Advanced Monte Carlo methods and applications

by

Shijia Wang

M.Sc., Memorial University, 2015
M.Sc., Wuhan University, 2013
B.Sc., Yunnan University, 2011

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in the Department of Statistics and Actuarial Science Faculty of Applied Science

© Shijia Wang 2019
SIMON FRASER UNIVERSITY
Summer 2019

Copyright in this work rests with the author. Please ensure that any reproduction or re-use is done in accordance with the relevant national copyright legislation.
Approval

Name: Shijia Wang

Degree: Doctor of Philosophy (Statistics)

Title: Advanced Monte Carlo methods and applications

Examing Committee: Chair: Dr. Jinko Graham

Professor

Dr. Liangliang Wang
Senior Supervisor
Associate Professor

Dr. Lloyd T.Elliott
Supervisor
Assistant Professor

Dr. Tim Swartz
Internal Examiner
Professor

Dr. Longhai Li
External Examiner
Professor
Department of Mathematics and Statistics
University of Saskatchewan

Date Defended: August 16, 2019
Abstract

Monte Carlo methods have emerged as standard tools to do Bayesian statistical inference for sophisticated models. Sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC) are two main classes of methods to sample from high dimensional probability distributions. This thesis develops methodologies within these classes to address problems in different research areas.

Phylogenetic tree reconstruction is a main task in evolutionary biology. Traditional MCMC methods may suffer from the curse of dimensionality and the local-trap problem. Firstly, we introduce a new combinatorial SMC method, with a novel and efficient proposal distribution. We also explore combining SMC and Gibbs sampling to jointly estimate the phylogenetic trees and evolutionary parameter of genetic data sets. Secondly, we propose an “embarrassingly parallel” method for Bayesian phylogenetic inference, annealed SMC, based on recent advances in the SMC literature such as adaptive determination of annealing parameters.

Another application of the methods presented in this thesis is in genome wide-association studies. Linear mixed models (LMMs) are powerful methods for controlling confounding caused by population structure. We develop a Bayesian hierarchical model to jointly estimate LMM parameters and the genetic similarity matrix using genetic sequences and phenotypes. We develop an SMC method to jointly approximate the posterior distributions of the LMM and phylogenetic trees.

We also consider parameter estimation for nonlinear differential equation (DE) systems from noisy measurements of dynamic systems. We develop a fully Bayesian framework for non-linear DE systems. A flexible nonparametric function is used to represent the dynamic process such that expensive numerical solvers can be avoided. We derive an SMC method to sample from multimodal DE posterior distributions. In addition, we consider Bayesian computing problems related to importance sampling and misclassification in multinomial data. Lastly, motivated by a personalized recommender system with dynamic preference changes, we develop a new hidden Markov model (HMM) and propose an efficient online SMC algorithm by hybridizing with the EM algorithm for the HMM model.

Keywords: Bayesian statistics; phylogenetics; sequential Monte Carlo; genome-wide association studies; linear mixed model; differential equations; hidden Markov Model; importance sampling; latent variables; misclassification; Markov chain Monte Carlo
Acknowledgements

First of all, I want to express my gratitude to my senior supervisor Dr. Liangliang Wang. I really hope that I could write one chapter to show my appreciation. Thanks for introducing me to the area of Bayesian Monte Carlo methods and phylogenetics, and for making me gradually become an independent researcher. Also thanks for providing research discussions with me on weekends and holidays, and sending me to conferences.

I gratefully acknowledge Dr. Lloyd Elliott for supervising me in my last year, on genetics and machine learning. I appreciate his efforts on organizing a journal club, recommending good articles. I also gratefully acknowledge Dr. Tim Swartz for supervising me on projects involving Bayesian computing. I would also like to thank all of my collaborators Drs. Noel Cadigan, Alexandre Bouchard-Côté, Yee Whye Teh, Jiguo Cao, Caroline Coljin, Jinhui Liu, Nan Zheng and Hugues Benoît for research collaborations.

During my time at SFU, I had the pleasure of taking courses and having discussion with: Drs. Tom Loughin, Jinko Graham, Brad McNeney, Steve Thompson, Dave Campbell, Cedric Chauve. I would also like to thank my friends Zhiyang, Yuping, Tianyu, Dongmeng, Chenlu, Payman et al. for having fun together. Thanks to SFU for providing financial support and a friendly working environment.

I appreciate my parents for their support and encouragement. Most of all, I would like to show my appreciation to my wife Shufei, for her love, support and humour.
## Contents

Approval ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables vi

List of Figures vii

1 Introduction 1

2 Monte Carlo Methods 4
   2.1 Importance sampling (IS) .............................................. 4
   2.2 Markov chain Monte Carlo (MCMC) .................................. 5
      2.2.1 Metropolis-Hastings (MH) ..................................... 6
      2.2.2 Gibbs Sampling (GS) ........................................... 7
      2.2.3 Slice Sampler (SS) ............................................. 8
   2.3 Sequential Monte Carlo (SMC) ..................................... 8
      2.3.1 Standard SMC .................................................. 9
      2.3.2 Resampling Methods ......................................... 11
      2.3.3 General sequential Monte Carlo ............................. 12
   2.4 Particle Markov chain Monte Carlo ............................ 14
      2.4.1 Particle Gibbs Sampler (PGS) ................................. 14

3 Background on Phylogenetics 16
   3.1 Phylogenetic tree .................................................. 16
   3.2 Data for phylogenetics ............................................. 17
      3.2.1 Data Types .................................................... 17
   3.3 Model-based methods for phylogeny reconstruction ............. 17
      3.3.1 Stochastic Evolutionary Models ............................. 18
3.3.2 Maximum Likelihood ........................................... 19
3.3.3 Framework of Bayesian Phylogenetics ...................... 20
3.3.4 Bayesian Phylogenetics via MCMC .......................... 20
3.4 Consensus tree and tree distances ............................ 21

4 An Annealed Sequential Monte Carlo Method for Bayesian Phylogenetics 23
4.1 Introduction ...................................................... 23
4.2 Literature review ................................................ 24
4.3 Setup and notation ............................................. 25
4.4 Annealed SMC for phylogenetics .............................. 26
  4.4.1 Sequences of Distributions ................................ 26
  4.4.2 Basic Annealed SMC Algorithm .......................... 28
4.5 Adaptive mechanisms for annealed SMC ...................... 32
  4.5.1 Measuring Particle Degeneracy using Relative (Conditional) Effective Sample Size (ESS) .................. 34
  4.5.2 Dynamic Resampling ...................................... 34
  4.5.3 Adaptive Determination of Annealing Parameters .. 35
  4.5.4 Computational Complexity ............................... 37
4.6 Review of other marginal likelihood estimation methods .... 37
  4.6.1 Stepping Stone ............................................ 37
  4.6.2 Linked Importance Sampling .............................. 38
4.7 Theoretical properties .......................................... 39
  4.7.1 Properties of Annealed Sequential Monte Carlo ..... 39
  4.7.2 Using Unbiasedness to Test Implementation Correctness . 41
  4.7.3 Properties of the Stepping Stone Method ............... 41
  4.7.4 Comparison of Unbiased Marginal Likelihood Estimators . 42
4.8 Simulation studies .............................................. 42
  4.8.1 Simulation Setup and Tree Distance .................... 42
  4.8.2 Hidden Markov Models ................................... 42
  4.8.3 Comparison of Marginal Likelihood Estimates ........ 44
  4.8.4 Comparison of Model Selection by Annealed SMC versus Stepping Stone .......................... 45
  4.8.5 Comparison of Tree Distance Metrics .......................... 46
  4.8.6 Influence of Number of Threads, $\beta$, and $K$ ........... 47
  4.8.7 Trade-off between $R$ and $K$ ............................ 48
  4.8.8 Analysis of Subsampling SMC ........................... 48
  4.8.9 Comparison of ASMC and Combinatorial SMC (CSMC) ........... 50
4.9 Real datasets .................................................... 52
  4.9.1 Dataset M336 .............................................. 52
  4.9.2 Dataset M1809 ............................................ 54
4.10 Conclusion and discussion .................................................. 54
4.11 APPENDIX 1 ................................................................. 56
  4.11.1 Construction of Intermediate Distributions for Subsampling SMC 56
4.12 APPENDIX 2 ................................................................. 57
  4.12.1 Theoretical Foundations of Annealed SMC ......................... 57
  4.12.2 General Estimates of Marginal Likelihood ......................... 58
  4.12.3 Unbiasedness, Consistency and Central Limit Theorem for Annealed SMC 59
4.13 APPENDIX 3 ................................................................. 61
  4.13.1 MCMC Proposals for Bayesian Phylogenetics ..................... 61
4.14 APPENDIX 4 ................................................................. 62
  4.14.1 Comparison of Resampling Strategies .............................. 62
4.15 APPENDIX 5 ................................................................. 62
  4.15.1 Review of Particle Degeneracy Measures .......................... 62
  4.15.2 Comparison of Relative ESS (rESS) and Relative CESS (rCESS) .......... 64
4.16 APPENDIX 6 ................................................................. 66
  4.16.1 Estimates of Marginal Likelihood from LIS ....................... 66
4.17 APPENDIX 7 ................................................................. 67
  4.17.1 Derivation of Upper Bound of CV ................................. 67
4.18 APPENDIX 8 ................................................................. 67
  4.18.1 Tuning of $\beta$ and $K$ .......................................... 67
4.19 APPENDIX 9 ................................................................. 67
  4.19.1 Comparison of ASMC, DASMC, LIS and SS for large $K$ ............ 67

5 Particle Gibbs Sampling for Bayesian Phylogenetic inference 70
  5.1 Introduction ...................................................................... 70
  5.2 Background and notation .................................................. 72
    5.2.1 Background .......................................................... 72
    5.2.2 Notation ............................................................. 72
  5.2.3 Bayesian phylogenetic inference ..................................... 73
  5.2.4 Markov chain Monte Carlo .......................................... 73
    5.2.5 Limitations of MCMC .............................................. 74
  5.3 Phylogenetic tree inference .............................................. 75
    5.3.1 Combinatorial sequential Monte Carlo ............................ 75
    5.3.2 Revert-merge combinatorial sequential Monte Carlo ................ 76
    5.3.3 A phylogenetic tree example using revert-merge CSMC. ............ 81
  5.4 Joint estimation of phylogenetic tree and evolutionary parameter .... 82
    5.4.1 Particle Gibbs Sampler ........................................... 82
    5.4.2 Particle Gibbs Sampler with Ancestor Sampling .................. 83
  5.5 Interacting particle Markov chain Monte Carlo ...................... 84
List of Tables

Table 4.1  Number of annealing parameters ($R_{SMC}$) for ASMC with $K = 1000$ and $\beta = 5$. 44
Table 4.2  Comparison of model selection by ASMC and SS based on the Bayes factor. 46
Table 4.3  Comparison of tree distance metrics using ASMC and MCMC. 47
Table 4.4  Comparison of adaptive SMC algorithm with different numbers of particles and $\beta$. 48
Table 4.5  Comparison of running ASMC and MrBayes for M336 from TreeBASE. 52
Table 4.6  Comparison of running ASMC and MrBayes for M1809 from TreeBASE. 54

Table 5.1  Runtime (second) of CSMC and CSMC-RM, with mean and 95% confidence intervals, for 1 to 4 threads. 86
Table 5.2  Log-likelihood for consensus tree provided by PGs and IPMCMC-RM, with mean and standard deviation, with varying number of particles. 90

Table 6.1  AUC and TPR at FDR 0.05 for simulation Scenarios A and B. TreeLMM shows improved AUC and TPR in both Scenarios. 107
Table 6.2  TreeLMM runtime (second), with 50%, 2.5% and 97.5% quantiles, for 1 to 4 threads. 112

Table 7.1  Posterior mean and corresponding 95% credible interval (CI) for real data. 125
Table 7.2  Estimated parameters and MSE of $W(t)$ for three simulated data sets. 133
Table 7.3  Parameter estimates and corresponding 95% credible interval (CI) for nonlinear DDE models. 134

Table 8.1  Parameter summary for the misclassified multinomial model with covariates. 144

Table 9.1  Some summary results corresponding to Example 1. 158
Table 9.2  Estimates of the posterior mean of $\theta_1$ and standard errors corresponding to Example 2. 159
## List of Figures

| Figure 4.1 | An overview of the annealed SMC algorithmic framework for phylogenetic trees. The algorithm iterates the following three steps: (i) compute the weights using samples from the previous iteration, (ii) perform MCMC moves to propose new samples, and (iii) resample from the weighted samples to obtain an unweighted set of samples. | 31 |
| Figure 4.2 | Graphical representation of (a) a hidden Markov model; (b) transitions between hidden states. | 43 |
| Figure 4.3 | Estimates of the log marginal likelihood based on 100 independent random seeds for each configuration. The axis label “nTemps” refers to the number of equally spaced intermediate distributions (“inverse temperatures”). The black line shows the true value computed using the forward-backward algorithm. | 44 |
| Figure 4.4 | Marginal likelihood (in log scale) estimates for different numbers of taxa with a fixed computational budget. | 45 |
| Figure 4.5 | Coefficient of variation (CV) for the marginal likelihood estimates versus the number of taxa for ASMC and DSMC with a fixed number of particles. | 46 |
| Figure 4.6 | Computing time of ASMC using multiple threads. | 48 |
| Figure 4.7 | Performance of deterministic SMC algorithm on a fixed computational budget ($K \cdot R = 10^6$). We select 5 values of $K$, 100, 300, 1000, 3000, 10000, from left to right on X-axis. | 49 |
| Figure 4.8 | Ratio of cost (subsampling/annealing) versus the batch size. | 49 |
| Figure 4.9 | Comparison of subsampling SMC algorithms with different size of batch sites. | 50 |
| Figure 4.10 | Comparison of adaptive SMC algorithms with CSMC for three types of simulated trees: clock, relaxed clock, nonclock (from top to bottom). | 51 |
| Figure 4.11 | The majority-rule consensus trees for the M336 dataset estimated by (a) ASMC and (b) MrBayes. The numbers on the trees represent the clade posterior probabilities (number 100 is omitted). Most clades posterior probabilities provided by ASMC and MrBayes are close. ASMC provides lower posterior support for some clades, which is consistent with the hypothesized superior tree exploration provided by ASMC on a fixed budget. | 53 |
| Figure 4.12 | Comparison of three resampling thresholds, 0, 0.5 and 1. | 62 |
Figure 4.13 Comparison of rCESS (left) and rESS (right) in terms of rESS as a function of r (top) and \( \phi_r - \phi_{r-1} \) as a function of \( \phi_r \) (bottom). 65

Figure 4.14 Comparison of adaptive annealed SMC using rCESS and rESS in terms of estimating the log marginal likelihood, the log likelihood of the consensus tree, tree distance metrics, and the number of SMC iterations (R). 66

Figure 4.15 Comparison of adaptive SMC algorithm with different numbers of particles, from left to right \( K = 100, 300, 1000, 3000 \). 68

Figure 4.16 Comparison of adaptive SMC algorithm with different \( \beta \), from left to right \( \beta = 3, 4, 4.3, 5, 5.3 \) (x-axis). Here R is the total number of SMC iterations. 68

Figure 4.17 Comparison of marginal likelihood (in log scale) provided by ASMC, DASMC, LIS and SS with a fixed computational budget when \( K = 200000 \). 69

Figure 5.1 Phylogenetic tree construction examples: (a) ultrametric tree, (b) non-binary tree, (c) binary non-clock tree, (d) DNA sequences. 72

Figure 5.2 (a) overcounting issue in CSMC; (b) overcounting issue in revert-merge CSMC; (c) overcounting issue in the revert step of revert-merge CSMC. 75

Figure 5.3 An example to illustrate the limitation of CSMC. 76

Figure 5.4 An overview of the revert-merge CSMC framework. A set of partial states is kept at each SMC iteration. A positive-valued weight is associated with each partial state. The algorithm iterates the following three steps: (i) resample to prune particles with small weights, (ii) propose a new partial state through the revert and merge proposal, and (iii) compute the weights for new particles. 77

Figure 5.5 Revert-merge proposal for CSMC. 78

Figure 5.6 An example of ancestor sampling step with two CSMC. 84

Figure 5.7 Comparison of CSMC and revert-merge CSMC as a function of number of (1000) particles. The x-axis represents number of a thousand particles. The y-axis represents the normalized RF metric. The four levels from left to right for revert-merge CSMC are \( 1 \cdot 10^3, 3 \cdot 10^3, 1 \cdot 10^4 \) and \( 3 \cdot 10^4 \) respectively. The four levels from left to right for naive version of CSMC are \( 1.5 \cdot 10^3, 4.5 \cdot 10^4, 1.5 \cdot 10^4, 4.5 \cdot 10^4 \) respectively. The y-axis represents the Robinson Foulds metric. 87

