPCore with StratiCounter in several simulations and show that GPCore is able to outperform StratiCounter. Finally, we show the results of applying GPCore and StratiCounter to a real ice core dataset from BAS, and show that GPCore performs more strongly for this dataset also.

3.1 Generation a Simulated Ice Core Depth-time Mapping

We built the GPCore algorithm from scratch and its performance was initially relatively poor. We therefore chose to generate an 'easy' toy ice core dataset with significantly reduced statistical challenges compared with real ice core datasets. The logic behind starting out with such a simple dataset is that any human would be able to determine its layer boundaries with close to perfect accuracy, which we aimed to replicate with GPCore.

We firstly generated the simulated depth-time mapping with a regression approach by computing the posterior mean of the $t|d$ GP model for the layer counts associated with Ice Core 1 (shown in Fig. 1.3) using the predictive mean equation given by 2.12. This produced a simulated depth-time mapping that mimics the properties of a real ice core. Since the layer counts increase monotonically, this resulted in the posterior mean through the layer counts also increasing monotonically. Next, the mapping was sampled at regular 5cm intervals over roughly 130m, and the resulting vector of time points was scaled by a factor of 0.4 to yield the simulated ground truth time points, $\boldsymbol{t}_{gt}$. The scaling was performed so that the proxies would be sampled with high time-resolution and corresponds to simulating a high-accumulation ice core. To generate simulated proxy data, we drew samples from our $y_i|t$ prior at $\boldsymbol{t}_{gt}$, using a long time scale of 50 years for the EQ component driving the decorrelation of periods over time. We note that the lack of a fixed period in our proxy model, as discussed in Section **??**, means that the periods of our simulated proxy series decorrelate over time and thus their phase is not locked. This presents a small shortcoming in the tests described in this chapter. Future work should rerun simulated trials in a similar manner to the way described here, but using a $y_i|t$ prior that more closely models the fixed-phase behaviour of proxy data.

The properties of GPCore's average performance on this simulated ice core was determined by varying the random seed used to generate the toy proxy series. The simulations were further idealised by setting the hyperparameters used in GPCore's $y_i|t$ GP priors equal those that generated the proxies. Furthermore, the $d|t$ GP hyperparameters were trained on the ground truth $\boldsymbol{t}_{gt}$ samples by minimising the NLML over the $\{\boldsymbol{d}, \boldsymbol{t}_{\text{gt}}\}$ data. We call this toy ice core dataset with the ground truth manual layer counts included $\mathcal{D}^{(0)}_{\text{sim}}$.

3.1.1 Corrupt Simulated Ice Core 1: Missing Manual Layer Counts

Six of the ground truth layer boundaries from the generated depth-time mapping outlined in Section 3.1 were artificially deleted to simulate missed layers in the manual layer count data. We call this ice core $\mathcal{D}_{\text{sim}}^{(1)}$ and the resulting layer counts are shown in Fig. 3.1. This was the toy ice core used in the majority of trials for the iterative design improvement of GPCore by measuring the algorithm's ability to correct these missing layer count errors.

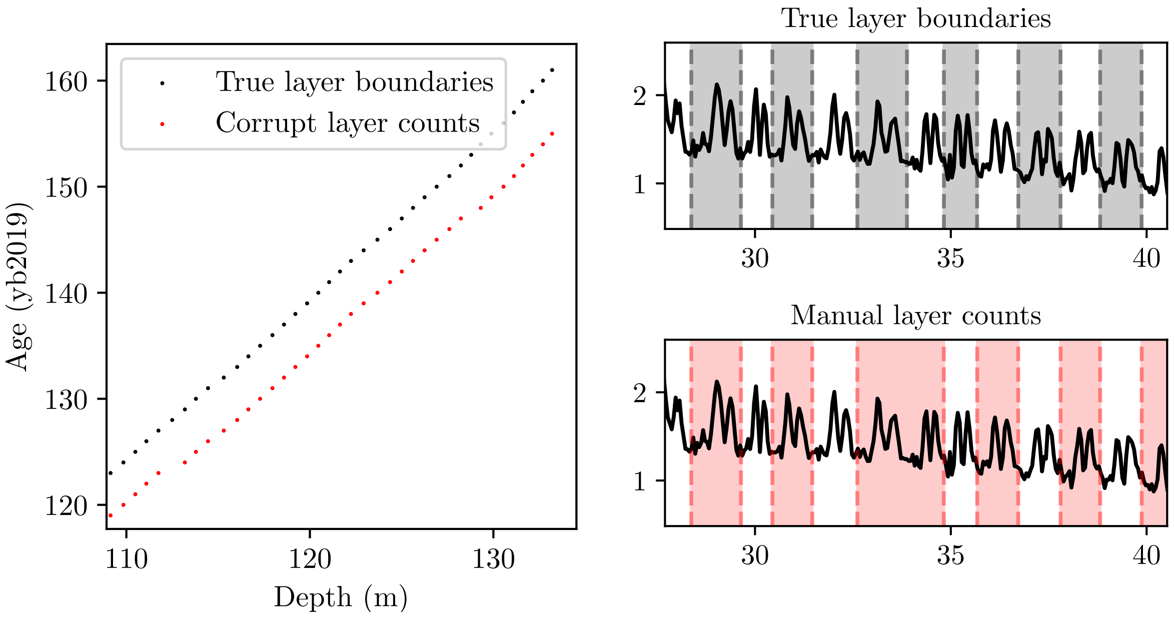
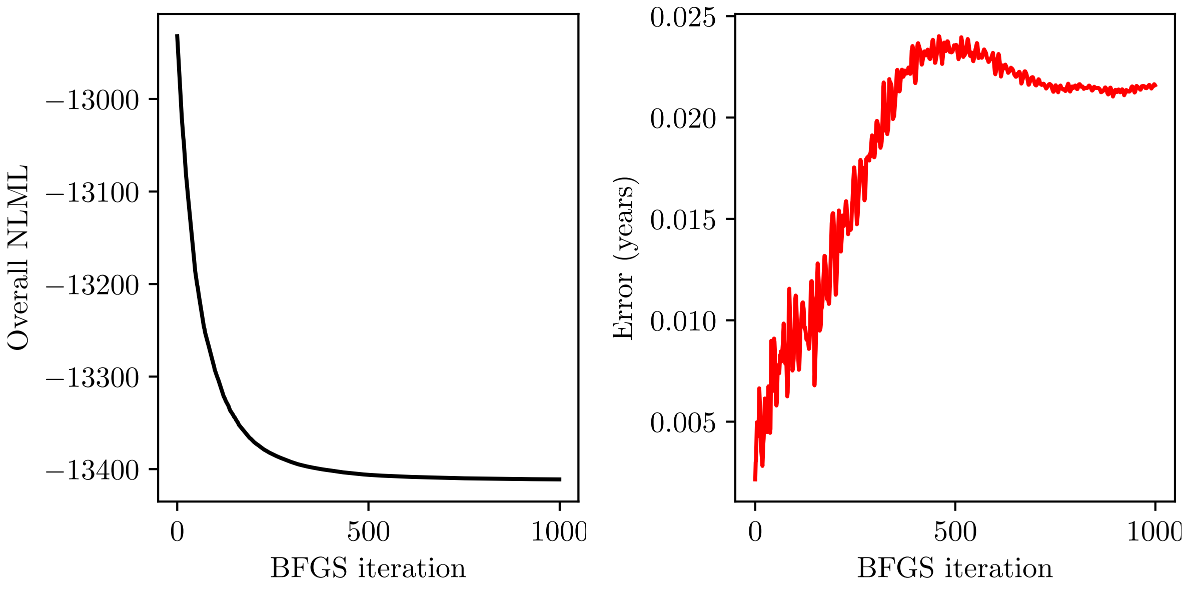


Fig. 3.1 Comparison of the true layer boundaries and the corrupt manual layer counts comprising the simulated ice core dataset $\mathcal{D}_{\text{sim}}^{(1)}$.