Figure 5.8 Comparison of PGS-RM with fixed computational budgets. 88

Figure 5.9 Comparison of PGs and IPMCMC with the naive version of CSMC (IPGS, PGS) and revert-merge CSMC (IPGS-RM, PGS-RM), PGAS with revert-merge CSMC (PGAS) as a function of number of particles. The x-axis represents number of particles. The three levels from left to right are 200, 500, 1000, respectively. The y-axis of represents the Consensus log likelihood and Robinson Foulds metric. The number of MCMC iterations are fixed at 4000. 89
Figure 5.10 Traceplot of nucleotide substitution parameter $\theta$ provided by one run IPMCMC-RM (Primate data). ................................................................. 90
Figure 5.11 Consensus tree of the primate data set provided by PGs-RM, IPMCMC-RM and PGAS. ................................................................. 91
Figure 5.12 Traceplot of nucleotide substitution parameter $\theta$ provided by PGAS (Cichlid Fishes dataset). ................................................................. 91
Figure 5.13 Consensus tree of the Cichlid Fishes dataset provided by PGs (left) and PGAS (right). ................................................................. 92

Figure 6.1 Graphical structure of TreeLMM. The phenotype $y$ is a collider (Pearl, 2000). Conditioning on $y$ and genotype $G$ induces dependence among the effect size $\beta$, the phylogeny $t$, and the environmental and genetic variance ($\sigma^2_g$ and $\sigma^2_e$). Plate diagram for a) TreeLMM, and b) TreeLMM conditioned on genotype and phenotype. ................................................................. 100
Figure 6.2 Segregation by an unobserved SNP on edge $d_e$. Computation of the expected mean $\mu_{e_d}$, and variance $\sigma_{e_d}$, for an unobserved allele involves integration over edges $e_d$, and examination of the connected components of the tree formed by removing edge $e_d$. ................................................................. 103
Figure 6.3 A Manhattan plot of SNP association with MDR-TB phenotype: $p$-values are shown for linear association tests (grey circles), the linear mixed model with empirical genetic similarity matrix (blue triangles), and tree-based linear mixed model (red squares). ................................................................. 106
Figure 6.4 ROC provided by linear regression (LM), linear mixed model with empirical genetic similarly matrix (LMM) and TreeLMM (TR) for datasets A (left panel) and B (right panel), the vertical dotted line is at FPR level of 0.05. . 107
Figure 6.5 TreeLMM performance in simulation Scenario A, as the number of particles $K$ (panels one and two, with $R = 200$) and intermediate target distributions $R$ (panels three and four, with $K = 100$) in the annealing are varied. . . . . 108

Figure 7.1 Optional caption for list of figures ................................................................. 116
Figure 7.2 The thirteen B-spline basis functions defined on [0, 1] with degree three and nine equally spaced knots. ................................................................. 117
Figure 7.3 Blowfly population in one experiment published in Nicholson (1954); the time unit is one day. ................................................................. 125
Figure 7.4 Estimated mean trajectory and 95% credible intervals for delay differential equation modelling Blowflies counts. ................................................................. 126
Figure 7.5 Posterior samples of $r$, $\tau$, $P$ for DDE in Equation (13) estimated via SMC. We resample the particles at last SMC iteration such that they are equally weighted. ................................................................. 126
Figure 7.6 Simulated ODE trajectories and observations. Red lines in Figure refer to simulated ODE trajectories and blue points refer to simulated observations.

Figure 7.7 Intermediate posterior for \( \theta \) and \( \sigma \) (two panels in the first row) and estimated ODE trajectories and the 95% confidence interval (two panels in the second row).

Figure 7.8 MCMC traceplot for \( \theta \). The yellow dashed line denotes the posterior mean, and the red dashed line denotes the true value of \( \theta \).

Figure 7.9 ODE parameter estimates with different rCESS in SMC.

Figure 7.10 ODE parameter estimates with different \( K \) in SMC.

Figure 7.11 ODE parameter estimates provided by different number of knots.

Figure 7.12 ODE parameter estimates with different choices of \( \lambda \).

Figure 7.13 One example of posterior samples of \( \lambda \).

Figure 7.14 Simulated DDE trajectories and observations. Red lines in Figure refer to simulated DDE trajectory and blue points refer to simulated observations.

Figure 7.15 Estimated DDE trajectories and the 95% confidence interval using \( L = 28 \) basis functions (upper two panels) and using \( L = 43 \) basis functions (lower two panels).

Figure 9.1 Contour plots of (a) \( RS N_2(0, 0, (0, 0)') \) (i.e. the bivariate standard normal), (b) \( RS N_2(0, 0, (1, 0)') \), (c) \( RS N_2(0, 0, (-2, 0)') \) and (d) \( RS N_2(0, 0, (1, 1)') \).

Figure 10.1 An HMM-based dynamical model for recommender system.

Figure 10.2 Trace plots of parameter estimates over time.

Figure 10.3 Estimated transitional probability.

Figure 10.4 Boxplots for parameter estimates.

Figure 10.5 Precision (P), Recall (R) and their harmonic mean (F) for the online SMC-EM and the batch EM of the simulation study.

Figure 10.6 Estimated \( P \) of recommender system.

Figure 10.7 Comparison of the NB dispersion parameter \( a \) for online SMC-EM and Batch EM.

Figure 10.8 Comparison of the NB mean parameter \( \sigma \) for online SMC-EM and Batch EM.

Figure 10.9 Comparison of the NB mean parameter \( \mu \) for online SMC-EM and Batch EM.

Figure 10.10 Comparison of the transition probability \( P \) for online SMC-EM and Batch EM.

Figure 10.11 Comparison of the transition probability \( P \) for online SMC-EM and Batch EM.

Figure 10.12 Comparison of the transition probability \( P \) for online SMC-EM and Batch EM.

Figure 10.13 Comparison of the transition probability \( P \) for online SMC-EM and Batch EM.
Chapter 1

Introduction

Monte Carlo methods are commonly used for performing high-dimensional numerical integration. With recent advances in computing techniques, Monte Carlo methods have become extremely popular in inference for complex statistical models. The most popular Monte Carlo methods used for statistical inference include importance sampling (IS), Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC).

In importance sampling, we generate random samples from a selected proposal distribution, the weights of each sample are evaluated by the ratio of a target distribution and the proposal distribution. A sufficient condition for an importance sampler to perform satisfactorably is that the proposal distribution is “similar” to the target distribution. In general, this is difficult to achieve in high dimensional cases. Markov chain Monte Carlo methods are an alternative class of methods to perform high dimensional statistical inference. The basic idea of Markov chain Monte Carlo is to construct an ergodic Markov chain which admits an invariant target posterior distribution. The Gibbs sampling, Metropolis Hastings algorithms, and slice sampling are popular MCMC methods. Sequential Monte Carlo is another class of methods for high dimensional posterior inference, which is built on the framework of importance sampling. In sequential Monte Carlo methods, we construct a list of “artificial” intermediate target distributions to bridge a gap between a complex target distribution and a relatively “simple” proposal distribution. In Chapter 2, we provide a brief review for classical Monte Carlo methods, such as importance sampling, Markov chain Monte Carlo, and sequential Monte Carlo.

In phylogenetics, we aim to uncover the evolutionary relationships among different species or populations, using observed biological sequencing data (e.g. DNA, RNA). This evolutionary relationship is represented by a phylogenetic tree. We will provide background on phylogenetics in Chapter 3. Bayesian approaches for phylogenetic reconstruction are extremely popular as they are able to automatically provide uncertainty of estimated trees. Bayesian phylogenetic reconstruction involves an intractable sum over phylogenetic tree topologies as well as a high-dimensional integral over branch lengths and evolutionary parameters. The computation of posteriors for phylogenetic trees is typically carried out by using Markov chain Monte Carlo (MCMC) algorithms. However, there are several challenges in traditional MCMC algorithms for phylogenetic inference. In Chapter
4, we present an annealed sequential Monte Carlo method for Bayesian phylogenetic inference. The algorithm provides an approximate posterior distribution over trees and evolutionary parameters as well as an unbiased estimator for the marginal likelihood. The annealed sequential Monte Carlo method can be relatively easy to incorporate into existing phylogenetic software packages based on Markov chain Monte Carlo algorithms. Chapter 5 introduces a revert-merge combinatorial sequential Monte Carlo (CSMC) algorithm for phylogenetic tree reconstruction. The proposed CSMC algorithm is also explored to serve as an efficient tree designer in particle Gibbs samplers (PGs).

Further, we consider using phylogenetic trees to control the population structure in genome-wide association studies (GWAS). Linear mixed models (LMMs) are popular approaches to control the confounding effects in GWAS, with a genetic similarity matrix used as fixed effects. However, for some studies, such as bacterial studies, in which small numbers of samples are present, or for which genotyping is sparse and noisy, uncertainty about the genetic similarity matrix may degrade the quality of LMM results. In Chapter 6, we proposed a tree-based linear mixed model (TreeLMM) for genome-wide association studies, using phylogenetic trees to control for population structure. We apply our TreeLMM to multidrug-resistant tuberculosis in a genome-wide association study of 469 tuberculosis subjects in a population from Lima, Peru (Grandjean et al., 2017).

Another application area of this thesis is parameter estimation in nonlinear differential equations. Traditional inference methods require expensive numerical DE solvers and are easily get stuck in local modes of DE likelihood functions. Chapter 7 introduces adaptive methods for parameter estimation in nonlinear Bayesian differential equations. Our differential equation examples involve both ordinary differential equations and delay differential equations.

In addition, we develop methodologies for classical Bayesian computing problems in Chapter 8 and Chapter 9. There is rich literature about the analysis of categorical data (Agresti and Kateri, 2011). However, categorical data can be misclassified in some cases. For example, a subject whose “true” classification is the first category may be incorrectly classified in the second category. In Chapter 8, we consider multinomial data subject to misclassification in the presence of covariates which affect both the misclassification probabilities and the true classification probabilities. In addition, we introduce a new and less problematic definition referred to as hierarchical nonidentifiability.

Importance sampling has several advantages over Markov chain Monte Carlo. For example, we do not require any convergence test in importance sampling as consistency is guaranteed. The computation of marginal likelihood in importance sampling is also straightforward. However, importance sampling approaches have fallen out of favour compared to many Markov chain Monte Carlo methods because of limited choice of proposal distributions. In Chapter 9, we propose adaptive importance sampling approaches for fitting a member of the restricted skew-Student family.

Our finally application involves online inference for hidden Markov models via sequential Monte Carlo methods. Most of the literature for collaborative filtering focuses on recommendations for users with static preference (Bennett et al., 2007; Koren, 2009). However, users’ preference could change over time due to, for example, the change of users’ work and social environment. Hence, recommender systems based on static preference may no longer be valid if the users’ preference is
dynamical. In Chapter 10, we propose a hidden Markov model (HMM) for the dynamic changes in
users’ preference of recommender systems. In addition, we propose sequential Monte Carlo algorithm
to conduct online inference for the HMM.

In Chapter 11, we conclude this thesis work.
Monte Carlo Methods

Monte Carlo methods are important tools to approximate high dimensional intractable integrals using random samples. We can always use a list of discrete samples to represent a complex probability distribution $\pi$. Monte Carlo approximations aim to find efficient representation of $\pi$ in terms of an approximation by samples. In this chapter, we provide a brief review for major Monte Carlo Methods that will be used throughout this dissertation. We also refer readers to Robert and Casella (2005) for an excellent review on Monte Carlo statistical methods.

In this chapter, we let $\pi$ denote the target distribution of random variable $x$ defined on a measurable space $(E, \mathcal{E})$. Our interest relies on computing the integral of a measurable function $\phi : E \to \mathbb{R}$ with respect to $\pi$

$$E_\pi(\phi) = \int_E \phi(x) \pi(dx). \quad (1)$$

### 2.1 Importance sampling (IS)

In this section, we briefly review the importance sampling algorithm to sample from a given target distribution $\pi$. Importance sampling (IS) is a basic class of Monte Carlo methods. It is also the basis of sequential Monte Carlo methods we will introduce later in this chapter. Our target distribution is assumed to be known up to a normalizing constant

$$\pi(x) = Z^{-1} \gamma(x),$$

where $\gamma : E \to \mathbb{R}^+$ can be evaluated pointwisely, but the normalizing constant $Z = \int \gamma(x) dx$ is unknown.

In importance sampling, we generate random samples from an importance density (proposal density, or instrumental density) $q(x)$. To be a valid importance sampler, the support of proposal distribution $q(x)$ is supposed to cover the support of the target $\pi(x)$,

$$\pi(x) > 0 \Rightarrow q(x) > 0.$$
The unnormalized weight function for a random sample \( x \) can be evaluated by

\[
w(x) = \frac{\gamma(x)}{q(x)}.
\]

In general, we can select an importance density \( q(x) \) from which is easy to draw samples; e.g. a multivariate Gaussian. If we generate \( N \) random samples \( x_1, x_2, \ldots, x_N \) from a proposal distribution \( q(x) \), the integral of a measurable function \( \phi \),

\[
I(\phi) = \int \phi(x)\pi(x)dx,
\]

can be approximated by

\[
I^{IS}(\phi) = \int \phi(x)\hat{\pi}(x)dx = \sum_{i=1}^{N} W_i \phi(x_i).
\]

Here \( W_i \) is the normalized weight function for \( x_i \),

\[
W_i \propto \frac{\gamma(x_i)}{q(x_i)}.
\]

For high dimensional integrals, it is generally challenging to design a good proposal distribution \( q(x) \), to guarantee satisfactory performance of importance sampling. In the next section, we introduce Markov chain Monte Carlo methods, which serve as good alternatives to importance sampling algorithms.

### 2.2 Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo (MCMC) methods (Robert and Casella, 2005) are widely used to draw random samples from \( \pi \) by constructing ergodic Markov chain which admits a target distribution \( \pi \) as a stationary distribution. A Markov chain, \( \{X_n\} \), is a sequence of random variables evolving over time with the following property: given the present state, future transitions of the chain are independent of the past history. More precisely,

\[
P(X_{n+1} \in A | x_0, x_1, \cdots, x_n) = P(X_{n+1} \in A | x_n) = \int_A M(x_n, dx),
\]

where \( M \) is a transition kernel defined by a measurable space \( (E, \mathcal{E}) \) such that:

1. \( \forall x \in E, \ M(x, \cdot) \) is a probability measure;
2. \( \forall A \in \mathcal{E}, \ M(\cdot, A) \) is measurable.
For an irreducible and aperiodic Markov chain \( \{X_n\} \), we let \( \pi \) denote the equilibrium distribution. We have the following convergence theorem

\[
P(X_n = y | X_0 = x) \to \pi(y) \text{ as } n \to \infty \text{ for all } x, y.
\]

Further, given that the chain is irreducible, for any function \( \phi(\cdot) \) such that \( \mathbb{E}_\pi[|\phi(X)|] < \infty \),

\[
\frac{1}{K} \sum_{k=1}^{K} \phi(X_k) \to \mathbb{E}(\phi(X)) \text{ as } K \to \infty,
\]

with almost sure convergence in probability. If the equilibrium distribution of this Markov chain is our target distribution, this Markov chain converges to the required target distribution. We refer readers to Robert and Casella (2005) for more details.

We now introduce the detailed balance condition. Generally we require the transition kernel to satisfy this condition to ensure our constructed Markov chain Monte Carlo algorithms are valid.

**Definition 1.** A Markov chain with transition kernel \( M \) satisfies the detailed balance condition if there exists a function \( f \) satisfying

\[
M(y, x)f(y) = M(x, y)f(x),
\]

for every \( (x, y) \).

Further, we have the following theorem (Robert and Casella, 2005),

**Theorem 1.** Suppose that a Markov chain with transition function \( M \) satisfies the detailed balance condition with \( \pi \) a probability density function. Then:

1. The density \( \pi \) is the invariant density of the chain.
2. The chain is \( \pi \)-reversible.

### 2.2.1 Metropolis-Hastings (MH)

The Metropolis-Hastings (MH) algorithm is a popular Markov chain Monte Carlo method (Metropolis et al., 1953; Hastings, 1970). We let \( \pi \) denote the target distribution. Let \( q(x, \cdot) \) denote the proposal distribution, which is used to propose a candidate state. For a proposed value \( y \), we calculate the Metropolis-Hastings (MH) acceptance probability \( \alpha(x, y) \) based on the current value \( x \) as follows

\[
\alpha(x, y) = \min\left(1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right).
\]

We update the new state of the Markov chain with probability \( \alpha(x, y) \), otherwise it remains at the value \( x \). Algorithm 1 summarizes the Metropolis-Hastings (MH) algorithm. Regardless of the initial value, the MH algorithm will converge to the target distribution \( \pi \).
Algorithm 1 Metropolis-Hastings Algorithm

1: Choose a starting value $x_0$.
2: for $k > 1$ do
3:   Sample $y \sim q(x_{k-1}, \cdot)$.
4:   Compute a Metropolis-Hastings acceptance ratio $\alpha(x_{k-1}, y)$.
5:   Accept $y$ with probability $\alpha(x_{k-1}, y)$. Otherwise $x_k = x_{k-1}$.