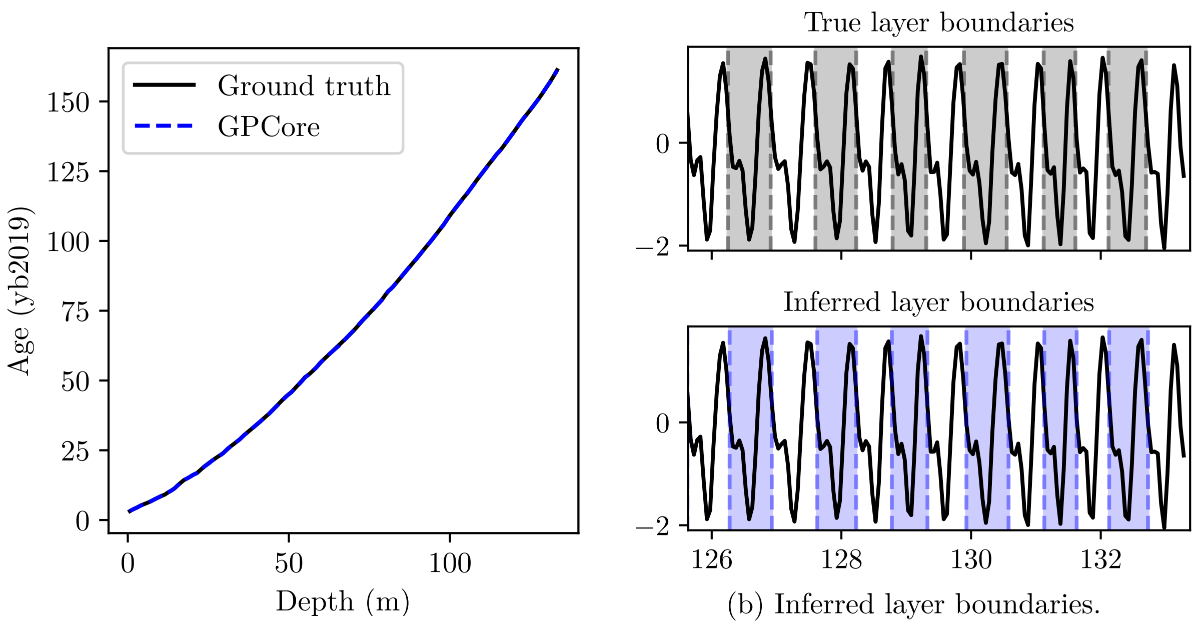
3.2 Ground Truth Stability Test

Before iteratively refining the optimisation algorithm, the statistical model was verified using the simulated depth-time mapping by initialising $\boldsymbol{t}$ at ground truth and performing a FT pass of GPCore using 1,000 L-BFGS-B iterations (without the annealing noise sweep) and observing how the solution was perturbed. This determines whether or not ground truth is a well-behaved local minimum of the NLML objective function when GPCore's statistical model matches the generative process. Fig. 3.2 verifies that $\boldsymbol{t}_{gt}$ is almost exactly a local minimum of the NLML. While the NLML drops from around -12,950 to -13,400 during the optimisation, this corresponds to a negligible error in the inferred mapping of only ∼0.022 years. Furthermore, the inferred mapping and layer boundaries remain almost exactly at ground truth. Thus ground truth is indeed a local minimum of the objective function. For this simulated ice core, the final NLML of -13,400 was the lowest observed throughout the project.



(a) Evolution of the NLML. (b) Evolution of the error.

Fig. 3.2 Evolution of the NLML and error with ground truth over 1,000 BFGS iterations after initialising at ground truth of the simulated ice core.



(a) Inferred depth-time mapping. (b) Inferred layer boundaries.

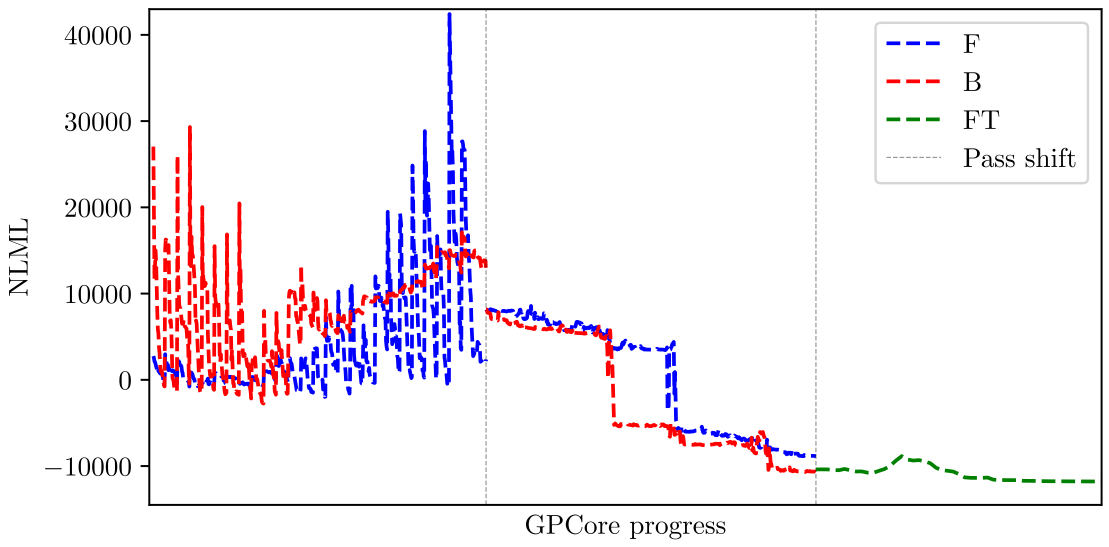
Fig. 3.3 Inferred depth-time mapping layer boundaries after 1,000 BFGS iterations and initialising at ground truth of the simulated ice core.

3.3 Illustrating the GPCore Algorithm

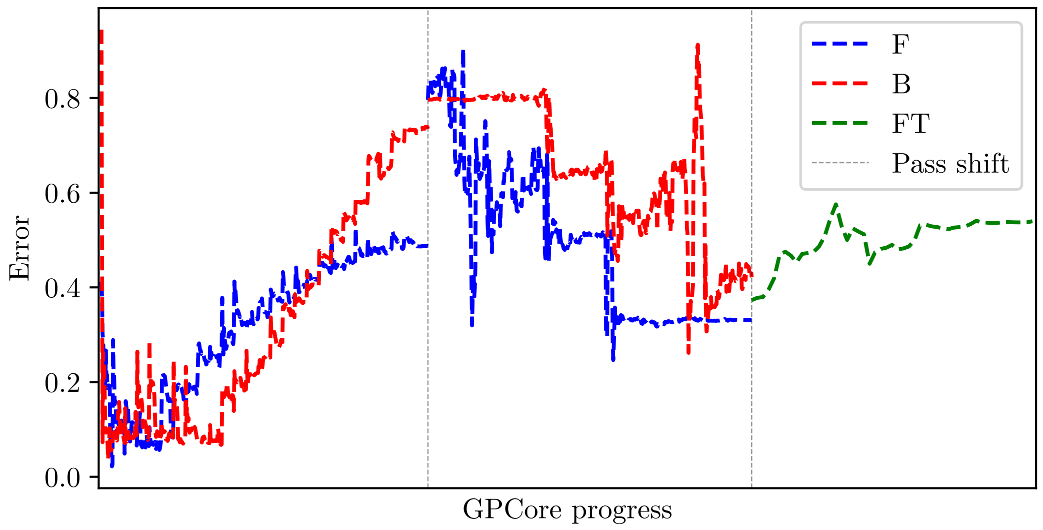
An important question is whether optimisation over low-dimensional chunks of $\mathcal{D}_{\text{core}}$ using the GPCore procedure outlined in Section 2.5 also corresponds to a reduction in the global NLML evaluated over the whole time vector $\boldsymbol{t}$. Another key concern is whether a lower NLML correlates with an inferred chronology that is closer to ground truth. We provide partial answers to these questions using the simulated dataset by tracking the evolution of the NLML and the error as the algorithm progresses, as shown in Fig. 3.4. This test used the toy ice core dataset $\mathcal{D}_{\text{sim}}^{(1)}$.

A subtlety of Fig. 3.4 is that during the first FB pass, the error $E(\boldsymbol{t}, \boldsymbol{t}_{gt})$ and the NLML $\mathcal{L}(\boldsymbol{t})$ are only computed over the time samples for which an estimate exists. This results in the illusion that the error starts very small and gets progressively worse as the first pass progresses, whereas the objectives used during the first pass actually change for each chunk, and are only comparable to later passes upon its completion. We also comment that during the first pass, the error and NLML of the backwards estimate is evaluated using reversed forms of $E(\boldsymbol{t}, \boldsymbol{t}_{gt})$ and $\mathcal{L}(\boldsymbol{t})$.

Fig. 3.4 illustrates all of the features of GPCore outlined in the previous section. For example, the spikes in the NLML and error during the first pass correspond to the revealing of chunks, which then drop as the chunk time points $\boldsymbol{t}^{(c)}$ are fit by GPCore. Also illustrated is the second F and B passes both initialising from the average of the first F and B pass, as well as the FT phase initialising at the solution found from averaging the second F and B passes. For this particular run, the FT does not reduce the NLML significantly and actually increases the error with ground truth.



(a) Evolution of the NLML.



(b) Evolution of the error.

Fig. 3.4 Evolution of the NLML and the error with $\boldsymbol{t}_{gt}$ as the GPCore algorithm processes $\mathcal{D}_{\text{sim}}^{(1)}$. Two FB CAP passes one FT pass were performed. Note that for the first pass, the NLML and error are only evaluated using the subsection of $\boldsymbol{t}$ that has been inferred.

3.4 Performance Comparison with StratiCounter on Simulated Datasets

With the performance metric of the error $E(\boldsymbol{t}, \boldsymbol{t}_{gt})$ defined, we may now justify some of GPCore's design choices and compare the performance of GPCore against StratiCounter. To convert StratiCounter's output of inferred layer boundaries to an estimate for the vector of time points $\boldsymbol{t}$, linear interpolation is performed between the depths and ages associated with the inferred layers.

3.4.1 Missing Manual Layer Counts: $\mathcal{D}^{(1)}_{\textbf{sim}}$

Fig. 3.5 shows results from a number of single-proxy datasets with missing manual layer counts, $\mathcal{D}^{(1)}_{\text{sim}}$, and compares the mean and standard deviation of the error $E(\boldsymbol{t}, \boldsymbol{t}_{gt})$ for four possible set-ups of the GPCore algorithm, contrasting each of these with a benchmark of the average performance using StratiCounter. The 'Base' trial shown consisted of a single FB pass with OC and only 3 BFGS iterations[1] performed at each chunk's annealing noise discontinuity position. The following three trials result from switching on or off one of the GPCore features w.r.t the Base trial. Turning our attention initially to these first four trials performed with GPCore, we see that very poor performance can occur if the smoothness of the depth-time mapping is not restricted using the TP or ES $t|d$ hyperparameter design options outlined in Section 2.5. We point out that the TP parameters μ_t, σ_t^2 were computed using the ground truth time differences $\boldsymbol{t}_{gt,\delta}$, and so performance for the TP trial should be considered as an upper bound on performance using a time penalty. Fig. 3.6 plots the solutions from the two runs with poor performance in the Base trial, characterising these wild artefacts as resulting from a small number of short sections that experience a dramatic increase in the number of layers that GPCore infers. The origin of such artefacts has not yet been determined and future work on GPCore should investigate and characterise these artefacts further. We point out, however, that when the trials with these artefacts are ignored, the performance of the Base, no OC and TP trials are quite similar, with the ES trial experiencing poorer average performance. GPCore achieves stronger average performance than StratiCounter over the no-artefact runs. Including all runs, only the TP and ES trials outperform StratiCounter. Despite poorer average performance, however, we comment that StratiCounter achieves very small errors more often than the GPCore trials, corresponding to the layer boundaries being inferred with near-perfect accuracy.

[1]We restricted the number of iterations here in order to speed up the trials to allow for a sufficient number of runs to determine average performance.

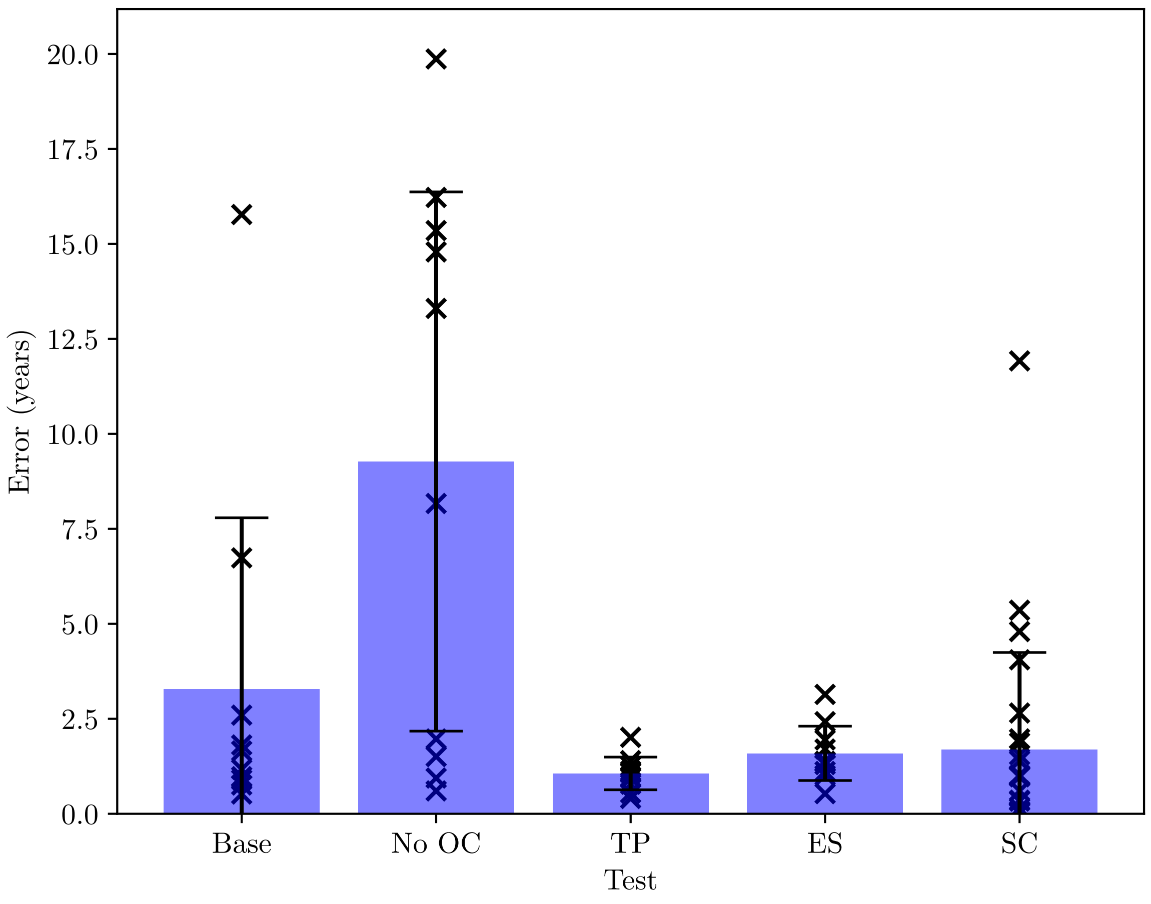


Fig. 3.5 Comparison of the mean and standard deviation of the error $E(\boldsymbol{t}, \boldsymbol{t}_{gt})$ for four possible set-ups of the GPCore algorithm using ten trials with different simulated proxy series for toy ice core dataset $\mathcal{D}_{\text{sim}}^{(1)}$. Here the 'Base' set-up for GPCore consisted of a single FB pass with OC and only 3 BFGS iterations completed at each chunk's annealing noise discontinuity position. Also shown is a benchmark determined from 25 of such single-proxy toy ice core datasets using the StratiCounter algorithm. The errors for the individual runs are shown as black crosses.

Fig. 3.7 plots the inferred depth-time mapping for the poorest run of StratiCounter with an error of +12.5 years, whereby extra layers are incorrectly inferred. It is possible that one could tweak the StratiCounter set-up to improve its performance here. However, this experiment shows that when used as a black box by a non-expert, StratiCounter's performance on our simulated datasets is generally poorer than that of GPCore and the algorithm is liable to exhibit wild artefacts in its solution of inferred layer boundaries.