Tierney (1994) show that the transition kernel of Metropolis-Hastings algorithm satisfies the detailed balance condition for $\pi$. The MH transition kernel, $M : E \to E$, is,

$$M(x, dy) = \alpha(x, y)q(x, dy) + (1 - \alpha(x, y))\delta_x(dy)$$

$$= \left[1 \wedge \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right]q(x, dy) \cdot \left[1 - \left(1 \wedge \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right)\right] \delta_x(dy),$$

where $\delta_x$ is the Dirac delta function for $x \in E$. Using the MH acceptance probability $\alpha(x, y) \leq 1$, we can show that,

$$\pi(x)M(x, y) = \pi(x)\left[\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}q(x, y) + \left(1 - \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right)\delta_x(dy)\right]$$

$$= \pi(y)\left[q(y, x) + \left(\frac{\pi(x)}{\pi(y)} - \frac{q(y, x)}{q(x, y)}\right)\delta_x(y)\right]$$

$$= \pi(y)\left[\alpha(x, y)q(y, x) + (1 - \alpha(x, y))\delta_y(x)\right]$$

$$= \pi(y)M(y, x).$$

2.2.2 Gibbs Sampling (GS)

Gibbs sampling is applicable to problems in which the full conditional distribution of the element given the most recent values of all the other variables exits in a closed form, and we can easily sample from this full conditional distribution.

We let $x = (x_1, \cdots, x_D)$ denote a multivariate random variable, and let $x_k = (x_{k,1}, \cdots, x_{k,D})$ denote the $k$-th sample of $x$. Algorithm 2 displays the pseudo-code of a Gibbs sampler. In this algorithm, $x_{k,-d} = (x_{k,1}, \cdots, x_{k,d-1}, x_{k,-1,d+1}, \cdots, x_{k,-1,D})$.

Algorithm 2 Gibbs Sampling

1: Choose a starting value $x_0$.
2: for $k > 1$ do
3:   for $d = 1$ to $D$ do
4:     Sample $x_{k,d} \sim \pi(\cdot|x_{k,-d})$. 

Gibbs sampler can be regarded as a special case of the Metropolis-Hastings algorithm (Geman and Geman, 1984), with proposal distribution given by the full conditional posterior. In other words, Gibbs sampling is a Metropolis-Hastings with acceptance probability always equal to 1, shown as
follows:

\[
\alpha(x, (x_{-d}, y_d)) = \frac{\pi((x_{-d}, y_d))\pi(x_d|x_{-d})}{\pi(x)\pi(y_d|x_{-d})} = \frac{\pi(x_{-d})\pi(y_d|x_{-d})\pi(x_d|x_{-d})}{\pi(x_{-d})\pi(x_d|x_{-d})\pi(y_d|x_{-d})} = 1,
\]

where \((x_{-d}, y_d) = (x_1, \cdots, x_{d-1}, y_d, x_{d+1}, \cdots, x_D)\).

For many complex models, we may not be able to use Gibbs sampling to update all variables, as the Gibbs sampler requires the existence of full conditional distributions. In such cases, a hybrid of Gibbs sampling and Metropolis-Hastings algorithms can be used to sample from a complex target distribution. Even though in Gibbs sampling, we are always accepting samples. The Gibbs sampler can be ineffective when variables are strongly correlated or when sampling from the full conditionals is extremely expensive or inefficient.

### 2.2.3 Slice Sampler (SS)

Gibbs samplers are only applicable to problems in which the full conditional distributions exist in a closed form, and for which the full conditional distributions is relatively easy to sample from. Metropolis Hastings algorithms are an alternative to sampling from complex posterior distributions, while appropriate "proposal" distributions are required to sample efficiently. Neal (2003) propose slice sampling (SS) methods which can be applied to a wide variety of problems.

The basic idea of slice sampling is to sample random variables uniformly from the region below the graph of target distribution. First, we introduce an auxiliary variable \(y\). Then we define a joint distribution over \(x\) and \(y\) over the region below \(\gamma(x)\). Finally, we obtain a marginal distribution \(\pi(x)\) by discarding \(y\). Algorithm 3 displays a univariate slice sampler, with a "stepping out" procedure for finding a slice interval. We refer readers to Neal (2003) and Robert and Casella (2005) for other methods for finding a slice interval, and multivariate versions of slice sampler.

**Algorithm 3 A univariate slice sampler**

1. Choose a starting value \(x_0\).
2. for \(k > 1\) do
3. Sample \(y_k \sim \text{Uniform}(0, \gamma(x_{k-1}))\).
4. Use the "Stepping out" procedure to find an interval, \(I = (L, R)\), around \(x_{k-1}\) that contains all or much of the slice (see Algorithm 4).
5. Sample \(x_k\) from the slice within \(I = (L, R)\).

### 2.3 Sequential Monte Carlo (SMC)

Sequential Monte Carlo methods are a popular alternative to Markov chain Monte Carlo to sample from complex high dimensional distributions. In this section, we first provide a brief review of standard sequential Monte Carlo algorithms. We provide an illustration is based on an application
Algorithm 4 The “Stepping out” procedure for iteration $k$ of slice sampler

1: Set the current sample $x_{k-1}$, the vertical level defining the slice $y_k$, an estimate of the typical size of a slice $w$, and an integer $m$.
2: Sample $u_k \sim \text{Uniform}(0, 1)$.
3: Define $L_k = x_{k-1} - w \cdot u_k$.
4: Define $R_k = L_k + w$.
5: Sample $V_k \sim \text{Uniform}(0, 1)$.
6: Define $J_k = \text{Floor}(m \cdot V_k)$.
7: Define $K_k = (m - 1) - J_k$.
8: While $J_k > 1$ and $y_k < \gamma(L_k)$ do
9: \quad $L_k = L_k - w$.
10: \quad $J_k = J_k - 1$.
11: While $K_k > 0$ and $y_k < \gamma(R_k)$ do
12: \quad $R_k = R_k + w$.
13: \quad $K_k = K_k - 1$.

to state space models. Then, we review some commonly used resampling methods used in sequential Monte Carlo. Finally, we introduce a more general framework for sequential Monte Carlo.

### 2.3.1 Standard SMC

In this section, we introduce the standard sequential Monte Carlo algorithm to sample from a target distribution $\pi$ defined on a measurable space $(\mathcal{E}, \mathcal{F})$. We let $\pi(x)$ ($x \in \mathcal{E}$) denote a density function. In sequential Monte Carlo methods, we are required to introduce a sequence of intermediate probability distributions

$$\{\pi_r(x_r), r = 1, \cdots, R\}$$

defined on a sequence of measurable spaces $\{(\mathcal{E}_r, \mathcal{F}_r), r = 1, \cdots, R\}$ such that $\pi_R(x_R) = \pi(x)$. Here $r$ is a general index for the intermediate distributions. The index does not necessary have to be an index for time as in state space models. Typically, $x_r = x_{1:r} = (x_1, \cdots, x_r)$, where $x_r \in \mathcal{X}$ implying that $\mathcal{E}_r = \mathcal{X}^r$.

The distribution of each intermediate target is assumed to be known up to a normalizing constant. For $r = 1, \cdots, R$,

$$\pi_r(x_r) = Z_r^{-1} \gamma_r(x_r),$$

where we can evaluate $\gamma_r : \mathcal{E}_r \rightarrow \mathbb{R}^+$ pointwise. The normalizing constant $Z_r$ is unknown and generally involves intractable integration. We will denote $Z_R$ by $Z$.

State space models (Gordon et al., 1993; Fearnhead and Meligkotsidou, 2007; Buckland et al., 2007; Barndorff-Nielsen and Shephard, 2001; Fearnhead et al., 2008; Boys et al., 2008; Golightly and Wilkinson, 2006; Boys et al., 2008; Golightly and Wilkinson, 2006) are one of the most standard
We can rewrite the normalizing constant of \( \pi \) sequentially update \( \pi \) weighted samples (also called particles), of the observations given latent variables. We approximate the \( r \)-th intermediate distribution \( \pi_r \) is approximated by a set of \( K \) weighted samples (also called particles), \( \{x_{r,k}, W_{r,k}\}_{k=1,...,K} \). We approximate the \( r \)-th intermediate target distribution by particles and weights as follows

\[
\pi_{r,K}(dx_r) = \sum_{k=1}^{K} W_{r,k} \delta_{x_{r,k}}(dx_r).
\]

An estimate of the normalizing constant \( Z \) is a byproduct of the sequential Monte Carlo algorithm. We can rewrite the normalizing constant of \( \pi_1(x_1) \) as

\[
Z_1 = \int \frac{\gamma_1(x_1)}{q_1(x_1)} q_1(x_1) dx_1 = \int w_1(x_1) q_1(x_1) dx_1.
\]
Correspondingly, an estimate of $Z_1$ is

$$Z_{1,K} = \frac{1}{K} \sum_{k=1}^{K} w_{1,k}.$$  

Similarly, we can rewrite the ratio of the normalizing constants as

$$\frac{Z_r}{Z_{r-1}} = \frac{\int \gamma_r(x_r)dx_r}{Z_{r-1}} = \frac{\int \gamma_r(x_r)dx_r}{\gamma_{r-1}(x_{r-1})/\pi_{r-1}(x_{r-1})} \cdot \frac{\pi_{r-1}(x_{r-1})}{\pi_{r-1}(x_{r-1})} dx_r
= \int \frac{\gamma_r(x_r)}{\gamma_{r-1}(x_{r-1})} \pi_{r-1}(x_{r-1})q_r(x_{r-1}, x_r)dx_r
= \int w_r(x_r)\pi_{r-1}(x_{r-1})q_r(x_{r-1}, x_r)dx_r.$$  

Straightforwardly, an estimate of $Z_r/Z_{r-1}$ is provided by

$$\frac{Z_r}{Z_{r-1}} \approx \frac{1}{K} \sum_{k=1}^{K} w_{r,k}.$$  

Since the estimate of the normalizing constant can be rewritten as

$$Z \equiv Z_R = Z_1 \prod_{r=2}^{R} \frac{Z_r}{Z_{r-1}},$$

an estimate of the normalizing constant $Z$ is

$$Z_{R,K} = \prod_{r=1}^{R} \left( \frac{1}{K} \sum_{k=1}^{K} w_{r,k} \right), \quad (4)$$

which can be obtained from a sequential Monte Carlo algorithm readily. Moreover, Equation (4) is an unbiased estimates of $Z$ (Doucet and Johansen, 2009; Del Moral, 2004).

One important step in sequential Monte Carlo algorithms is the resampling step. The most commonly used resampling algorithm is multinomial resampling. The performance of sequential Monte Carlo algorithms can be improved by using more advanced resampling methods, such as stratified resampling. We will briefly review some resampling methods in the next section.

2.3.2 Resampling Methods

In sequential Monte Carlo algorithms, the variance of the weights $\{W_{r,k}\}$ tends to increase and all the mass tends to concentrate on a few number of particles, as the iteration $r$ increases. A resampling step is used to refine the approximation by pruning the particles with low weights and multiplying
the particles with high weights. The rationale is that we want to work on more promising particles. We believe that particles at iteration \( r \) with low weights will still have low weights at iteration \( r + 1 \).

In the following, we summarize multinomial resampling and stratified resampling.

**Multinomial Resampling**

Multinomial resampling is the simplest resampling scheme. In multinomial resampling, we reset the approximation by sampling \( K \) times from \( \tilde{x}_{r,k} \sim \pi_r(dx_r) \). The new approximation is

\[
\tilde{\pi}_{r,K}(dx_r) = \frac{1}{K} \sum_{k=1}^{K} \delta_{\tilde{x}_{r,k}}(dx_r).
\]

The above formula can be rewritten as

\[
\tilde{\pi}_{r,K}(dx_r) = \sum_{k=1}^{K} \frac{N_{r,k}}{K} \delta_{\tilde{x}_{r,k}}(dx_r),
\]

where \((N_{r,1}, \ldots, N_{r,K}) \sim \text{Multinomial}(K; W_{r,1}, \ldots, W_{r,K})\) thus

\[
E[N_{r,k}] = KW_{r,k}, \quad \text{var}[N_{r,k}] = KW_{r,k}(1 - W_{r,k}).
\]

**Stratified Resampling**

In Algorithm 6, we describe stratified resampling, which is a more advanced resampling scheme such that

\[
E[N_{r,k}] = KW_{r,k}, \quad \text{var}[N_{r,k}] < KW_{r,k}(1 - W_{r,k}).
\]

**Algorithm 6 Stratified Resampling**

1: Normalise the weights \( \{w_{r,k}\} \) of the \( K \) particles and label them according to the order of the corresponding \( x_{r,k} \) to obtain \( W_{r,k}, k = 1, \ldots, K \).
2: Construct the corresponding cumulative distribution function \( Q_r \).
3: Set \( Q_r(0) \equiv 0 \).
4: for \( k = 1, \ldots, K \) do
5: \hspace{1em} \( Q_r(k) \equiv \sum_{j\leq k} W_{r,j} \).
6: Sample \( U_1 \) uniformly on \([0, 1/K]\).
7: for \( j = 2, \ldots, K \) do
8: \hspace{1em} set \( U_j = U_1 + (j - 1)/K \).
9: for \( k = 1, \ldots, K \) do
10: \hspace{1em} If there exists \( j \in \{1, \ldots, K\} \) such that \( Q_r(k - 1) \leq U_j < Q_r(k) \), then \( x_{r,k} \) is resampled.
11: Update \( N_{r,k} = \#\{U_j : Q_r(k - 1) \leq U_j < Q_r(k)\} \).
2.3.3 General sequential Monte Carlo

Recent advances in sequential Monte Carlo focus on sampling from arbitrary sequences of distributions. The general sequential Monte Carlo framework proposed by Del Moral et al. (2006, 2007) is a very general method for obtaining a set of samples from a sequence of distributions which can exist on the same or different spaces. The general sequential Monte Carlo algorithm is a generalization of the standard sequential Monte Carlo method (Doucet et al., 2001) in which the target distribution exists on a space of strictly increasing dimension.

The sequence of intermediate target distributions can be defined on a common continuous space $\mathcal{X}$. Suppose that $\pi_r$ is the posterior distribution of a parameter given the data collected until time $r$, i.e. $\pi_r(x) = p(x|y_{1:r})$. We can construct a sequence of distributions that admit the distribution of interest, $\pi_r(x_r)$, as the recent iteration marginal

$$
\tilde{\pi}_r(x_r) = \pi_r(x_r) \prod_{j=1}^{r-1} L_j(x_{j+1}, x_j),
$$

where $L_j(x_{j+1}, x_j)$ is the artificial backward Markov kernels from iteration $j + 1$ to $j$. Then we apply the standard sequential Monte Carlo framework on this sequence of distributions. Suppose that the sample at iteration $r$ is

$$
x_{r,k} \sim K_r(x_{r-1,k}, \cdot),
$$

where $K_r$ is a Markov kernel defined on $E_{r-1} \times E_r$. Then, the resulting sampler has a weight update

$$
W_{r,k} \propto \frac{\pi_r(x_{r,k})L_{r-1}(x_{r,k}, x_{r-1,k})}{\pi_{r-1}(x_{r,k})K_r(x_{r-1,k}, x_{r,k})},
$$

which is different from the one in a standard sequential Monte Carlo.

**Algorithm 7 A general sequential Monte Carlo**

1: Sample $x_{1,k} \sim q_1(\cdot)$.
2: Set its unnormalized weight $w_{1,k} = \gamma_1(x_{1,k})/q_1(x_{1,k})$.
3: Normalize weights $W_{1,k} = w_{1,k}/\sum_{k=1}^{K} w_{1,k}$.
4: Resample $\{x_{1,k}, W_{1,k}\}$ to obtain new particles denoted $\{\tilde{x}_{1,k}\}$.
5: for $r \in 2, \ldots, R$ do
6: Sample $x_{r,k} \sim K_r(\tilde{x}_{r-1,k}, \cdot)$.
7: Compute

$$
w_{r,k} = w(\tilde{x}_{r-1,k}, x_{r,k}) = \frac{\gamma_r(x_{r,k})}{\gamma_{r-1}(\tilde{x}_{r-1,k})} \frac{L_{r-1}(x_{r,k}, \tilde{x}_{r-1,k})}{K_r(\tilde{x}_{r-1,k}, x_{r,k})}.
$$

8: Normalize weights $W_{r,k} = w_{r,k}/\sum_{k=1}^{K} w_{r,k}$.
9: Resample $\{x_{r,k}, W_{r,k}\}$ to obtain new particles denoted $\{\tilde{x}_{r,k}\}$.

Algorithm 7 displays a general sequential Monte Carlo algorithm. In general SMC, it is common to choose the forward kernels $K_r(x_{r-1}, x_r)$ to be $\pi_r$-invariant, typically Markov chain Monte Carlo kernels. A convenient backward Markov kernel that allows an easy evaluation of the importance
weight is
\[ L_{r-1}(x_r, x_{r-1}) = \frac{\pi_r(x_r)K_r(x_{r-1}, x_r)}{\pi_r(x_r)}. \]

With this backward kernel, the incremental importance weight can be evaluated by
\[
w_r = w(x_{r-1}, x_r) = \frac{\gamma_r(x_r)}{\gamma_{r-1}(x_{r-1})} \cdot \frac{L_{r-1}(x_r, x_{r-1})}{K_r(x_{r-1}, x_r)} \cdot \frac{1}{K_r(x_{r-1}, x_r)}
\]
\[= \frac{Z_r\pi_r(x_{r-1})}{\gamma_{r-1}(x_{r-1})} \propto \frac{\gamma_r(x_{r-1})}{\gamma_{r-1}(x_{r-1})}. \]

### 2.4 Particle Markov chain Monte Carlo

Andrieu et al. (2010) proposed particle Markov chain Monte Carlo methods (PMCMC), a class of efficient and flexible Markov chain Monte Carlo algorithms to approximate a target distribution (e.g. a posterior distribution \(\pi(x) \equiv p(x|y)\)). The PMCMC algorithms are combinations of Markov chain Monte Carlo and SMC algorithms, in which SMC methods are used to build efficient high dimensional proposal distributions at each of the MCMC steps. The PMCMC algorithms include the particle independent Metropolis Hastings (PIMH), particle marginal Metropolis Hastings (PMMH), and particle Gibbs sampler (PGS). We introduce PGS in this section, for the other PMCMC algorithms we refer readers to Andrieu et al. (2010).

#### 2.4.1 Particle Gibbs Sampler (PGS)

A standard Gibbs algorithm targeting \(\pi(\theta, x)\) iterates between sampling from the full conditionals \(\pi(\theta|x)\) and \(\pi(x|\theta)\). Generally, sampling from \(\pi(\theta|x)\) is relatively easy. While it could be a challenge to sample exactly from \(\pi(x|\theta)\), as the object \(x\) is often high dimensional, e.g. a large phylogenetic tree.

To sample from \(\pi(x|\theta)\), the particle Gibbs sampler uses conditional sequential Monte Carlo, to approximate the standard Gibbs sampler. In Algorithm 8, we describe conditional sequential Monte Carlo. The conditional sequential Monte Carlo sampler is similar to a standard sequential Monte Carlo algorithm, except for a reference trajectory. Conditional SMC ensures that a pre-specified particle and its ancestral lineage survive and samples the remaining \(K - 1\) particles as usual. Suppose the \(j\)-th particle is the reference one and we denote its ancestral lineage by \((A_{j1}^1, A_{j2}^2, \cdots, A_{jR}^R)\), where \(A_{jr}^j\) represents the index of the “parent” at iteration \(r - 1\) of particle \(x_{r,j}\) for \(r = 2, \cdots, R\). We describe the particle Gibbs sampler in Algorithm 9.
Algorithm 8 A conditional sequential Monte Carlo

1: At iteration $r = 1$,
2: for $k \neq A_j^1$ do
3: Sample $x_{1,k} \sim q_1(\cdot)$.
4: Set its unnormalized weight $w_{1,k} = \frac{\gamma_\theta(x_{1,k})}{q_1(x_{1,k})}$.
5: Resample $K - 1$ times from $\{x_{1,k}, w_{1,k}\}$ to obtain $\{\tilde{x}_{1,k} : k \neq A_j^1\}$ and set $\tilde{x}_{1,A_j^1} = x_{1,A_j^1}$.
6: for $r \geq 2$ do
7: for $k \neq A_j^r$ do
8: Sample $x_{r,k} \sim q_r(\tilde{x}_{r-1,k}, \cdot)$.
9: Set $x_{r,k} = (\tilde{x}_{r-1,k}, x_{r,k})$.
10: Compute $w_{r,k} = \frac{\gamma_\theta(x_{r,k})}{\gamma_\theta(\tilde{x}_{r-1,k} q_r(\tilde{x}_{r-1,k}, x_{r,k}))}$.
11: Resample $\{x_{r,k}, w_{r,k}\}$ to obtain new particles denoted $\{\tilde{x}_{r,k} : k \neq A_j^r\}$ and set $\tilde{x}_{r,A_j^r} = x_{r,A_j^r}$.

Algorithm 9 A particle Gibbs Sampler

1: Initialization: $i = 0$
2: Sample $\theta(0)$ arbitrarily.
3: Run a sequential Monte Carlo algorithm targeting $\pi(x|\theta(0))$.
4: Sample $x(0) \sim \pi_{R,K}(\cdot|\theta(0))$ and record its ancestral lineage.
5: for $i \geq 1$ do
6: Sample $\theta(i) \sim \pi(\cdot|x(i - 1))$.
7: Run a conditional sequential Monte Carlo algorithm targeting $\pi(x|\theta(i - 1))$ conditional on $x(i - 1)$ and its ancestral lineage.
8: Sample $x(i) \sim \pi_{R,K}(\cdot|\theta(i - 1))$. 
Chapter 3

Background on Phylogenetics

In this section, we provide some background on phylogenetics. In Section 3.1, we introduce phylogenetic trees. In Section 3.2, we introduce the data used for phylogenetic tree reconstruction. In Section 3.3, we review model-based methods for phylogenetic reconstruction.

3.1 Phylogenetic tree

We let $X$ denote a set of taxa (e.g. modern species). For each taxa, we observe its biological sequence, a string of symbols or characters from a finite alphabet $\Sigma$. For example, the sequence could be DNA sequences composed of four nucleotides $\{A, C, G, T\}$. A phylogenetic $X$-tree represents the relationship among taxa via a tree topology, which is represented by a connected acyclic graph, $(V, E)$. Here $V$ is the set of vertices for the observed and unobserved taxa and $E$ is the set of edges between taxa. The vertices in $X$ are called leaves, which only have one neighbour. A tree is a connected graph without cycles.

In this thesis, we only consider binary phylogenetic $X$-trees, which is also the main type of trees studied in the phylogenetics literature. Each vertex of a binary phylogenetic tree has a maximum of three neighbors. A binary phylogenetic tree can be rooted or unrooted. A rooted phylogenetic tree is a directed tree with a unique node corresponding to the most recent common ancestor of all the leaf taxa of the tree. In a rooted binary tree, vertices with one, two, and three neighbors correspond to leaves, root, and internal nodes, respectively. In contrast, an unrooted phylogenetic tree represents the relatedness of leaves without making assumptions about their common ancestor. Branch lengths are positive real numbers associated with each edge, specifying the amount of evolution between nodes.

A rooted phylogenetic $X$-tree describes the evolution of a set of species in the tree from a common ancestor at the root. We regard the edges as being directed away from the root. In ultrametric (or clock) trees, the leaf nodes are all assumed equally distant from the root. With the molecular clock assumption (Hasegawa et al., 1985), we can specify the edge lengths of a clock tree via the distances among the taxa. Typically, we reparameterize the branch lengths of a clock tree as distances between heights from the leaves.
The clock assumption of a phylogenetic tree is restricted and unrealistic as it implies a constant evolutionary rate. In some cases, we use a *non-clock tree* to model a more general tree without these constraints. A non-clock tree is generally defined without a rooted assumption. However, most evolutionary models assume the evolutionary events are reversible such that the likelihood of an unrooted tree is equivalent to the likelihood of a rooted tree by rooting at an arbitrary point in the tree.

### 3.2 Data for phylogenetics

#### 3.2.1 Data Types

Modern phylogenetics mainly focus on studying phylogenetic reconstruction at a molecular level. The three major macromolecules for living species are *Deoxyribonucleic acid* (DNA), *Ribonucleic acid* (RNA), and proteins (which are derived from amino acids). Hence, DNA, RNA and amino acid data are most commonly-used types of data for phylogenetic reconstruction. The projects in this thesis mainly use the DNA sequences to reconstruct phylogenetic trees. Usually DNA consists of two complementary chains twisted around each other. Each chain is composed of four nucleotides, two purines: adenine (A) and guanine (G), and two pyrimidines: cytosine (C) and thymine (T). Though we use DNA sequences as examples to illustrate our methods, the methodologies are general enough to cope with other type of biological sequence data.

The sequence data we used for analysis are aligned in columns using multiple sequence alignment (MSA) algorithms, e.g. ClustalW (Larkin et al., 2007; Goujon et al., 2010) and T-coffee C. et al. (2000). We let $\mathcal{Y}$ denote the set of observed molecular sequences for tree reconstruction. In addition, we assume that the form of $\mathcal{Y}$, is a matrix $y_{x,s}$, where $x \in X$ and $s$ denotes an aligned position on the genomes, called a site (or marker). In our phylogenetic projects, we will assume sites are independent.

### 3.3 Model-based methods for phylogeny reconstruction

Many methodologies are proposed for phylogeny reconstruction, including score-based approaches and model-based approaches. We refer readers to Holder and Lewis (2003) for a good review of these methods.

In this thesis, we mainly focus on model-based tree reconstruction as score-based methods may not be easily estimated and interpreted. In model-based approaches, we model the evolution of nucleotide, codon, or amino acid sequences via stochastic models. In contrast to score-based approaches, we can interpret parameters and models more explicitly. Hypothesis testing and model comparison are allowed via stochastic modelling.

The main approaches for estimating phylogenetic trees include maximum likelihood methods (Felsenstein, 1981; Kishino et al., 1990; Guindon and Gascuel, 2003) and Bayesian methods (Rannala and Yang, 1996; Yang and Rannala, 1997; Mau et al., 1999; Larget and Simon, 1999; Li et al., 2000).
Both methods depend on a probabilistic model of evolution. Before introducing details on Bayesian phylogenetics, we first review stochastic evolutionary models.

### 3.3.1 Stochastic Evolutionary Models

#### Continuous time Markov Chain (CTMC) over characters

In model-based phylogeny reconstruction, we use a stochastic process called the Continuous time Markov Chain (CTMC) to specify the transition probability between a node \( v \) and its parent \( \varrho(v) \). The CTMC is defined on \( S = [0, T] \), which represents the time interval for evolution, and a countable domain \( \Sigma \) (e.g. \( \Sigma = \{A, C, G, T\} \) for DNA data) is assumed for \( Y_s \).

We use \( P \) to denote the transition probabilities, in which the \((i, j)\)-th element, \( P_{ij}(T) \) denotes the probability of transitioning from \( Y_0 = i \) to \( Y_T = j \) along a branch of length \( T \). The matrix \( P \) satisfies a matrix differential equation \( \frac{dP}{dt} = PQ \) with initial condition \( P(0) = I \), where \( I \) is the identity matrix, and \( Q \) is called the rate matrix.

The Markov assumption \( P(T + T^*) = P(T)P(T^*) \) implies the solution of \( \frac{dP}{dt} = PQ \) is

\[
P(T) = \exp (QT) = I + \sum_{i=1}^{\infty} Q^i T^i / i!,
\]

where \( \exp \) denotes the matrix exponential, and \( I \) denotes the \( |\Sigma| \times |\Sigma| \) identity matrix.

In practice, we use diagonalization to compute the matrix exponential. We let \( \lambda_1, \cdots, \lambda_{|\Sigma|} \) denote the eigenvalues of \( Q \), and let \( U \) denote the orthogonal matrix. The columns in \( U \) are composed of the corresponding eigenvectors. Using the diagonalization \( Q = U \Lambda U^{-1} \), Equation (1) becomes

\[
P(T) = \exp(QT) = U \text{diag}(\exp(\lambda_1 T, \lambda_2 T, \cdots, \lambda_{|\Sigma|} T)) U^{-1}.
\]

There are rich evolutionary models for biological sequences. One of the popular model is the HKY85 model (Hasegawa et al., 1985). In HKY85, the rate matrix is modelled by

\[
Q = \begin{bmatrix}
- \pi_C r_{tv} & \pi_G r_{ri} & \pi_T r_{tv} \\
\pi_A r_{tv} & - \pi_G r_{ri} & \pi_T r_{ri} \\
\pi_A r_{ri} & \pi_C r_{tv} & - \pi_T r_{tv} \\
\pi_A r_{tv} & \pi_C r_{ri} & \pi_G r_{tv} & -
\end{bmatrix},
\]

where \( r_{tv} \) is the transversion rate (i.e. a mutation between two pyrimidines (T \( \leftrightarrow \) C) or two purines (A \( \leftrightarrow \) G)), and \( r_{ri} \) is the transition (i.e. a mutation between a pyrimidine and a purine (A \( \leftrightarrow \) C, A \( \leftrightarrow \) T, G \( \leftrightarrow \) C, or G \( \leftrightarrow \) T)), and \( (\pi_A, \pi_C, \pi_G, \pi_T)' \) is the stationary distribution of CTMC. The notation \( - \) is short for minus the sum of all the other elements in the same row. For example, \( - \) in the first row of \( Q \) represents \( -(\pi_C r_{tv} + \pi_G r_{ri} + \pi_T r_{tv}) \).
We often reparameterize the HKY 85 model by the ratio of transition to transversion, i.e. \( \kappa = r_{ti}/r_{tv} \),

\[
Q = \alpha \begin{bmatrix}
- & \pi_C & \kappa\pi_G & \pi_T \\
\pi_A & - & \pi_G & \kappa\pi_T \\
\kappa\pi_A & \pi_C & - & \pi_T \\
\pi_A & \kappa\pi_C & \pi_G & -
\end{bmatrix},
\]

where the parameter \( \alpha \) represents an overall transversion rate.

Another popular model for mutation is known as the Kimura two-parameter model (Kimura, 1980). This model is a special case of the HKY85 model, in which the stationary state frequencies are fixed to be equal, i.e. \( \pi_A = \pi_C = \pi_G = \pi_T = 0.25 \). The parameter \( \alpha \) and branch lengths are not identifiable using the Kimura two-parameter model. Without loss of generality, we assume \( \alpha = 1 \). In this thesis, we set \( \kappa = 2 \) as the default in the Kimura two-parameter model.

Let \( \theta \) denote the vector of parameters of the evolutionary model; \( \theta \in \Theta \), we call this \( \theta \) the evolutionary parameters. For example, we only have one parameter \( \kappa \) in the Kimura two-parameter model. Recall that we denote the observed biological sequences by \( \mathcal{Y} \). We let \( t \) denote a phylogenetic tree in a tree space \( \mathcal{X} \). The likelihood \( L(\theta, t, \mathcal{Y}) \) is computed recursively by the sum-product algorithm. We use \( L_v(j) \) to denote the probability of everything observed below the internal node \( v \), conditional on \( v \) having character \( j \). For the recursion, we first compute the likelihoods of leaves. If the leaf \( v \) has an observation \( i \), let \( L_v(i) = 1 \), and \( L_v(j) = 0 \) for all \( j \in \Sigma, j \neq i \). We express this recursion as follows:

\[
L_v(i) = \begin{cases} 
1, & \text{if } v \in X \text{ and } \mathcal{Y}(v) = i \\
0, & \text{if } v \in X \text{ and } \mathcal{Y}(v) \neq i \\
\prod_{v \in \mathcal{G}(v)} \left[ \sum_j P_{ij}(b_v)L_v(j) \right], & \text{if } v \text{ is an internal node}
\end{cases}
\]

where \( P_{ij}(b_v) \) is the transition probability from \( i \) to \( j \) along the branch of length \( b_v \), computed by Equation (1). The recursive algorithm continues the until it computes \( L_0(i) \), the probability of the tree with the root having character \( i \). Finally, the likelihood is computed by

\[
L(\theta, t, \mathcal{Y}) = \sum_{i \in \Sigma} \pi_i L_0(i).
\]

### 3.3.2 Maximum Likelihood

In maximum likelihood inference, we maximize the likelihood function in Equation (3) to estimate trees and evolutionary parameters. We search through a multidimensional parameter space to maximize Equation (3). The computational cost of the maximum likelihood depends on the thoroughness of the search.

Although maximum likelihood (Felsenstein, 1981) is able to provide point estimates of phylogenies, it cannot provide confidence intervals for estimated trees (Goldman et al., 2000). Nonparametric bootstrap (Felsenstein, 1985) is commonly used to assess the uncertainty of estimated phylogenies.
However, the confidence intervals provided by bootstrap methods are too conservative compared to Bayesian methods. In the next section, we introduce Bayesian methods for phylogenetic inference.

### 3.3.3 Framework of Bayesian Phylogenetics

With recent advances of computing techniques, Bayesian phylogenetics is becoming more popular as it has several advantages over other approaches. First, complex evolutionary models are able to be implemented in Bayesian phylogenetics. Second, Bayesian phylogenetics automatically consider the uncertainty of estimated phylogenetic trees. These methods are computationally cheaper approaches for assessing the support for trees than maximum likelihood bootstrapping (Douady et al., 2003).

The evolutionary models used in Bayesian phylogenetics are the same as maximum likelihood approaches, but with specification of prior distributions for both the phylogenetic tree and the evolutionary parameters. The prior distributions incorporate previous knowledge of phylogenetic inference gathered from beyond the dataset used for analysis. Bayesian methods estimate the posterior distribution for both the tree and evolutionary parameters.

Recall that $Y$ is a set of observations on the leaves of a phylogenetic $X$-tree. For $X' \subset X$, we use the notation $Y(X')$ for the subset of observations corresponding to a subset $X'$ of the leaves. Our objective is to use $n$ observed biological sequences, $Y$, to estimate a phylogenetic tree using Bayesian approaches. Let $\theta \in \Theta$ denote the vector of evolutionary model parameters. Let $p(\theta)$ denote the prior distribution for $\theta$. For a tree $t \in X$, we denote the density of the prior distribution given $\theta$ by $p(t|\theta)$.