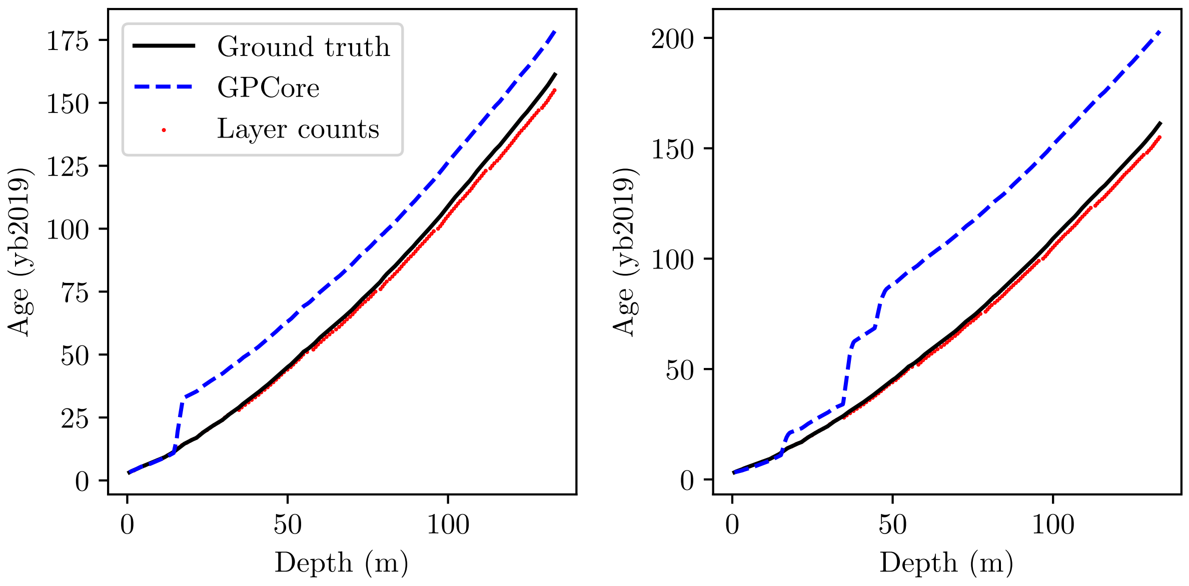


Fig. 3.6 Inferred GPCore depth-time mappings for the two runs with poorest performance from the ‘Base’ case in Fig. 3.5.

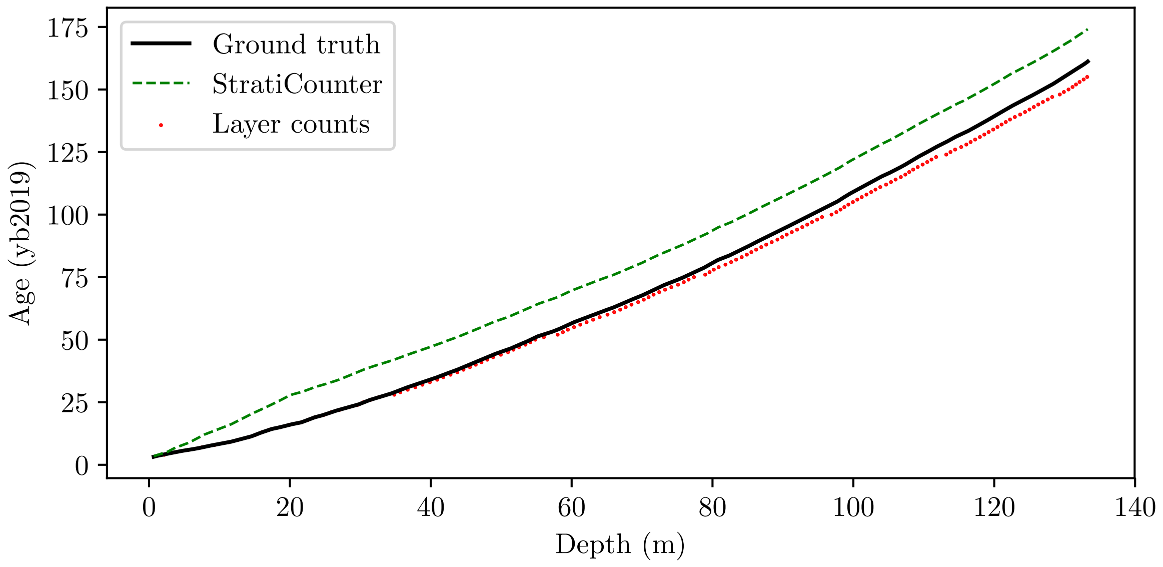
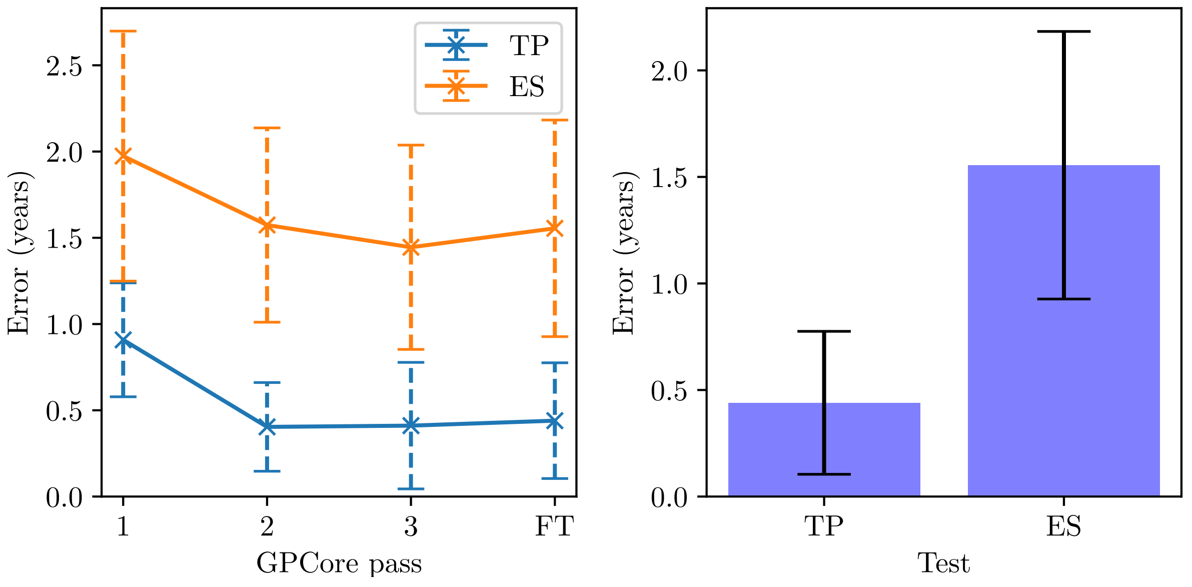


Fig. 3.7 Inferred StratiCounter depth-time mapping for the run with poorest performance in Fig. 3.5.

To observe the impact of multiple passes, the mean and standard deviation of the error (over ten runs) are plotted against the pass number for the TP and ES trials in Fig. 3.8a, where 3 FB passes and a final FT phase were performed. This shows that extra FB passes yield diminishing benefit and that a FT phase does not improve the average performance for either the TP or ES trials. Fig. 3.9 plots each of the individual runs for the TP trial whose errors were averaged to yield Fig. 3.8a. This shows that for most of the runs, the error decreases monotonically as the pass number increases, but a small number of outliers perturb the average statistics. If these outliers can be diagnosed then there is promise that GPCore could be modified to improve performance monotonically with further passes, as would be expected from a robust algorithm.



(a) Average performance vs. pass number. (b) Average performance after the FT pass.

Fig. 3.8 Comparison between the TP and ES trials using multiple FB passes and a FT phase.