The probability model for observed data $Y$ given $\theta$ and $t$ is $P(Y|\theta, t)$. Bayesian inference aims to infer the normalized posterior density

$$
\pi(\theta, t|Y) = \frac{P(Y|\theta, t)p(t|\theta)p(\theta)}{\int_{\theta} \int_{t} P(Y|\theta, t)p(t|\theta)p(\theta) dt d\theta}
$$

where $P(Y|\theta, t)$ is the likelihood function.

The marginal likelihood function $\int_{\theta} \int_{t} P(Y|\theta, t)p(t|\theta)p(\theta) dt d\theta$ for phylogenetic models is too complex to evaluate analytically, so we use Monte Carlo methods to approximate the posterior distribution of phylogenetic trees and evolutionary parameters.

### 3.3.4 Bayesian Phylogenetics via MCMC

Markov chain Monte Carlo is a popular approach to approximate posterior distribution of phylogenetic trees (Rannala and Yang, 1996; Yang and Rannala, 1997; Mau et al., 1999; Larget and Simon, 1999; Li et al., 2000; Holder and Lewis, 2003; Lakner et al., 2008; Höhna et al., 2008; Höhna and Drummond, 2012). Many user-friendly software packages have been developed for implementing Markov chain Monte Carlo for phylogenetics (e.g. BAMBE (Simon and Larget, 2000), MrBayes (Huelsenbeck and Ronquist, 2001; Ronquist and Huelsenbeck, 2003; Ronquist et al., 2012), and BEAST (Drummond and Rambaut, 2007)).
One challenge in MCMC for Bayesian phylogenetic inference is to explore the posterior of phylogenetic trees efficiently. Due to the combinatorial constraints of phylogenetic tree space, the posterior tree distribution is a very complex multimodal distribution. It is difficult to design very efficient tree proposal distributions. A “bold” move leads to a low acceptance rate of MH move, while a “small” tree move may get stuck in a local mode. These proposals require a long run of an MCMC algorithm to fully explore the posterior distribution of tree distribution.

With a few exceptions (Lakner et al., 2008; Höhna et al., 2008; Höhna and Drummond, 2012), the MCMC moves have remained unchanged in most literatures in the past decade (Larget and Simon, 1999). Lakner et al. (2008) proposed two variates of proposal distributions to modify tree topology and branch lengths for unrooted trees simultaneously. Höhna et al. (2008) investigated proposal distributions for constrained rooted clock trees. Höhna and Drummond (2012) studied new proposal distributions to improve the efficiency of Markov chain Monte Carlo algorithms.

In this thesis, the proposal distributions for phylogenetic trees are listed as follows:

1. The multiplicative branch proposal: this proposal picks one edge at random and multiplies its current value by a random number distributed uniformly in \([1/a, a]\) for some fixed parameter \(a > 1\) (controlling how bold the move is) (Lakner et al., 2008).

2. The global multiplicative branch proposal: this proposal proposes modifying all the branch lengths by applying the above multiplicative branch proposal to each branch.

3. The stochastic NNI proposal: we consider the nearest neighbor interchange (NNI) (Jow et al., 2002) to propose a new tree topology. NNI randomly picks an internal edge, and exchanges the connectivity of four subtrees formed by this picked edge within the main tree.

4. The stochastic NNI proposal with resampling the edge: this proposal uses the above NNI proposal in (3) and the multiplicative branch proposal in (1) for the edge under consideration.

5. The Subtree Prune and Regraft (SPR) move: this proposal selects and removes a subtree from the main tree and reinserts it elsewhere on the main tree to create a new tree.

### 3.4 Consensus tree and tree distances

In this section, we introduce how to summarize a set of phylogenetic trees, e.g. MCMC posterior samples. In addition, we introduce a method to measure the distance between two trees for the purpose of evaluating the quality of the estimated tree.

Unlike summarizing a set of real number (for example, with a mean), summarizing a set of trees is more complicated. Trees are composed of topologies and branch lengths. The summary tree is usually referred to the consensus tree (Larget and Simon, 1999). In this thesis, we use the majority-rule consensus tree to summarize posterior tree samples. The majority-rule consensus tree consists of clades that are present in no less than a half of the trees (Felsenstein, 1981).
In this thesis, the distance between a reference tree \( t \) and an estimated consensus tree \( t' \) is measured using the Robinson Foulds metric (Robinson and Foulds, 1979), the partition metric (Felsenstein, 2003), and Kuhner Felsenstein metric (Kuhner and Felsenstein, 1994). In order to define these metrics, for a rooted tree, we first discard the root to obtain an unrooted tree. Bipartition is defined as the two subsets of whole set formed by removing one branch. We denote the set of all bipartitions of \( t \) by \( S(t) = \{B_i, i = 1, \ldots, n_e\} \), where \( B_i \) is the bipartition by removing edge \( i \). We denote the set of different bipartitions of \( t \) and \( t' \) by \( D(t, t') = S(t) \Delta S(t') \), where \( A_1 \Delta A_2 \) denotes the symmetric difference between sets \( A_1 \) and \( A_2 \). The partition metric of \( t \) and \( t' \) is defined as the number of their different bipartitions, denoted as \( |D(t, t')| \). The Robinson Foulds metric of \( t \) and \( t' \) is defined as \( \sum_{B \in D(t, t')} |b(B; t) - b(B; t')| \), where \( b(B; t) \) denotes the length of the branch corresponding to the bipartition \( B \) on tree \( t \). The Kuhner Felsenstein metric is defined as \( \sum_{B \in D(t, t')} (b(B; t) - b(B; t'))^2 \).

The partition metric of \( t \) and \( t' \) is normalized to make the distance more comparable:

\[
\text{Normalized partition} = \frac{|D(t, t')|}{(|S(t)| + |S(t')|)},
\]  

(5)
we also normalize the RF metric of \( t \) and \( t' \):

\[
\text{Normalized RF} = \frac{\sum_{B \in D(t, t')} |b(B; t) - b(B; t')|}{(\sum_i n_i b(B_i; t) + \sum_j n_j' b(B_j; t'))},
\]

(6)
Chapter 4

An Annealed Sequential Monte Carlo Method for Bayesian Phylogenetics

4.1 Introduction

The Bayesian paradigm is widely used in systematic biology, principally for the purpose of phylogenetic reconstruction as well as for evaluating the empirical support of evolutionary models (Chen et al., 2014). Both of these tasks, Bayesian phylogenetic reconstruction and model selection, involve an intractable sum over topologies as well as a high dimensional integral over branch lengths and evolutionary parameters. Consequently, Markov chain Monte Carlo (MCMC) methods have been widely used in the past twenty years to approximate posterior distributions defined over the space of phylogenetic trees (Rannala and Yang, 1996).

Despite their success, MCMC phylogenetic methods are still afflicted by two key limitations, hence motivating the need for alternative approximations method for posterior distributions over phylogenetic trees.

Firstly, MCMC methods do not readily take advantage of highly parallel computer architectures. This is problematic in the current context as progress in computational power mostly comes in the form of parallelism gains. While there are techniques available to parallelize phylogenetic MCMC methods, they are generally not “embarrassingly parallel”: for example, parallel Metropolis coupled MCMC (Altekar et al., 2004) may reach a point where the addition of cores actually reduces sampling efficiency (Atchadé et al., 2011).

A second challenge with MCMC-based phylogenetic approximations arises in the context of model selection. By comparing the marginal likelihood \(Z = p(y)\), where \(y\) denotes observed data, under different models, one can approach scientific questions under the Bayesian framework while naturally taking into account differences in model complexity. More specifically, the ratio \(r = p_1(y)/p_2(y)\) of two marginal likelihoods based on two evolutionary models, \(p_1(\cdot)\), \(p_2(\cdot)\), can be used to assess the strength of evidence \(y\) provides for \(p_1\) (when \(r > 1\)) or \(p_2\) (when \(r < 1\)). The ratio \(r\) is called the Bayes factor (Jeffreys, 1935; Lartillot et al., 2006; Oaks et al., 2018). In the context of
phylogenetics, the Bayes factor assesses how much support a set of sequencing data provides for one evolutionary model against another one.

Several methods have been proposed to estimate marginal likelihoods based on MCMC methods (Newton and Raftery (1994); Gelman and Meng (1998); Friel and Pettitt (2008), *inter alia*), including work tailored to the phylogenetic context (Huelsenbeck et al., 2004; Lartillot et al., 2006; Xie et al., 2010; Fan et al., 2010). However these methods all have different drawbacks, see for example the aptly named review, “Nineteen dubious ways to compute the marginal likelihood of a phylogenetic tree topology” (Fourment et al., 2018). Moreover, one limitation shared by all MCMC-based marginal likelihood estimators is that they are generally biased (in the technical sense of the term as used in computational statistics, reviewed in the theory section of the paper)—unless one is able to initialize the MCMC chains to the exact stationary distribution, which in practice is not possible. We argue that in certain scenarios, it can be useful to have unbiased methods. One example we elaborate on is for the purpose of a new test to ascertain correctness of posterior simulation software. Another class of examples comes from the burgeoning field of pseudo-marginal methods (Andrieu and Roberts, 2009).

Sequential Monte Carlo (SMC) methods (see Doucet and Johansen (2009) for an accessible introduction to SMC) provide a flexible framework to construct unbiased estimators and past work has shown they can be very efficient in a phylogenetic context (Teh et al., 2008; Görür and Teh, 2009; Bouchard-Côté et al., 2012; Görür et al., 2012; Wang et al., 2015; Everitt et al., 2016; Dinh et al., 2017; Smith et al., 2017; Fourment et al., 2018). One drawback caused by the high degree of flexibility that comes with SMC is that the phylogenetic SMC algorithms developed so far are non-trivial to adapt to existing MCMC-based phylogenetic frameworks. Here we propose a different construction based on the seminal work of Del Moral et al. (2006), in turn based on annealed importance sampling (AIS) (Neal, 2001), which yields an SMC method which is in a sense much closer to standard MCMC, while providing unbiased estimators of the marginal likelihood. The proposed method, which we call phylogenetic annealed SMC, can directly make use of any existing phylogenetic MCMC proposals, a rich literature covering many kinds of phylogenetic trees (Rannala and Yang, 1996; Yang and Rannala, 1997; Mau et al., 1999; Larget and Simon, 1999; Li et al., 2000; Holder and Lewis, 2003; Rannala and Yang, 2003; Lakner et al., 2008; Höhna et al., 2008; Höhna and Drummond, 2012). It is easy to incorporate the proposed annealed SMC into existing phylogenetic software packages that implement MCMC algorithms, such as RevBayes (Höhna et al., 2016) or BEAST (Drummond and Rambaut, 2007). At the same time, our method can leverage state-of-the-art advances in the field of adaption of SMC algorithms, making the algorithm fully automated in most cases.

Our implementation of the proposed method is available at https://github.com/liangliangwangsfu/annealedSMC. All our experimental setups and results are available at https://github.com/shijiaw/AnnealingSimulation. The algorithms described here are also available in the Blang probabilistic programming language https://github.com/UBC-Stat-ML/blangSDK, which supports a small but growing set of phylogenetic models.
4.2 Literature review

There is a growing body of work on SMC-based Bayesian phylogenetic inference. Indeed, a powerful feature of the general SMC framework (Del Moral et al., 2006) is that the space on which the distributions $\pi_r$ are defined is allowed to vary from one iteration to the next. All previous work on SMC methods for phylogenetics has exploited this feature for various purposes reviewed here.

In one direction, several “bottom up” approaches (Teh et al., 2008; Görür and Teh, 2009; Bouchard-Côté et al., 2012; Görür et al., 2012; Wang et al., 2015) have been proposed to allow more efficient reuse of intermediate stages of the Felsenstein pruning recursions. For these methods, the intermediate distributions are defined over forests over the observed taxa, and hence their dimensionality increases with $r$. These methods are most effective in clock or nearly-clock trees. For general trees, it is typically necessary to perform additional MCMC steps, which makes it harder to use in the context of estimation of marginal likelihoods.

In a related direction, Dinh et al. (2017) and Fourment et al. (2018) use a sequence of targets where $\pi_r$ is a tree over the first $r$ tips. This construction is especially useful in scenarios where taxonomic data come in an online fashion.

Another use case of SMC methods in phylogenetics arises from Bayesian analysis of intractable evolutionary models. For example, SMC has been used for Bayesian phylogenetic analysis based on infinite state-space evolutionary models (Hajiaghayi et al., 2014) or for joint inference of transmission networks (Smith et al., 2017).

Finally, a concurrent line of work (Everitt et al., 2016) has explored a combination of reversible jump methods with phylogenetic models.

One drawback of letting the dimensionality of $\pi_r$ vary with $r$ as all the above methods do, is that it makes it significantly harder to incorporate SMC into existing Bayesian phylogenetic inference packages such as MrBayes (Huelsenbeck and Ronquist, 2001), RevBayes or BEAST. In contrast, in our method the target distributions $\pi_r$ are all defined over the same space. The annealed SMC framework in this context utilizes Metropolis-Hastings kernels in the inner loop but combines them in a different fashion compared to standard MCMC algorithms, or even compared to parallel tempering MCMC algorithms.

4.3 Setup and notation

We let $t$ denote a phylogenetic tree with tips labelled by a fixed set of operational taxonomic units $X$. The variable $t$ encapsulates the tree topology and a set of positive branch lengths. Our methodology is directly applicable to any class of phylogenetic trees where MCMC proposal distributions are available. This includes for example clock trees (Höhna et al., 2008) as well as non-clock trees (Lakner et al., 2008).

We let $\theta$ denote evolutionary parameters, for example the parameters of a family of rate matrices such as the general time reversible (GTR) model (Tavaré, 1986), or diffusion parameters in the case
of continuous traits (Lemey et al., 2010). Again our method is applicable to any situation where
MCMC proposals are available for exploring the space of \( \theta \). We use \( x = (t, \theta) \) to denote these two latent variables.

We let \( y \) denote observed data indexed by the tips \( X \) of \( t \). We assume a likelihood function \( p(y|x) \)
is specified such that for any hypothesized tree and parameters, the value \( p(y|x) \) can be computed efficiently. This assumption is sometimes called pointwise computation. This is a typical assumption in Bayesian phylogenetics, where this computation is done with some version of Felsenstein pruning (Felsenstein, 1973, 1981) (an instance of the Forward-Backward algorithm (Forney, 1973)).

Finally, let \( p(x) \) denote a prior on the parameters and trees, which we assume can also be computed pointwise efficiently. This defines a joint distribution, denoted \( \gamma(x) = p(x)p(y|x) \). We ignore the argument \( y \) from now on since the data is viewed as fixed in a Bayesian analysis context.

We are interested in approximating a posterior distribution on \( x \) given data \( y \), denoted:

\[
\pi(x) = \frac{\gamma(x)}{\int \gamma(x') \, dx'}.
\]

Here the integral and \( dx' \) are viewed in an abstract sense and include both summation over discrete latent variables such as topologies and standard integration over continuous spaces.

The denominator can be interpreted as the marginal likelihood under the model specified by the prior and likelihood functions, which we denote by \( Z \):

\[
Z = p(y) = \int \gamma(x) \, dx.
\]

Computation of this quantity, also called the normalization constant or evidence, is the main challenge involved when doing Bayesian model selection.

Other quantities of interest include expectations with respect to the posterior distribution, characterized by a real-valued function of interest \( f \) based on which we would like to compute

\[
\int \pi(x)f(x) \, dx.
\]

For example if we seek a posterior clade support for a subset \( X' \subset X \) of the leaves \( X \),

\[
f(x) = f(t, \theta) = 1[t \text{ admits } X' \text{ as a clade}],
\]

where \( 1[s] \) denotes the indicator function which is equal to one if the boolean expression \( s \) is true and zero otherwise.
4.4 Annealed SMC for phylogenetics

4.4.1 Sequences of Distributions

In standard MCMC methods, we are interested in a single probability distribution, the posterior distribution. However, there are several reasons why we may use a sequence of distributions rather than only one.

A first possibility is that we may have an online problem, where the data is revealed sequentially and we want to perform inference sequentially in time based on the data available so far. The distribution at step \( r \) is then the posterior distribution conditioning on the first \( r \) batches of data. This approach is explored in the context of phylogenetics in Dinh et al. (2017), where a batch of data consists in genomic information for one additional operational taxonomic unit. We do not pursue this direction here, but discuss some possibilities for combinations in the discussion.

A second reason for having multiple distributions, and the focus of this work, is to facilitate the exploration of the state space. This is achieved for example by raising the likelihood term to a power \( \phi_r \) between zero and one, which we multiply with the prior

\[
\gamma_r(x) = p(y|x)^{\phi_r} p(x).
\]

MCMC may get stuck in a region of the space of phylogenetic trees around the initial value. This may happen for example around a local maximum (mode) in the posterior density. Such a region is sometimes called a “basin of attraction”, and no single basin of attraction may be enough to well represent the full posterior distribution. Introducing a series of powered posterior distributions can alleviate this issue. A small value of \( \phi_r \) flattens the posterior and makes MCMC samplers move easily between the different basins of attractions. The samples are initially overly dispersed but are then coerced into the posterior distribution \( \pi(x) \) by slowly increasing the annealing parameter \( \phi_r \).