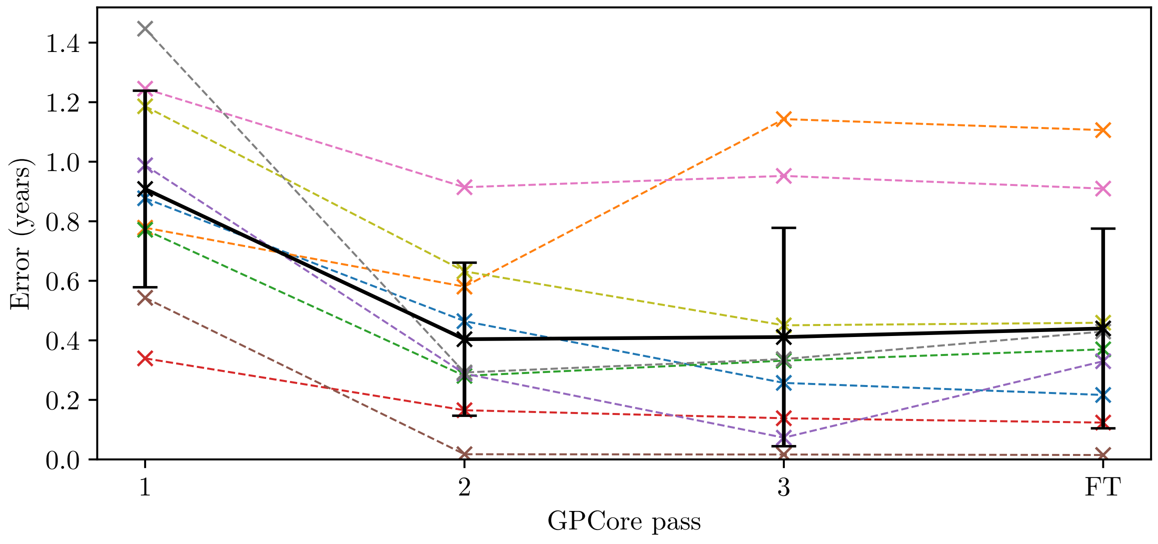


Fig. 3.9 Evolution of the error with pass number for each of the ten TP runs whose average performance are shown in Fig. 3.8a.

3.4.2 Missing Proxy Data: $\mathcal{D}_{\text{sim}}^{(2)}$

Toy dataset $\mathcal{D}_{\text{sim}}^{(2)}$ was generated by removing approximately 6 layers of proxy data from the middle of the toy ice core but, retaining the manual layer counts from $\mathcal{D}_{\text{sim}}^{(0)}$. One single-proxy trial was performed with this toy dataset for both GPCore and StratiCounter and Fig. 3.10 demonstrates how StratiCounter's solution of layer boundaries makes errors over the missing section, whilst GPCore's solution is unperturbed. Here we have not attempted to infer the depth-time mapping within the corrupted region, but we comment that GPCore's probabilistic model can readily be extended to perform inference over samples from the depth-time mapping that are not associated with a set of proxy samples. In this case, the (d_j, t_j) pairs without corresponding proxy samples will only contribute to the t|d component of the NLML.

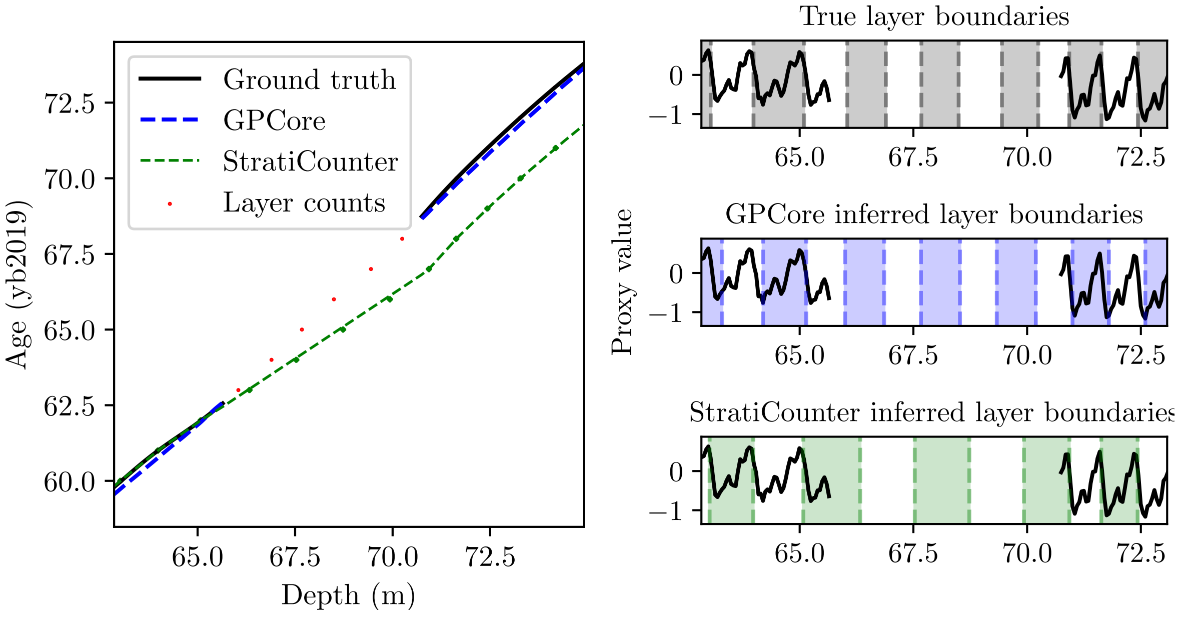


Fig. 3.10 Comparison between GPCore and StratiCounter on $\mathcal{D}_{\text{sim}}^{(2)}$.

3.4.3 Missing Proxy and Layer Count Data: $\mathcal{D}_{\textbf{sim}}^{(3)}$

A similar experiment to Section 3.4.2 was conducted, but this time with only 3 layers of proxy data removed. Here the layer counts over the section were also removed to yield $\mathcal{D}_{\text{sim}}^{(3)}$ and Fig. 3.11 shows GPCore's and StratiCounter's inferred solutions. Again, StratiCounter makes errors over the missing section while GPCore makes no errors. This and the previous test indicate that GPCore is able to robustly handle sections of missing data.