We do not anneal the prior to ensure that \( \gamma_r(x) \) has a finite normalization constant,

\[
\int \gamma_r(x) \, dx = \mathbb{E}_{p(x)}[(p(y|X))^{\phi_r}]
\leq \left( \mathbb{E}_{p(x)}[p(y|X)] \right)^{\phi_r} = (p(y))^{\phi_r} < \infty,
\]

where the first inequality follows from the concavity of \( (-)^{\phi_r} \) and Jensen’s inequality.

A third scenario is that we may encounter a “tall data” problem, e.g. biological sequences with a large number of sites. When the number of sites is large, evaluation of the unnormalized posterior \( \gamma_r(x) \) defined in Equation (10) is computationally expensive. The idea of data subsampling (Quiroz et al., 2018; Bardenet et al., 2017; Gunawan et al., 2018; Quiroz et al., 2018) could be used to define the sequence of distributions. The construction of the sequence of distributions is described in Appendix 1.
The probability distributions

\[ \pi_r(x) = \frac{\gamma_r(x)}{\int \gamma_r(x') \, dx'} \]  

(5)

are therefore well defined and we denote their respective normalization constants by

\[ Z_r = \int \gamma_r(x) \, dx. \]  

(6)

If the exponent \( \phi_r \) is zero, then the distribution \( \pi_r \) becomes the prior which is often easy to explore and in fact independent samples can be extracted in many situations. At the other extreme, the distribution at power \( \phi_r = 1 \) is the distribution of interest.

The intermediate distributions \( \{ \pi_r \}_{r=1,...,R} \) are defined on a common measurable space \( (\mathcal{X}, \mathcal{E}) \). The annealed SMC is a generalization of the standard SMC method (Doucet et al., 2001). In standard SMC, the intermediate distributions are defined on a space of strictly increasing dimension.

4.4.2 Basic Annealed SMC Algorithm

We now turn to the description of annealed SMC in the context of Bayesian phylogenetic inference. The algorithm fits into the generic framework of SMC samplers (Del Moral et al., 2006): at each iteration, indexed by \( r = 1, 2, \ldots, R \), we maintain a collection indexed by \( k \in \{1, 2, \ldots, K\} \) of imputed latent states \( x_{r,k} \), each paired with a non-negative number called a weight \( w_{r,k} \); such a pair is called a particle. A latent state in our context consists in a hypothesized tree \( t_{r,k} \) and a set of evolutionary parameters \( \theta_{r,k} \), i.e. \( x_{r,k} = (t_{r,k}, \theta_{r,k}) \). In contrast to previous SMC methods, \( x_{r,k} \) is always of the same data type: no partial states such as forest or trees over subsets of leaves are considered here.

A particle population consists in a list of particles \( (x_r, w_r) = \{(x_{r,k}, w_{r,k}) : k \in \{1, \ldots, K\}\} \). A particle population can be used to estimate posterior probabilities as follows: first, normalize the weights, denoted after normalization using capital letter, \( W_{r,k} = w_{r,k} / \sum_{k'} w_{r,k'} \). Second, use the approximation:

\[ \int \pi_r(x)f(x) \, dx \approx \sum_{k=1}^{K} W_{r,k} f(x_{r,k}). \]  

(7)

For example if we seek a posterior clade support for a subset \( X' \subset X \) of the leaves \( X \), this becomes

\[ \sum_{k=1}^{K} W_{r,k} 1[\text{sampled tree } t_{r,k} \text{ admits } X' \text{ as a clade}]. \]

The above formula is most useful at the last SMC iteration, \( r = R \), since \( \pi_R \) coincides with the posterior distribution by construction.

At the first iteration, each of the particles’ tree and evolutionary parameters are sampled independently and identically from their prior distributions. We assume for simplicity that this prior
sampling step is tractable, a reasonable assumption in many phylogenetic models. After initialization, we therefore have a particle-based approximation of the prior distribution. Intuitively, the goal behind the annealed SMC algorithm is to progressively transform this prior distribution approximation into a posterior distribution approximation.

To formalize this intuition, we use the sequence of distributions introduced in the previous section. The last ingredient required to construct an SMC algorithm is an SMC proposal distribution $K_r(x_{r-1,k}, x_{r,k})$, used to sample a particle for the next iteration given a particle from the previous iteration. Since $x_{r-1,k}$ and $x_{r,k}$ have the same dimensionality in our setup, it is tempting to use MCMC proposals $q_r(x_{r-1,k}, x_{r,k})$ in order to build SMC proposals, for example, subtree prune and regraft moves, and Gaussian proposals for the continuous parameters and branch lengths. Indeed, there are several advantages of using MCMC proposals as the basis of SMC proposals. First, this means we can leverage a rich literature on the topic (Rannala and Yang, 1996; Yang and Rannala, 1997; Mau et al., 1999; Larget and Simon, 1999; Li et al., 2000; Holder and Lewis, 2003; Rannala and Yang, 2003; Lakner et al., 2008; Höhna et al., 2008; Höhna and Drummond, 2012). Second, it makes it easier to add SMC support to existing MCMC-based software libraries. Third, it makes certain benchmark comparison between SMC and MCMC more direct, as we can then choose the set of moves to be the same for both. On the flip side, constructing MCMC proposals is somewhat more constrained, so some of the flexibility provided by the general SMC framework is lost.

Naively, we could pick the SMC proposal directly from an MCMC proposal, $K_r(x_{r-1,k}, x_{r,k}) = q_r(x_{r-1,k}, x_{r,k})$. However, doing so would have the undesirable property that the magnitude of the fluctuation of the weights of the particles from one iteration to the next, $\|W_{r-1} - W_r\|$, does not converge to zero when the annealing parameter change $\phi_r - \phi_{r-1}$ goes to zero. This lack of convergence to zero can potentially cause severe particle degeneracy problems, forcing the use of a number of particles larger than what can be realistically accommodated in memory (although workarounds exist, e.g. Jun and Bouchard-Côté (2014)). To avoid this issue, we follow Del Moral et al. (2006) and use as SMC proposal the accept-reject Metropolis-Hastings transition probability based on $q_r$ (called a Metropolized proposal), reviewed in Algorithm 10.

**Algorithm 10 Accept-reject Metropolis-Hastings algorithm**

1: Propose a new tree and/or new evolutionary parameters, $x_r^* \sim q_r(x_{r-1}, \cdot)$. For example, using a nearest neighbour interchange, and/or a symmetric normal proposal on branch lengths and/or evolutionary parameters.
2: Compute the Metropolis-Hastings ratio based on $\gamma_r$:
   $$
   \alpha_r(x_{r-1}, x_r^*) = \min \left\{ 1, \frac{\gamma_r(x_r^*)q(x_{r-1}, x_r^*)}{\gamma_r(x_r^*)q(x_{r-1}, x_r)} \right\}.
   $$
3: Simulate $u \sim U(0, 1)$.
4: if $u < \alpha_r(x_{r-1}, x_r^*)$ then
5: \hspace{1cm} $x_r = x_r^*$.
6: else
7: \hspace{1cm} $x_r = x_{r-1}$.

> Output the proposal $x_r^*$.
> Output the previous state $x_{r-1}$. 

29
The key point is that a theoretical argument (reviewed in the Appendix 2) shows that provided that (1) $K_r$ has stationary distribution $\pi_r$ (which is true by construction, a consequence of using the Metropolis-Hastings algorithm) and (2) we use the weight formula:

$$w_{r,k} = \frac{\gamma_r}{\gamma_{r-1}}(x_{r-1,k}),$$

then we obtain a valid SMC algorithm, meaning that the key theoretical properties expected from SMC hold under regularity conditions, see Section Theoretical Properties.

In the important special case where $\gamma_r(x_r)$ is equal to the prior times an annealed likelihood, we obtain

$$w_{r,k} = \left[ p(y|x_{r-1,k}) \right]^{\phi_r - \phi_{r-1}}.$$  

As hoped, the update shown in Equation (9) has the property that weight fluctuations vanish as the annealing parameter difference $\phi_r - \phi_{r-1}$ goes to zero. This will form the basis of the annealing parameter sequence adaptation strategies described in the next section. But for now, assume for simplicity that the number of iterations $R$ and the annealing schedule $\phi_r, r \in \{1, \ldots, R\}$ is pre-specified. For example, a simple choice for the annealing parameter sequence (Friel and Pettitt, 2008) is $\phi_r = (r/R)^3$, where $R$ is the total number of SMC iterations. In this case, the difference between successive annealing parameters is $(3r^3 - 3r + 1)/R^3$. An annealed SMC with a larger value of $R$ is computationally more expensive but has a better performance.

In contrast to other SMC algorithms, the annealed SMC algorithm does not require pointwise evaluation of the proposal $K_r(x_{r-1,k}, x_r)$, i.e. given $x_{r-1,k}$ and a sampled $x_r$, we do not need to compute the numerical value of $K_r(x_{r-1,k}, x_r)$ as it does not appear in the weight update formula, Equation (6). This point is important, since for Metropolis-Hastings kernels, pointwise evaluation would require computation of a typically intractable integral under the proposal in order to compute the total probability of rejection. The theoretical justification as to why we do not need pointwise evaluation of $K_r$ is detailed in Appendix 2.

In practice, many proposals are needed to modify different latent variables and to improve mixing. We give in Appendix 3 the list of MCMC proposals we consider. Let $q^i_r, i = 1, \ldots, M$, denote the various proposals, and $K^i_r$ the corresponding Metropolized transition probabilities. We need to combine them into one proposal $K_r$. To ensure that condition (1) above is satisfied, namely that $K_r$ obeys global balance with respect to $\pi_r$, use the following property (Tierney, 1994; Andrieu et al., 2003): if each of the transition kernels $\{K^i\}, i = 1, \ldots, M$, respects global balance with respect to $\pi$, then the \textit{cycle hybrid kernel} $\prod_{i=1}^{M} K^i$ and the \textit{mixture hybrid kernel} $\sum_{i=1}^{M} p_i K^i, \sum_{i=1}^{M} p_i = 1$, also satisfy global balance with respect to $\pi$. The global balance condition, $\int \pi_r(x) K_r(x, x') \, dx = \pi_r(x')$, ensures that the Markov chain encoded by $K_r$ admits $\pi_r$ as a stationary distribution. In practice, the mixture kernel is implemented by randomly selecting $K^i$ with probability $p_i$ at each iteration (Andrieu et al., 2003).

We can now introduce in Algorithm 11 the simplest version of the annealed SMC, which alternates between reweighting, propagating, and resampling. Figure 4.1 presents an overview of
Figure 4.1: An overview of the annealed SMC algorithmic framework for phylogenetic trees. The algorithm iterates the following three steps: (i) compute the weights using samples from the previous iteration, (ii) perform MCMC moves to propose new samples, and (iii) resample from the weighted samples to obtain an unweighted set of samples.

the annealed SMC algorithmic framework. In the proposal step, we propose new particles through MCMC moves (typically Metropolis-Hastings moves). Finally, we use resampling to prune particles with small weights. A list of unweighted particles is obtained after the resampling step.

In the annealed SMC algorithm, note that the weighting and proposal steps can be interchanged. This is different from standard SMC algorithms, where in general the proposal has to be computed before weighting. This interchange is possible because in the annealed SMC algorithm, the weighting function, Equation (6), only depends on particles from the previous iteration and not from those just proposed as in standard SMC algorithms. This flexibility will come handy when designing adaptive schemes.

Before moving on to more advanced versions of the algorithm, we provide first some intuition to motivate the need for resampling. Theoretically, the algorithm produces samples from an artificial distribution with state space $\mathcal{X} \times \mathcal{X} \times \cdots \times \mathcal{X} = \mathcal{X}^R$ (this is described in more detail in Appendix 2). However since we only make use of one copy of $\mathcal{X}$ (corresponding to the particles at the final SMC iteration), we would like to decrease the variance of the state at iteration $R$ (more precisely, of Monte Carlo}
Algorithm 11 The simplest version of annealed SMC algorithm (for pedagogy)

1: **Inputs:**
2: (a) Prior over evolutionary parameters and trees, \( p(x) \), where \( x = (\theta, t) \);
3: (b) Likelihood function \( p(y|x) \);
4: (c) Sequence of annealing parameters \( 0 = \phi_0 < \phi_1 < \cdots < \phi_R = 1 \).
5: **Outputs:** Approximation of the posterior distribution, \( \sum_k \tilde{W}_{r,k} \delta_{\tilde{x}_{r,k}}(\cdot) \approx \pi(\cdot) \).
6: Initialize SMC iteration index: \( r \leftarrow 0 \).
7: Initialize annealing parameter: \( \phi_r \leftarrow 0 \).
8: for \( k \in \{1, 2, \ldots, K\} \) do
9:   Initialize particles \( x_{0,k} \leftarrow (\theta_{0,k}, t_{0,k}) \sim p(\cdot) \).
10: Initialize weights to unity: \( w_{0,k} \leftarrow 1 \).
11: for \( r \in \{1, 2, \ldots, R\} \) do
12:   for \( k \in \{1, 2, \ldots, K\} \) do
13:     Sample particles \( \tilde{x}_{r,k} \sim K_r(x_{r-1,k}, \cdot) \); \( K_r \) is a \( \pi_r \)-invariant Metropolis-Hastings kernel.
14:     Compute unnormalized weights: \( w_{r,k} = [p(y|x_{r-1,k})]^{\phi_r - \phi_{r-1}} \).
15:     if \( r < R \) then
16:       for \( k \in \{1, 2, \ldots, K\} \) do
17:         Resample the particles: \( x_{r,k} \sim \sum_k \tilde{W}_{r,k} \delta_{\tilde{x}_{r,k}}(\cdot) \).
18:     else
19:       No resampling needed at the last iteration.
20:   Return the particle population \( \tilde{x}_{r}, \tilde{W}_{r} \).

Carlo estimators of functions of the state at iteration \( R \)). This is what resampling for iteration \( r < R \) achieves, at the cost of increasing the variance for the auxiliary part of the state space \( r < R \). From this argument, it follows that resampling at the last iteration should be avoided.

When resampling is performed at every iteration but the last, an estimate of the marginal likelihood, \( p(y) \), is given by the product of the average unnormalized weights, namely:

\[
\hat{Z}_K := \prod_{r=1}^{R} \frac{1}{K} \sum_{k=1}^{K} w_{r,k}.
\]  

4.5 Adaptive mechanisms for annealed SMC

We discuss how two adaptive schemes from the SMC literature can be applied in our Bayesian phylogenetic inference setup to improve the scalability and usability of the algorithm described in the previous section. The first scheme relaxes the assumption that resampling is performed at every step, and the second is a method for automatic construction of the annealing parameter sequence. The two mechanisms go hand in hand and we recommend using both simultaneously. The combination yields Algorithm 12 which we explain in detail in the next two subsections.

The two adaptive mechanisms make theoretical analysis considerably more difficult. This is a common situation in the SMC literature. A common work-around used in the SMC literature is to run the algorithm twice, a first time to adaptively determine the resampling and annealing schedules,
and then a second independent time using the schedule fixed in the first pass. We call it debiased adaptive annealed SMC.

Algorithm 12 An adaptive annealed SMC algorithm

1: **Inputs:** (a) Prior over evolutionary parameters and trees \( p(x) \), where \( x = (\theta, t) \); (b) Likelihood function \( p(y|x) \).
2: **Outputs:** (a) Approximation \( Z \) of the marginal data likelihood, \( Z \approx p(y) = \int p(dx)p(y|x) \); (b) Approximation of the posterior distribution, \( \sum_k \tilde{W}_{r,k} \delta_{\tilde{x}_{r,k}}(\cdot) \approx \pi(\cdot) \).
3: Initialize SMC iteration index: \( r \leftarrow 0 \).
4: Initialize annealing parameter: \( \phi_r \leftarrow 0 \).
5: Initialize marginal likelihood estimate: \( Z \leftarrow 1 \).
6: for \( k \in \{1, 2, \ldots, K\} \) do
7: Initialize particles with independent samples: \( x_{0,k} \leftarrow (\theta_{0,k}, t_{0,k}) \sim p(\cdot) \).
8: Initialize weights to unity: \( w_{0,k} \leftarrow 1 \).
9: for \( r \in \{1, 2, \ldots\} \) do
10: Determine next annealing parameter: \( \phi_r = \text{NextAnnealingParameter}(x_{r-1}, \cdot, w_{r-1}, \phi_{r-1}) \).
11: for \( k \in \{1, \ldots, K\} \) do
12: Compute pre-resampling unnormalized weights: \( \tilde{w}_{r,k} = w_{r-1,k} [p(y|x_{r-1,k})]^{\phi_r - \phi_{r-1}} \).
13: Sample particles \( \tilde{x}_{r,k} \sim K_r(x_{r-1,k}, \cdot) \); \( K_r \) is a \( \pi_r \)-invariant Metropolis-Hastings kernel.
14: if \( \phi_r = 1 \) then
15: update \( Z \leftarrow (Z/K) \cdot \sum_k \tilde{w}_{r,k} \), then return updated \( Z \) and particle population \( \tilde{x}_{r, \cdot}, \tilde{W}_{r, \cdot} \).
16: else
17: if particle degeneracy is too severe, i.e. \( \text{rESS}(\tilde{W}_{r, \cdot}) < \epsilon \) then
18: Update marginal likelihood estimate, \( Z \leftarrow (Z/K) \cdot \sum_k \tilde{w}_{r,k} \).
19: Resample the particles.
20: for \( k \in \{1, \ldots, K\} \) do
21: Reset particle weights: \( w_{r,k} = 1 \).
22: else
23: for \( k \in \{1, \ldots, K\} \) do
24: \( w_{r,k} = \tilde{w}_{r,k} \); \( x_{r,k} = \tilde{x}_{r,k} \). » No resampling is needed.