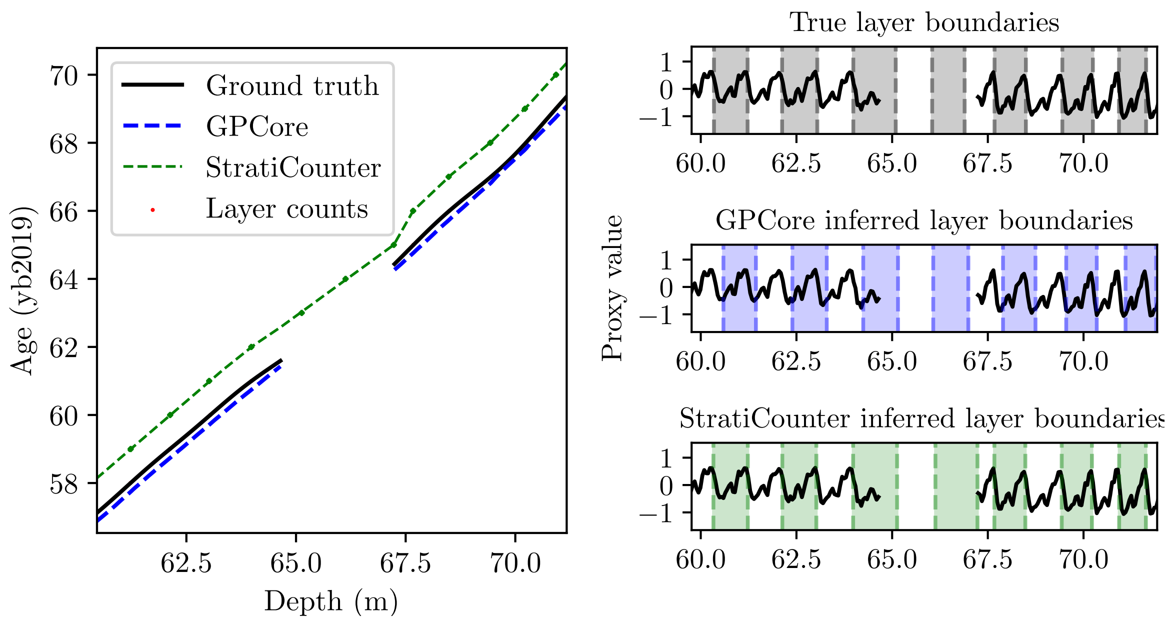


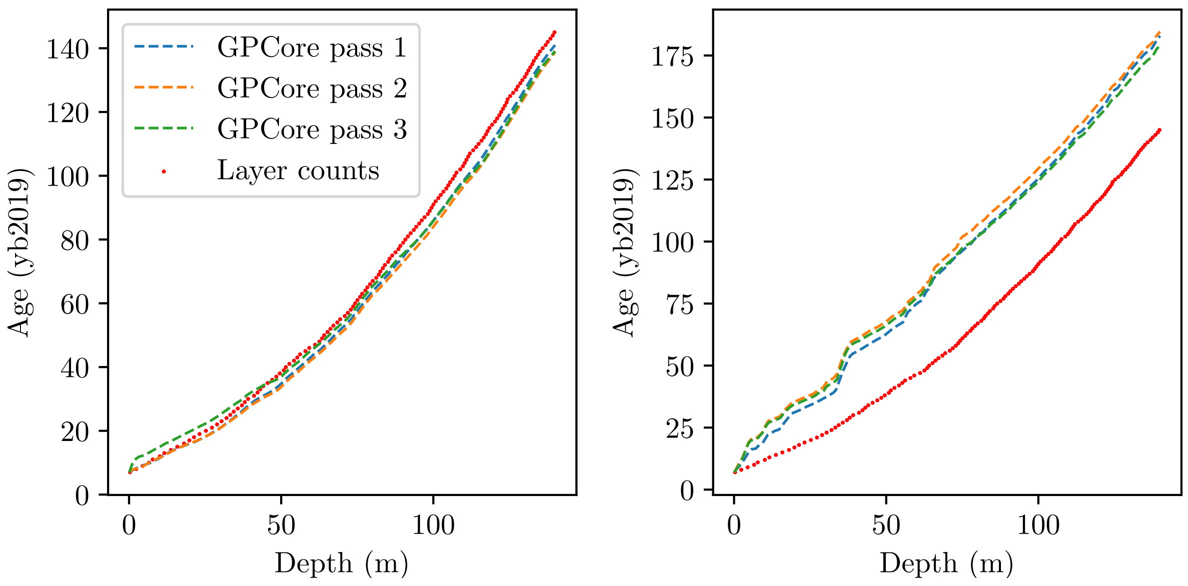
Fig. 3.11 Comparison between GPCore and StratiCounter on $\mathcal{D}_{\text{sim}}^{(2)}$.

3.5 Performance Comparison with StratiCounter on a Real Ice Core Dataset

We finally turn our attention to the dataset of real ice core measurements provided by BAS, consisting of 3 ice cores, each with 9 proxies measured. A challenge for both algorithms here is that almost all of the 27 proxy series in the dataset do not have clean annual signals. The oxygen isotope δ^{18}O displays the cleanest annual signal and BAS Ice Core 3 exhibits layers with the greatest thickness. Thus we chose the 'easiest' dataset for this experiment to be the one corresponding to the proxy series shown in the lower-right corner of Fig. 1.1 in Section 1.1. Although not tested in this project, we point out that in such cases where the annual cycles are less clear, including multiple proxy series could offer significant performance improvement.

In this test we assume that the manual layer counts are close to the ground truth layer boundaries of the ice core, since we presume the layer counts were determined by taking multiple proxy series into account. However, we now do not have access to the ground truth proxy time points $\boldsymbol{t}_{gt}$ and for this reason we do not attempt to use a TP. Furthermore, we do not immediately know an appropriate setting of the GP hyperparameters in GPCore. In this experiment we simply choose the hyperparameters manually from inspection of the dataset. In general, a principled procedure would be to train the $t|d$ hyperparameters on the (d_m, t_m) layer count data, then manually zero out the noise level to constrain the search to smooth d,t mappings. For the $\theta_{y|t}$ hyperparameters, one option would be to find an initial estimate the proxy time values $\boldsymbol{t}$ by linearly interpolating between the (d_m, t_m) manual count dataset and then training $\theta_{y|t}$ on the resulting proxy time series.

For this dataset, a two different settings for the GPCore hyperparameters were trialled. The first model was given ‘smoother’ hyperparameters with the second $\theta_{t|d}$ EQ component zeroed out (i.e. the same ES $t|d$ hyperparameters outlined in the previous sections). The $\theta_{y|t}$ hyperparameters were given a length scale of 10 years to reflect the greater degree of decorrelation between periods. For the second model, the 2nd EQ’s σ_f hyperparameter was increased to a value of 0.2 years, allowing for less smooth mappings, and $\theta_{y|t}$ was as for the first model but with an even shorter EQ length scale of 3 years. Fig. 3.12 plots GPCore’s inferred depth-time mappings after each of the 3 FB passes for these two models, and verifies that the smoother mappings produce more accurate inferred chronologies. Contrast GPCore’s inferred mapping with that of StratiCounter’s shown in Fig. 3.13 and we see that GPCore produces a more accurate depth-time mapping. Fig. 3.14 further illustrates this by comparing the inferred layers from the best performing GPCore run in Fig. 3.12a with the manual layer counts, also showing that StratiCounter generally infers layers that are too thin.



(a) Smoother hyperparameters.

(b) Less smooth hyperparameters.

Fig. 3.12 Inferred GPCore depth-time mappings after each FB pass of BAS Ice Core 3 using two different settings for the hyperparameters.

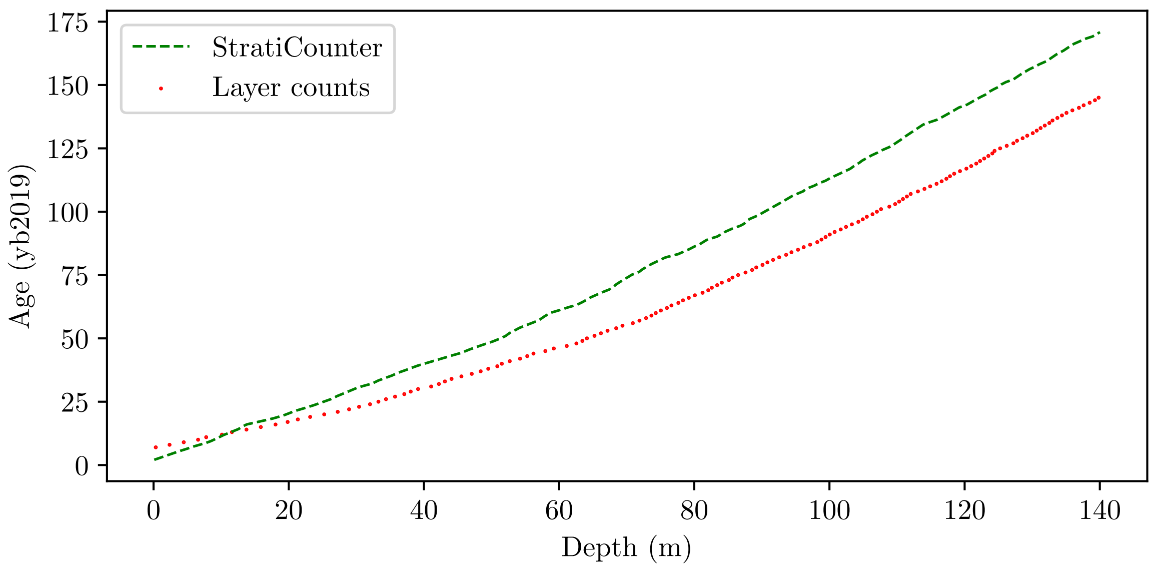


Fig. 3.13 StratiCounter's inferred depth-time mapping for BAS Ice Core 3 using the δ^{18}O proxy data.

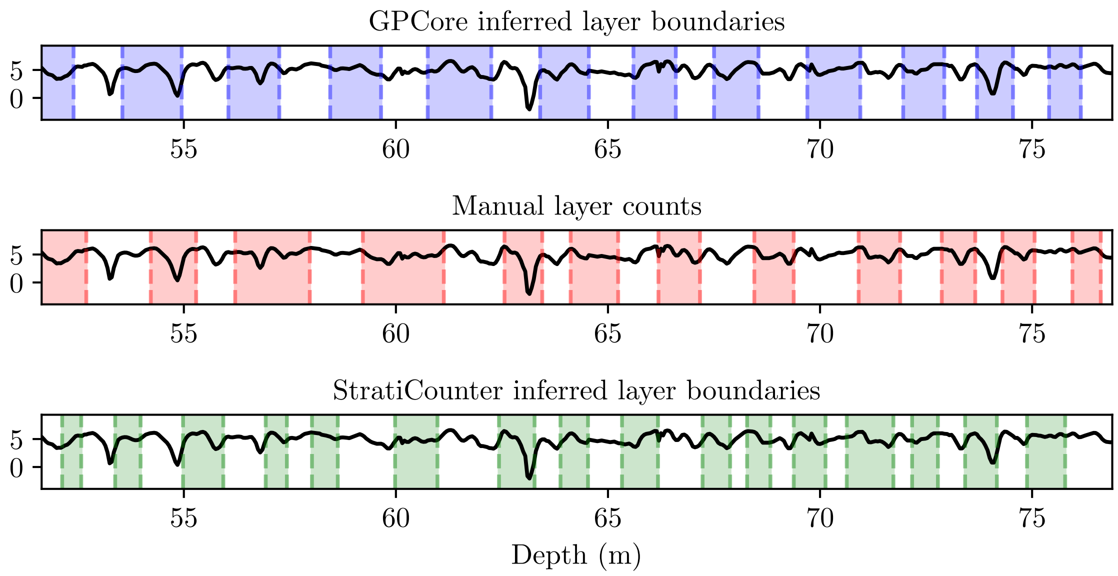


Fig. 3.14 Comparison of the best performing run of GPCore from Fig. 3.12 and StratiCounter inferred layer boundaries with the manual layer counts of BAS Ice Core 3 when the δ^{18}O proxy data was used.

Chapter 4

Conclusions

In this report we have presented a flexible, non-parametric Bayesian approach for fitting high-resolution chronologies in ice cores with annually-periodic proxy data. The model is based on Gaussian processes, avoiding the need for subjective selection of model parameters to elicit expert knowledge. The method searches for the maximum-a-posteriori chronology vector $\boldsymbol{t}$ of ages associated with a set of query depths $\boldsymbol{d}$ (typically at each proxy measurement location) given the ice core data and assumptions of the model. Naïve gradient-based optimisation over the chronology posterior is infeasible because of the computational cost, the need for a good initialisation, and the large number of local minima exhibited by the objective function. We have developed a bespoke optimisation procedure to overcome these issues. The resulting algorithm, which we call GPCore, is a work-in-progress. In its current form, it requires a set of initial annual layer boundary estimates, which are used for initialisation of the algorithm. Because of the loose way in which the input layer counts are used, GPCore has demonstrated error-correcting capability when layer count errors were introduced in simulations. GPCore offers a refined, high-resolution transformation of the chronology associated with the input layer counts, as well as the potential for rigorous uncertainty estimates. In this sense, GPCore can be thought of as a post-processing algorithm once a set of layer counts have been obtained. Although an ice core's chronology is unlikely to be required at such a high resolution, we hypothesise that operating over the finely-sampled timescale offers improved robustness in the inferred chronology. The work in this report indicates that this is the case and future work should verify these findings with more tests.

We conclude by outlining some of the main differences between GPCore and a state-of-the-art algorithm for ice core chronology inference, StratiCounter. According to the performance measures we adopt in this project, GPCore has generally exceeded the performance of StratiCounter in early-stage trials with both simulated and real ice core datasets. Finally, avenues are outlined that could be explored by researchers developing GPCore further in the future.

4.1 Differences Between the StratiCounter and GPCore Algorithms

A shortcoming in the current GPCore statistical model is that it does not use a fixed periodic component in the proxy model. For this reason, the form of GPCore at the end of this project will struggle to re-align its inferred depth-time mapping with the seasonal cycle after making a mistake. Thus GPCore does not work directly with phase information from the proxy signal and its solution can drift and begin to predict the 'summer peak' of a proxy series during winter. However, observe in Fig. 4.1 that StratiCounter is able to recover the correct phase after a series of errors. This is because of StratiCounter's approach of performing inference over layer boundaries, as well as its use of manual layer counts to extract basis vectors for its annual cycle model (as discussed in Section 2.1). An alternative GPCore model for proxy signals with a fixed periodic component should mitigate this phase drift issue and is discussed in Section 2.4.

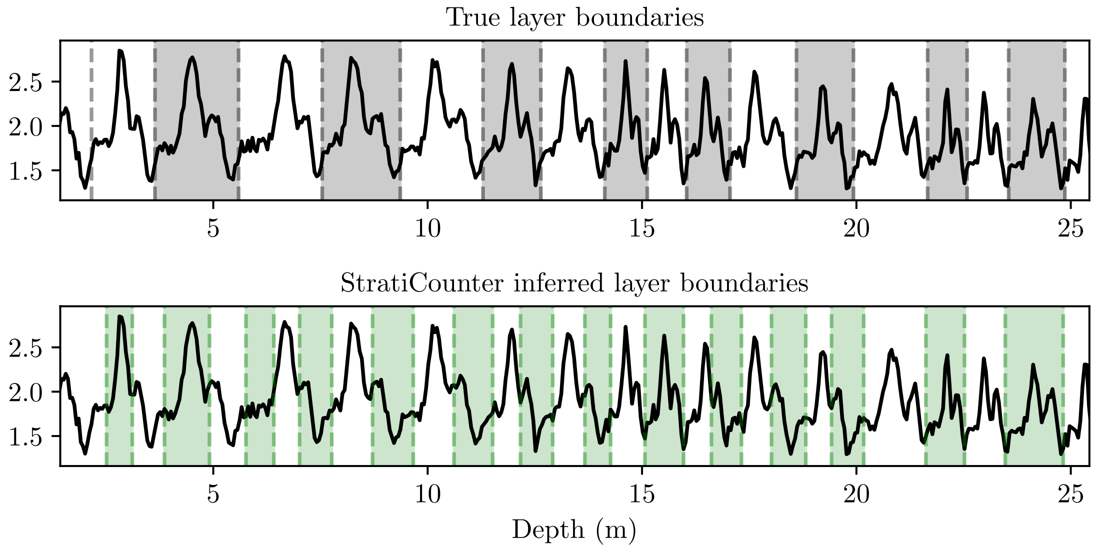


Fig. 4.1 StratiCounter is able to recapture the phase of annual proxy cycles after a section where mistakes are made (here demonstrated with simulated proxy data).

Compared with our approach in GPCore of inferring the ice core age associated with the proxy samples, StratiCounter's layer boundary approach reduces the computational complexity by minimising the dimension of the chronology vector, while still operating over informative tie points in the chronology. Fig. 1.3 highlights the new paradigm with which we frame the problem, and is what gives GPCore the ability to infer *sub-annual* warping phenomena, while StratiCounter is blinded to such properties of the mapping by working only with layer boundaries.

GPCore and StratiCounter use the initial set of manual layer counts in different ways. StratiCounter uses them to extract PCA components for the basis matrix X used in its probabilistic model, as discussed in Section 2.1. This could explain why GPCore generally outperforms StratiCounter in each of the simulated trials that involve corrupted manual layer counts. In contrast, GPCore only uses the input layer count dataset for the initialisations of each chunk, which makes it far more robust to errors in the layer count dataset.

We believe that the sum of theoretical concepts underpinning GPCore are more accessible than that of StratiCounter and require only a basic understanding on Bayesian statistics. Indeed, Section 2.2 covers almost all of the theory behind GPCore in a few pages. In contrast, StratiCounter employs the forward-backward and expectation maximisation algorithms for the HSMM, which involve a greater volume of theory and require a strong existing knowledge of statistics to be understood. GPCore also involves less code than StratiCounter. This means that GPCore is less of a black box and lends itself better to be adopted in the research of climate scientists.

StratiCounter allows for the use of ‘tie points’ where the age of a certain depth of the ice core is known exactly, typically due to layers of ash associated with a particular volcanic eruption, by constraining the number of layers between tie points. The current form of GPCore cannot readily handle such constraints because of the way the dataset is processed in chunks. One option to utilise tie points in GPCore would be to fit the entire chronology between tie points at once, applying an equality constraint to the sum of time differences. However, this would make the chronology search harder when it is already a very challenging optimisation problem.

As we have discussed, GPCore is significantly more computationally expensive than StratiCounter. However, we note that efficiency and speed have not been a priority in this project and that there is potential to achieve significant speed improvements. We also comment that ice cores are certainly not extracted in an online fashion and a chronology inference algorithm that takes several days to process a kilometre of ice core data would not cause significant problems to the research of that core (as long as the performance improvement over faster methods is justified).

4.2 Future Avenues

Future work with GPCore will investigate embedding fixed periodic structure in the proxy model, such as the form given in 2.18. It is expected that this will enable GPCore to capture the phase of annual proxy cycles and improve performance. The simulated trial procedure utilised in this study should be performed again with the modified proxy model. The toy proxy series generated from such a model will appear as warped, noisy sinusoids and will more closely resemble real proxy data, rather than the random-period, decorrelating GP samples used in this project.