4.5.1 Measuring Particle Degeneracy using Relative (Conditional) Effective Sample Size (ESS)

Both adaptive methods rely on being able to assess the quality of a particle approximation. For completeness, we provide more background in Appendix 5 on the notions of Effective Sample Size (ESS) and conditional ESS (CESS), a recent generalization which we use here (Zhou et al., 2016). The notion of ESS in the context of importance sampling (IS) or SMC is distinct from the notion of ESS in the context of MCMC. The two are related in the sense of expressing a variance inflation compared to an idealized Monte Carlo scheme but they differ in the details. We will assume from now on that ESS refers to the SMC context.

We will use a slight variation of the definition of ESS and CESS where the measures obtained are normalized to be between zero and one. Some tuning parameters of the adaptive algorithms are easier to express in this fashion. We use the terminology relative (conditional) ESS to avoid
confusion. Motivated by the analysis of the error of Monte Carlo estimators, the key measure of particle degeneracy needed in the following is the relative conditional effective sample size:

$$rCESS(W, u) = \left( \frac{\sum_{k=1}^{K} W_k u_k}{\sum_{k=1}^{K} W_k u_k^2} \right)^{2/2}$$

(11)

where $W = (W_1, W_2, \ldots, W_K)$ is a vector of weights of a set of reference weighted particles being updated using a vector of non-negative values $u = (u_1, u_2, \ldots, u_K)$. What $W$ and $u$ specifically represent will be explained in the next subsection.

Having a high rCESS value is a necessary but not sufficient condition for a good SMC approximation. If it is low during some of the intermediate SMC iterations, then the ESS at the final iteration may not be representative of the true posterior approximation quality.

### 4.5.2 Dynamic Resampling

As explained in Section Basic Annealed SMC Algorithm 4.4.2, the construction of the proposal guarantees that as the difference $\phi_r - \phi_{r-1}$ goes to zero, the fluctuation of the weights vanishes. In this context (of having small weight updates), resampling at every iteration is wasteful. Fortunately, SMC algorithms can be modified to forgo a subset of the resampling steps. From a theoretical standpoint, this is achieved by “grouping” the SMC proposals when they are not separated by a resampling round (and grouping similarly the intermediate distributions $\gamma_r$). For example, to resample every other round, we use a transformed SMC algorithm with proposal $K'_{r/2}(x_r, (x_{r+1}, x_{r+2})) = K_{r+1}(x_r, x_{r+1})K_{r+2}(x_{r+1}, x_{r+2})$, for each even $r$. For convenience, this can be implemented as an algorithm over $R$ iterations instead of $R/2$, with two modifications: first, when resampling is skipped, we multiply the weights; otherwise, we reset the weights to one after resampling. This is implemented in Lines 10 and 18 of Algorithm 12. Second, we only use the weights corresponding to resampling rounds in the estimate of the marginal likelihood (Equation (10)). This is implemented in Lines 13 and 18 of Algorithm 12.

Instead of specifying in advance the subset of iterations in which resampling should be performed, it is customary in the SMC literature to determine whether to resample in an adaptive fashion (Doucet and Johansen, 2009). To do so, the standard approach is to compute a measure of particle degeneracy at every iteration, and to perform resampling only when the particle degeneracy exceeds a pre-determined threshold. In Appendix 4, we empirically compare the performance of adaptive annealed SMC algorithm with different resampling thresholds. All our numerical experiments use the multinomial resampling method, but we recommend more advanced schemes such as stratified resampling (Douc and Cappé, 2005).

The standard measure of particle degeneracy used for this purpose is called the relative ESS, defined as:
\[ r\text{ESS}(\tilde{W}_r) = \left( K \sum_{k=1}^{K} \tilde{W}_{r,k}^2 \right)^{-1}. \] (12)

The above formula can be shown to be a special case of rCESS, Equation (11), as follows. Let \( r^* \) denote the iteration of the latest resampling round preceding the current iteration \( r \). This implies \( W_{r^*,k} = 1/K \) for all \( k \). Plugging in the weight update \( u_k = \tilde{w}_{r,k} \) into Equation (11), we obtain

\[
r\text{CESS}(W_{r^*}, \tilde{w}_r) = \left( \sum_{k=1}^{K} \frac{1}{K} \tilde{W}_{r,k} \right)^2 / \sum_{k=1}^{K} \frac{1}{K} \tilde{W}_{r,k}^2 \]
\[
= \frac{1}{K} \left( \sum_{k=1}^{K} \tilde{w}_{r,k} \right)^2 / \sum_{k=1}^{K} \tilde{w}_{r,k}^2
\]
\[
= \left( K \sum_{k=1}^{K} \tilde{W}_{r,k}^2 \right)^{-1}.
\]

### 4.5.3 Adaptive Determination of Annealing Parameters

Our sequence of intermediate artificial distributions \( \pi_r \) as defined in Equation (10) is determined by the choice of the annealing schedule, \( \{\phi_r\} \), or equivalently, by choosing the successive differences \( \phi_r - \phi_{r-1} \). Ideally, the sequence of intermediate distributions changes gradually from the prior distribution (\( \phi_0 = 0 \)) to the posterior distribution (\( \phi_R = 1 \)) so that the propagated particles from the current iteration can well approximate the next intermediate distribution.

In practice constructing such a sequence \( \{\phi_r\}_{r=1,...,R} \) is difficult and inconvenient. Not only the number of distributions \( R \) to get a certain accuracy may depend on the number of taxa, the number of sites, and the complexity of the evolutionary model, but also the optimal spacing between consecutive annealing parameters is in general non-regular. To alleviate this, in the following we borrow an adaptive strategy from the Approximate Bayesian Computation literature (Del Moral et al., 2012), also generalized to Bayesian model selection in Zhou et al. (2016).

The adaptive annealing scheme is based on two observations. First, our discrete set of intermediate distributions \( \pi_1, \pi_2, \ldots, \pi_R \) are actually continuously embedded into a continuum of distributions indexed by \( \phi \in [0, 1] \). Second, in the SMC algorithm presented in Algorithm 11, the weight update, Line 14, depends only on \( x_{r-1} \) (whereas in general SMC algorithms, the weight update could depend on both \( x_{r-1} \) and \( x_r \); here it does not because of cancellation explained in Appendix 2). The consequence of the lack of dependence on \( x_r \) is that we can swap the order of proposal (Line 13) and particle weighting (Line 14) in Algorithm 11. So instead of computing the weights only for one pre-determined annealing parameter \( \phi_r \), we can search over several tentative values. For each tentative value, we can score the choice using a measure of weight degeneracy applied to the putative weights. Crucially, each choice can be quickly scored without having to propose particles, which is key since proposals are typically the computational bottleneck: in a phylogenetic context, the cost of one proposal step scales linearly in the number of sites whereas the search over \( \phi_r \) proposed in
this section has a running time constant in the number of sites and taxa. This is because the search involves fixed values of $p(y|x_{r-1,k})$ cached from the last proposal step, which are exponentiated to different values.

Based on these observations, we select an annealing parameter $\phi$ such that we achieve a controlled increase in particle degeneracy, namely such that

$$g(\phi) = \alpha g(\phi_{r-1}),$$

where the function $g : [\phi_{r-1}, \infty) \rightarrow [0, 1]$ is defined as

$$g(\phi) = \text{rCESS} \left( \int_{\phi_{r-1}}^{1} p(y|x_{r-1,k})^{\phi - \phi_{r-1}} \right),$$

and $\alpha \in (0, 1)$ is a tuning parameter, which in practice is close to 1. By construction, $g(\phi_{r-1}) = 1$, so Equation (13) is equivalent to $g(\phi) = \alpha$.

More precisely, since we want $\phi \in [0, 1]$, the annealing parameter adaptation procedure, NextAnnealingParameter (Algorithm 13), is designed to return $\phi_r = 1$ if $g(1) \geq \alpha$. Otherwise, because there is no closed-form solution for $\phi$ in Equation (13), we use bisection to solve this one-dimensional search problem in the interval $\phi \in (\phi_{r-1}, 1)$ (Line 7 of Algorithm 13).

We now argue that the search problem in Line 7 of Algorithm 13 always has a solution. Indeed, $g$ is a continuous function with, on the left end of the search interval, $g(\phi_{r-1}) = 1$, and on the right end, $g(1) < \alpha$ (otherwise the algorithm sets $\phi_r = 1$ in Line 5). It follows that there must indeed be an intermediate point $\phi^*$ with $g(\phi^*) = \alpha$. Note that continuity and the identification of the left end point of the interval is possible thanks to the form of our weight update in Equation (9), hence justifying the earlier informal argument about the need to have the fluctuation of the weights disappearing as $\phi_r - \phi_{r-1}$ goes to zero.

As in the previous section on dynamic resampling, NextAnnealingParameter is again based on relative conditional ESS, but this time, we are interested in the degeneracy of a single iteration, i.e. we do not trace back until the previous resampling step (since the optimization over the annealing schedule can only impact the current iteration). As a corollary, the previous iteration’s particles are not always equally weighted, hence the simplification in Equation (22) is not possible here and we use the full formula for relative conditional ESS.

The parameter $\alpha$ used in Algorithm 13 encodes the decay in particle population quality that we are aiming for. Based on our experiments we recommend values very close to one. For this reason, we reparameterize the parameter $\alpha$ into $\alpha = 1 - 10^{-\beta}$ and recommend a default value of $\beta = 5$ as a reasonable starting point. Increasing $\beta$ improves the approximation accuracy.

### 4.5.4 Computational Complexity

The computational complexity of annealed SMC is linear in both the number of intermediate distributions $R$ and the number of particles $K$. Naively, the resampling step scales like $O(K^2)$,
Algorithm 13 Procedure NextAnnealingParameter

1: **Inputs:** (a) Particle population from previous SMC iteration \( (x_{r-1}, w_{r-1}) \); (b) Annealing parameter \( \phi_{r-1} \) of previous SMC iteration; (c) A degeneracy decay target \( \alpha \in (0, 1) \).
2: **Outputs:** automatic choice of annealing parameter \( \phi_r \).
3: Initialize the function \( g \) assessing the particle population quality associated to a putative annealing parameter \( \phi \):
\[
g(\phi) = \text{rCESS} \left( w_{r-1}, \cdot, p(y|x_{r-1}, \cdot)^{\phi_{r-1} - \phi} \right) = \frac{\left( \sum_{k=1}^{K} w_{r-1,k} p(y|x_{r-1,k})^{\phi_{r-1} - \phi} \right)^2}{\sum_{k=1}^{K} w_{r-1,k} p(y|x_{r-1,k})^{2(\phi_{r-1} - \phi)}}.
\]
4: if \( g(1) \geq \alpha \) then
5: return \( \phi_r = 1 \).
6: else
7: return \( \phi_r = \phi^* \in (\phi_{r-1}, 1) \) such that \( g(\phi^*) = \alpha \) via bisection.

but a linear time multinomial resampling algorithm is obtained by generating order statistics via normalization of a Poisson process (Devroye, 1986, Section 2.1, p.214). This technique is well known in the SMC literature (Doucet and Johansen, 2009). Alternatively, one can use stratified or systematic resampling (Doucet and Johansen, 2009), which provides a simple to implement linear time resampling algorithm.

The memory consumption of annealed SMC is linear in \( K \) and constant in \( R \).

### 4.6 Review of other marginal likelihood estimation methods

For completeness, we review here some alternatives to Equation (10) for estimating marginal likelihoods, which we will compare to SMC from both a theoretical and empirical stand-point.

#### 4.6.1 Stepping Stone

The Stepping Stone algorithm (Xie et al., 2010) is a method for marginal likelihood estimation. It is widely used via its MrBayes implementation (Huelsenbeck and Ronquist, 2001). As with SMC, the Stepping Stone method introduces a list of annealed posterior distributions connecting the posterior distribution and the prior distribution. We use a notation analogous to SMC, with \( \{\pi_d\}_{d=0,1,\ldots,D} \) denoting the intermediate distributions, \( \pi_d(x) \propto \gamma_d(x) = p(y|x)^{\phi_d} \pi(x) \), \( 0 = \phi_0 < \phi_1 < \phi_2 < \cdots < \phi_D = 1 \). The marginal likelihood \( Z \) can be written as
\[
Z = Z_D = Z_0 \prod_{d=1}^{D} \frac{Z_d}{Z_{d-1}}.
\]
We can rewrite the ratio of \( Z_d \) and \( Z_{d-1} \) as
\[
\frac{Z_d}{Z_{d-1}} = \int \frac{\gamma_d(x)}{\gamma_{d-1}(x)} \pi_{d-1}(x) dx.
\]
The Stepping Stone method prescribes running several MCMC chains targeting \(\pi_{d-1}(x)\) to obtain \(N\) posterior samples \(x_{d-1,1}, x_{d-1,2}, \ldots, x_{d-1,N}\), then

\[
\frac{Z_d}{Z_{d-1}} = \frac{1}{N} \sum_{i=1}^{N} \left( p(y|x_{d-1,i}) \right)^{\phi_{d-}\phi_{d-1}}.
\]

The estimator of the marginal likelihood admits the form

\[
\hat{Z}_D = \prod_{d=1}^{D} \frac{1}{N} \sum_{i=1}^{N} \left( p(y|x_{d-1,i}) \right)^{\phi_{d-}\phi_{d-1}}.
\]

The number of intermediate distributions is a trade-off between computing cost and accuracy. A larger number of MCMC chains can provide a better approximation for the marginal likelihood, but the computational cost will be higher. To make fair comparison between the marginal likelihood estimators provided by the annealed SMC and Stepping Stone, we set \(K_{SMC}R_{SMC} = N_{SS}D_{SS}\).

Another factor that will impact the Stepping Stone estimator is the choice of annealing parameter sequence \(\{\phi_d\}_{d=1,2,\ldots,D}\). In this paper, we use the annealing scheme \(\phi_d = (d/D)^{1/\alpha}\) recommended by Xie et al. (2010), where \(\alpha\) is between 0.2 and 0.4.

### 4.6.2 Linked Importance Sampling

Stepping stone uses importance sampling to approximate the ratio of marginal likelihoods for two intermediate distributions. However, the importance sampling approximation would be poor if the two successive distributions do not have enough overlaps. Linked Importance Sampling (Neal, 2005) improves the performance of importance sampling by introducing bridge distributions, e.g., “geometric” bridge: \(\gamma_{d-1,d}(x) = \sqrt{\gamma_{d-1}(x)\gamma_d(x)}\). More importantly, Linked Importance Sampling provides an unbiased marginal likelihood estimator. The ratio of two marginal likelihoods can be written as

\[
\frac{Z_d}{Z_{d-1}} = \frac{Z_{d-1,d}}{Z_{d-1}} \frac{Z_{d-1,d}}{Z_d} = \left\{ \int \gamma_{d-1,d}(x) \pi_{d-1}(x) dx \right\} \left\{ \int \gamma_{d-1,d}(x) \pi_d(x) dx \right\}.
\]

For \(d = 1, \ldots, D\), to estimate the ratio \(Z_d/Z_{d-1}\), we first run MCMC targeting \(\pi_{d-1}(x)\) to obtain \(N\) posterior samples \(x_{d-1,1}, x_{d-1,2}, \ldots, x_{d-1,N}\) (when \(d = 1\), we sample from the prior distribution). Then we sample the initial state of \(\pi_d\). Two successive MCMC chains \(\pi_{d-1}(x)\) and \(\pi_d(x)\) are linked by a state \(x_{d-1,\mu_{d-1}}\) where index \(\mu_{d-1}\) is sampled from \(\{1, 2, \ldots, N\}\) according to the following probabilities:

\[
p(\mu_{d-1}|x_{d-1,1:N}) = \frac{\gamma_{d-2,d-1}(x_{d-1,\mu_{d-1}})}{\gamma_{d-1}(x_{d-1,\mu_{d-1}})} \frac{1}{\sum_{i=1}^{N} \gamma_{d-2,d-1}(x_{d-1,i})}.
\]

In case \(d = 1\), the linked state \(\mu_0\) is uniformly sampled from the \(N\) samples of \(\pi_0(x)\). Finally, we run MCMC chain \(\pi_d(x)\) starting from initial state \(x_{d-1,\mu_{d-1}}\) to obtain \(N\) posterior samples.
The ratio of two marginal likelihoods can be approximated by
\[
\frac{\hat{Z}_d}{Z_{d-1}} = \frac{\hat{Z}_{d-1}}{Z_{d-1}} \times \frac{1}{1 - \frac{1}{N} \sum_{i=1}^{N} \gamma_{d-1}(x_{d-1,i})}.
\]
In this paper, we use the “geometric” bridge. Hence, the estimator of ratio can be simplified to
\[
\frac{\hat{Z}_d}{Z_{d-1}} = \left\{ \sum_{i=1}^{N} \left[ p(y|x_{d-1,i})^{\frac{d_d - d_{d-1}}{2}} \right] \right\} / \left\{ \sum_{i=1}^{N} p(y|x_{d,i})^{d_d - d_{d-1} - \frac{1}{2}} \right\}.
\]
We refer to Appendix 6 for more background on the Linked Importance Sampling algorithm.