GPCore's less strict use of the initial set of layer boundaries motivates the hypothesis that GPCore could run in an 'unsupervised mode' with no initial manually-counted layer boundaries, which would be a clear improvement over StratiCounter. This approach could initialise each chunk with a few different initial estimates – either linearly or by sampling from some GP – and running each of these chunk optimisations to convergence and picking the solution with the lowest NLML. However, this would increase computational cost and room for error. If annual layer counts are simple to obtain through a manual or automated procedure, they are likely to provide the best means to initialise and guide GPCore's optimisation procedure. GPCore could then be thought of as refining the estimated chronology from the input layer counts, correcting errors and providing principled uncertainty estimates.

GPCore can be significantly parallelised – each forward and backward pass is independent of the other and these can run simultaneously. More importantly, the chunking procedure within each forward and backward pass could be modified slightly to provide independence between chunk chronology solutions by initialising each $t_0^{(c)} = 0$, say, and processing each chunk of the ice core in parallel (caring only about the inferred time differences rather than the raw time values). The chunk solutions can then be combined through the linear interlacing of the inferred time differences, as discussed in Chapter 2. This would decrease GPCore's runtime by a factor proportional to the number of chunks that the ice core dataset is split into. The factor of speed increase could be over 100 and would allow many more simulations to be run, providing better estimates for GPCore's average performance using the 'error' metric adopted in this project.

Proxy data pre-processing could yield performance improvements by band-pass filtering the signal, for example, making the annual cycles easier to detect. The standardisation method outlined in Wheatley et al. (2012) could be used for this procedure.

While the ice core dataset is processed in chunks due to the cost of gradient evaluations and the need for good initialisations, the overall NLML is simple to evaluate once an estimate exists for the whole chronology vector. The NLML could be tracked as the optimisation progresses and used to guide the search, perhaps taking inspiration from metaheuristic optimisation methods (such as Simulated Annealing) that use stochasticity. The solution with the lowest NLML could be chosen as the output chronology, rather than only using the final solution to give $\boldsymbol{t}_{GPC}$ (as is currently the case). This could mitigate scenarios where further GPCore passes worsen the error in the estimated chronology.

Wheatley (2015) is the only work to attempt high-resolution chronology inference at every proxy measurement location using a parametric model and an MCMC sampling scheme. GPCore offers benefits over this work in that it is non-parametric and works directly with the posterior rather than sampling from it, thus avoiding sampling heuristics. Future research should further investigate Wheatley (2015) verify that GPCore addresses its shortcomings (such as its inability to deal with noisy proxy data). Inspiration could be taken from the way that the multimodality of the posterior is well characterised in Wheatley (2015) through the sampling procedure used.

The search over possible algorithm set-ups for GPCore is far from exhausted. There is thus a need for further grid searches over the GPCore parameters, θ_{GPC}, and seeing how the average performance depends on these parameters. There are also several alternative design choices that should be explored, such as using multiple chunk annealing phases with a growing chunk size.

Further simulations should test performance when the GP hyperparameters are fit to the data, rather than the unrealistic scenario of fixing them equal to the true values that generated the toy data. This will make the simulations more closely resemble the real case where the best choice for hyperparameters is not known. A discussion of how to find an appropriate choice of GP hyperparameters from an ice core dataset is given in Section 3.5. In addition, the effect of lower-resolution and noisier proxy data on GPCore's average performance should also be investigated.

We conclude by suggesting an alternative use case of the work presented in this report. GPCore's graphical model given in Fig. 2.1 is modular and can be extended to include more random variables and their dependencies. We demonstrate this with a toy example in Fig. 4.2, where the ice core's proxies could be used to predict an unknown climate variable c (such as temperature) through multi-input, single-output regression (possibly modelled with a GP), taking into account uncertainty in the chronology. This simple model should only be considered as an illustration but could provide inspiration for a more detailed investigation in the future.

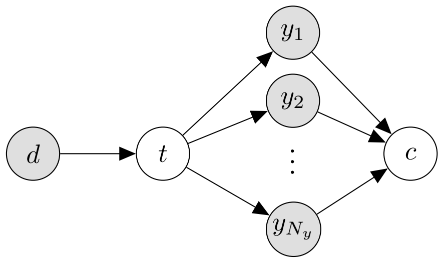


Fig. 4.2 Illustration of a simple fully-Bayesian ice core paleoclimate model for predicting an unknown climate variable c, taking into account uncertainty in the ice core's chronology.

References

Bishop, C. (2006). *Pattern recognition and machine learning.* Springer, New York.

Bruinsma, W. and Tebbutt, W. (2018). Stheno. https://github.com/wesselb/stheno. [Online; accessed 29-05-2019].

Duvenaud, D. (2014). *Automatic model construction with Gaussian processes.* PhD thesis, University of Cambridge.

Jouzel, J., Masson-Delmotte, V., Cattani, O., Dreyfus, G., Falourd, S., Hoffmann, G., Minster, B., Nouet, J., Barnola, J.-M., Chappellaz, J., et al. (2007). Orbital and millennial antarctic climate variability over the past 800,000 years. *science*, 317(5839):793–796.

Karpatne, A., Ebert-Uphoff, I., Ravela, S., Babaie, H. A., and Kumar, V. (2018). Machine learning for the geosciences: Challenges and opportunities. *IEEE Transactions on Knowledge and Data Engineering.*

Lemieux-Dudon, B., Blayo, E., Petit, J.-R., Waelbroeck, C., Svensson, A., Ritz, C., Barnola, J.-M., Narcisi, B. M., and Parrenin, F. (2010). Consistent dating for antarctic and greenland ice cores. *Quaternary Science Reviews*, 29(1-2):8–20.

Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

MacKay, D. (2003). *Information theory, inference, and learning algorithms.* Cambridge University Press, Cambridge, UK New York.

Maclaurin, D., Duvenaud, D., and Adams, R. P. (2015). Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML Workshop.*

Parrenin, F., Bazin, L., Capron, E., Landais, A., Lemieux-Dudon, B., and Masson-Delmotte, V. (2015). Icechrono1: a probabilistic model to compute a common and optimal chronology for several ice cores. *Geoscientific Model Development*, 8(5):1473–1492.

Rasmussen, C. and Williams, C. (2006). *Gaussian processes for machine learning.* MIT Press, Cambridge, Mass.

Svensson, A., Andersen, K. K., Bigler, M., Clausen, H. B., Dahl-Jensen, D., Davies, S., Johnsen, S. J., Muscheler, R., Parrenin, F., Rasmussen, S. O., et al. (2008). A 60 000 year greenland stratigraphic ice core chronology. *Climate of the Past*, 4(1):47–57.

Sweeney, J., Salter-Townshend, M., Edwards, T., Buck, C. E., and Parnell, A. C. (2018). Statistical challenges in estimating past climate changes. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(5):e1437.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574.

Turner, J., Phillips, T., Thamban, M., Rahaman, W., Marshall, G. J., Wille, J. D., Favier, V., Winton, H., Thomas, E., Wang, Z., et al. (2019). The dominant role of extreme precipitation events in antarctic snowfall variability. *Geophysical Research Letters*.

Wheatley, J. (2015). *Automated Bayesian Layer Counting of Ice Cores*. PhD thesis, University of Sheffield.

Wheatley, J., Blackwell, P., Abram, N., McConnell, J., Thomas, E., and Wolff, E. (2012). Automated ice-core layer-counting with strong univariate signals. *Climate of the Past*, 8:1869–1879.

Wheatley, J., Blackwell, P., Abram, N., and Wolff, E. (2014). Bayesian layer counting in ice-cores: Reconstructing the time scale. In *The Contribution of Young Researchers to Bayesian Statistics*, pages 121–125. Springer.

Winstrup, M. (2016). A hidden markov model approach to infer timescales for high-resolution climate archives. In *Twenty-Eighth IAAI Conference*.

Winstrup, M., Svensson, A., Rasmussen, S. O., Winther, O., Steig, E., and Axelrod, A. (2012). An automated approach for annual layer counting in ice cores. *Climate of the Past Discussions*, 8(6):1881–1895.

Yu, S.-Z. (2010). Hidden semi-markov models. *Artificial intelligence*, 174(2):215–243.