### 4.7 Theoretical properties

In this section, we review three theoretical properties of interest, consistency, marginal likelihood estimate unbiasedness, and asymptotic normality, with an emphasis on their respective practical importance.

#### 4.7.1 Properties of Annealed Sequential Monte Carlo

In the context of SMC algorithms, the first property, consistency means that as the number of particles is increased, the approximation of posterior expectations can become arbitrarily close to the true posterior expectation. This makes the approximation in Equation (7) more precise:
\[
\sum_{k=1}^{K} W_{r,k} f(x_{r,k}) \to \int \pi_r(x) f(x) \, dx \text{ as } K \to \infty,
\]
provided \( f \) satisfies regularity conditions, for example \( f \) is bounded, and where convergence of the random variables holds for a set of random seeds having probability one. See for example Wang et al. (2015).

Consistency can be viewed as the “bare minimum” expected from modern SMC algorithms. A more informative class of results consists in central limit theorem equivalents of Equation (16). These results can be used to assess the total variance of Monte Carlo estimators (whereas measures such as effective sample size described previously are local in nature), see Chan and Lai (2013). However, since numerically stable versions of these methods are still at their infancy (Olsson and Douc, 2017), we will focus the remaining on the third property, unbiasedness.

We say an estimator \( \hat{Z} \) for a constant \( Z \) is unbiased if \( E[\hat{Z}] = Z \). Here the expectation is defined with respect to the randomness of the approximation algorithm. This contrasts with the classical statistical definition of unbiasedness in which the randomness comes from the data generation process.
For SMC algorithms, unbiasedness holds in a more restrictive sense compared to consistency. In general:

\[
\mathbb{E}\left[ \sum_{k=1}^{K} W_{r,k} f(x_{r,k}) \right] \neq \int \pi_r(x)f(x) \, dx,
\]

in other words, repeatedly running SMC with a fixed number of particles but different random seeds and averaging the results does not provide arbitrarily precise approximations (the same negative result holds with MCMC). However, if we restrict our attention to marginal likelihood estimates, remarkably the unbiasedness property does hold (Del Moral et al., 2006), i.e. for any finite \( K \), \( \hat{Z}_K \) as defined in Equation (10) is such that:

\[
\mathbb{E}[\hat{Z}_K] = Z = \int \gamma_r(x) \, dx.
\]

More details on the unbiasedness of the marginal likelihood SMC estimator and other theoretical properties of annealed SMC can be found in Appendix 2, subsection Unbiasedness, Consistency and Central Limit Theorem.

While the notion of unbiasedness has been central to frequentist statistics since its inception, only in the past decade has it started to emerge as a property of central importance in the context of (computational) Bayesian statistics. Traditionally, the main theoretical properties analyzed for a given Monte Carlo method \( \hat{Z} \) estimating \( Z \) was consistency.

With the emergence of pseudo-marginal methods, the bias of Monte Carlo methods is now under closer scrutiny. Pseudo-marginal methods are MCMC methods which replace probability factors in the Metropolis-Hastings ratio by positive unbiased estimators of these probabilities. For example, Andrieu et al. (2010) provide examples where global parameters of state-space models are sampled using an MCMC algorithm where the probability of the data given the global parameters and marginally over the latent states is estimated using an SMC algorithm. We refer the reader to Andrieu and Roberts (2009) for more examples where unbiasedness is used to compose MCMC algorithms in order to attack inference in complex models. In the context of phylogenetic inference, this is useful for Bayesian analysis of intractable evolutionary models, see for example Hajiaghayi et al. (2014).

Another area where unbiasedness can play a role is for checking correctness of Monte Carlo procedures. In contrast to correctness checks based on consistency such as Geweke (2004), which are asymptotic in nature and hence necessarily have false positive rates (i.e. cases where the test indicates the presence of a bug when in fact the code is correct), checks based on unbiasedness can achieve a false positive rate of zero, using the strategy described in the next section.

### 4.7.2 Using Unbiasedness to Test Implementation Correctness

Typically, the algorithm shown in Algorithm 11 is implemented in a model-agnostic fashion. Hence it is reasonable to assume that we can construct test cases on discrete state spaces. For example, one can use phylogenetic trees with fixed branch lengths, or even simpler models such as hidden
Markov models (HMMs). Furthermore, we conjecture that many software defects can be detected in relatively small examples, where exhaustive enumeration is possible, and hence $Z$ can be computed exactly. We can determine sufficient complexity of the examples to use via code coverage tools (Miller and Maloney, 1963).

We would like to test if equality of Equation (18) holds for a given implementation. The right hand side can be computed easily since we assume the example considered is small. To compute analytically the expectation on the left-hand side, we use a method borrowing ideas from probabilistic programming (Wingate et al., 2011), and use an algorithm, called ExhaustiveRandom that automatically visits all possible execution traces $\tau_i$ of a given randomized algorithm. The execution trace of a randomized algorithm refers to a realization of all random choices in the algorithm (in the context of SMC, both the resampling steps and the proposal steps). ExhaustiveRandom enumerates all the execution traces while also computing the respective probability $p_i$ of each trace. This is done by performing a depth first traversal of the decision tree corresponding to the randomized algorithm being tested. The number of execution traces grows exponentially fast but this is still a useful tool as very small examples are generally sufficient to reach code coverage.

For each execution trace $\tau_i$, we can also obtain the normalization estimate $\hat{z}_i$ corresponding to that trace, and hence get the value of the left-hand side of Equation (18) as $\sum_i p_i \hat{z}_i$. We used this check via an open source implementation of ExhaustiveRandom (https://github.com/alexandrebouchard/bayonet/blob/1b9772e91cf2fb14a91f2e5e282fcf4ded61ee22/src/main/java/bayonet/distributions/ExhaustiveDebugRandom.java) to ensure that our software satisfies the unbiasedness property. See the numerical simulation section for details.

### 4.7.3 Properties of the Stepping Stone Method

For the stepping stone method, the expected value of Equation (15) depends on the nature of the samples $x_{d-1,1}, x_{d-1,2}, \ldots, x_{d-1,N}$. If they are independent, the procedure is unbiased. However, if the samples are obtained from a Markov chain, there are no guarantees that the procedure is unbiased unless the MCMC chain is initialized at the exact stationary distribution. In practice, this is not possible: Xie et al. (2010) use a burned-in MCMC chain, which implies that the chain is asymptotically unbiased, however for any finite number of iterations, a bias remains. Unfortunately, the two main motivations for unbiasedness (pseudo-marginal methods and the correctness checks described earlier) both require unbiasedness to hold for any finite number of Monte Carlo samples; asymptotic unbiasedness is not sufficient.

We show in the numerical simulation section an explicit counterexample where we compute the non-zero bias of the stepping stone method. This motivates the need for implementable unbiased methods, such as the annealed Sequential Monte Carlo method described in this work.
4.7.4 Comparison of Unbiased Marginal Likelihood Estimators

In Bayesian phylogenetics, a marginal likelihood estimate $\hat{Z}$ is generally a very small number. Instead of computing $\hat{Z}$ directly, we compute the logarithm of the marginal likelihood estimate, $\log(\hat{Z})$. For SMC and Linked Importance Sampling, although $\hat{Z}$ is an unbiased estimator, taking the logarithm of $\hat{Z}$ introduces bias. Jensen’s inequality shows that $\log(\hat{Z})$ is a biased estimator of $\log(Z)$, and is generally underestimated,

$$\mathbb{E}[\log(\hat{Z})] \leq \log(\mathbb{E}(\hat{Z})) = \log Z.$$  

This provides a tool to compare the performance of an unbiased normalization constant estimation method $m_1$ to another estimation method $m_2$. Suppose we run each method $M$ times with different seeds and a fixed computational budget. Let $L_i = \sum_{j=1}^{M} \log \hat{Z}_{i,j} / M$ denote the average estimate of the log marginal likelihood for the $i$-th method, where $\hat{Z}_{i,j}$ is the estimate with the $j$-th random seed using the $i$-th method. If $m_2$ is also unbiased then for $M$ large enough, both $m_1$ and $m_2$ underestimate $\log \mathbb{E}[Z]$, and the largest $L_i$ is closest to $\log \mathbb{E}[Z]$, which determines the best performing method. If $m_2$ is not unbiased, then if $L_1 > L_2$ and $M$ is large enough, we can conclude that $m_1$ is superior (but we cannot confidently order the methods if $L_2 > L_1$).

However the Monte Carlo counterparts of the orderings should be considered with a pinch of salt since the number of replicates $M$ needed may be intractable in some cases.

4.8 Simulation studies

4.8.1 Simulation Setup and Tree Distance

In order to simulate datasets, we first generated a set of random unrooted trees, including topology and branch lengths, as the reference trees. The tree topology was sampled from a uniform distribution. Each branch length was generated from an exponential distribution with rate 10.0.

Then, for each reference tree, we simulated DNA sequences using the K2P model with parameter $\kappa = 2.0$ (Kimura, 1980). While the main focus of this work is on marginal likelihood estimation, we also performed some benchmarking on the quality of the inferred trees. To do so, we used the majority-rule consensus tree (Felsenstein, 1981) to summarize the weighted phylogenetic tree samples obtained from annealed SMC. We measured the distance between each estimated consensus tree to its associated reference tree using three types of distance metrics: the Robinson-Foulds (RF) metric based on sums of differences in branch lengths (Robinson and Foulds, 1979), the Kuhner-Felsenstein (KF) metric (Kuhner and Felsenstein, 1994), and the partition metric (PM), also known as symmetric difference or topology only RF metric (Robinson and Foulds, 1981).

4.8.2 Hidden Markov Models

As discussed when we introduced the unbiasedness correctness test, it is useful to perform some preliminary experiments on finite state models. We used a hidden Markov model (HMM)
with a finite latent state (see the graphical representations of a hidden Markov model and hidden state transitions in Figure 4.2). The variables $X_t$ shown in the figure are unobserved and take on discrete values with a distribution depending on the previous variable $X_{t-1}$. For each unobserved variable, we define an observed variable $Y_t$, also discrete, with a conditional distribution depending on $X_t$. The latent state space in our experiment was set to $\{0, 1, 2, 3, 4\}$ and latent transitions were set uniformly across neighbour integers. The emissions we used take two possible values with conditional probabilities given by $(0.2, 0.8), (0.1, 0.9), (0.01, 0.99), (0.2, 0.8)$ and $(0.3, 0.7)$. The proposals were based on the Gibbs sampler on a single variable. The posterior distribution of interest is over the latent variables $X_1, X_2, \ldots$ given the observations $Y_1, Y_2, \ldots$. Of course, such a model would not normally be approached using approximate inference methods. Moreover, notice that this is a non-standard way of using SMC for a sequential model where we do not make use of the sequential structure of the model.

![Graphical representation of a hidden Markov model](image)

**Figure 4.2:** Graphical representation of (a) a hidden Markov model; (b) transitions between hidden states.

We first performed unbiasedness correctness tests on a chain of length two based on three equally spaced annealing parameters $(0, 1/2, 1)$, and observations $(0, 1)$. We first computed the true marginal likelihood, $0.345$. Using the method described in the Theoretical Properties Section, we computed the exact value of $\mathbb{E}[Z]$ by exhaustive enumeration of all execution traces for SMC and the Stepping Stone method. For SMC with two particles, the ExhaustiveRandom algorithm enumerated 1,992,084 traces resulting in an expectation of $0.34499999999999525$. For Stepping Stone with two MCMC iterations per annealing parameter, the ExhaustiveRandom algorithm enumerated 1,156,288 traces resulting in an expectation of $0.33299145257312235$. This supports that SMC is unbiased and provides an explicit counterexample of the bias of the stepping stone method.

Second, we ran experiments on larger versions of the same model, a chain of length 32, as well as with more annealing steps and particles per step. In this regime it is no longer possible to enumerate all the execution traces so we averaged over 100 realizations of each algorithm instead. The true marginal likelihood can still be computed using a forward-backward algorithm. We show the results in Figure 4.3.
4.8.3 Comparison of Marginal Likelihood Estimates

In this section, we benchmark the marginal likelihood estimates provided by adaptive annealed SMC (ASMC), debiased adaptive annealed SMC (DASMC), deterministic annealed SMC (DSMC), Linked Importance Sampling (LIS) and Stepping Stone (SS). In DASMC, the annealing scheme was determined before running annealed SMC using the same annealing parameters obtained from the ASMC. In DSMC, we used the annealing scheme $\phi_r = (r/R)^3$ with a predetermined $R$.

In the first experiment, we focus on evaluating the marginal likelihood estimates using ASMC, DASMC, LIS and SS with the same computing budget. We simulated unrooted trees of varying sizes (numbers of taxa): 5, 10, 15, 20, and 25. For each tree, we generated one data set of DNA sequences. Sequence length was set to 100. The execution of each algorithm and setting was repeated 100 times with different random seeds. We used $\beta = 5$ for adaptive annealed SMC, and the number of particles was set to 1000. In stepping stone and linked importance sampling, we set the total number of heated chains $D$ to 50, and the annealing scheme was set to $\phi_d = (d/D)^3$, where $d = 1, 2, \ldots, D$. We enforced $K_{SMC}R_{SMC} = N_{SS}D_{SS} = N_{LIS}D_{LIS}$ in order to make the comparisons fair. Information about $R_{SMC}$ are shown in Table 4.1.

<table>
<thead>
<tr>
<th>#taxa</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{SMC}$</td>
<td>1932</td>
<td>3741</td>
<td>5142</td>
<td>6047</td>
<td>7219</td>
</tr>
</tbody>
</table>

Figure 4.4 shows the comparison of the performance of the four algorithms in terms of the marginal likelihood in log scale as the number of taxa increases. As described in the theoretical analysis section, for the unbiased estimators, we have asymptotically that the log of the marginal likelihood should underestimate the marginal likelihood by Jensen’s inequality. The results support that ASMC and DASMC can achieve more accurate marginalized likelihood estimates compared to SS and LIS with the same computational cost. The performances of the two SMC algorithms are
quite similar, while the marginal likelihood estimates provided by LIS and SS are close to each other. In Appendix 9, we describe an experiment comparing ASMC, DASMC, LIS and SS with a very large value of $K$. The mean of log marginal likelihood for the four methods are close (reduction of the gap is expected, since all methods are consistent), while ASMC and DASMC still exhibit smaller variance across seeds compared to LIS and SS.

<table>
<thead>
<tr>
<th>Method</th>
<th>ASMC</th>
<th>DASMC</th>
<th>LIS</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$-402.6$</td>
<td>$-402.5$</td>
<td>$-402.4$</td>
<td>$-402.4$</td>
</tr>
<tr>
<td>10</td>
<td>$-772.50$</td>
<td>$-772.25$</td>
<td>$-772.00$</td>
<td>$-771.75$</td>
</tr>
<tr>
<td>15</td>
<td>$-1256.0$</td>
<td>$-1255.5$</td>
<td>$-1255.0$</td>
<td>$-1255.0$</td>
</tr>
<tr>
<td>20</td>
<td>$-1269.0$</td>
<td>$-1268.5$</td>
<td>$-1268.0$</td>
<td>$-1267.5$</td>
</tr>
<tr>
<td>25</td>
<td>$-1912$</td>
<td>$-1911$</td>
<td>$-1910$</td>
<td>$-1910$</td>
</tr>
</tbody>
</table>

Figure 4.4: Marginal likelihood (in log scale) estimates for different numbers of taxa with a fixed computational budget.

Another experiment was conducted to measure the variability of the marginal likelihood estimates from each algorithm, by comparing the coefficients of variation (CV) for different numbers of taxa with the same setting. The coefficient of variation is defined as $CV = sd(\hat{Z})/\bar{E}(\hat{Z})$. We simulated 70 trees, increasing the number of taxa (from 10, 15, 20, 25, 30, 35, 40; 10 trees of each size), and created 10 data sets for each tree. For each data set, we repeated each algorithm 10 times with different random seeds. The upper bound of CV equals $\sqrt{n-1}$, where $n$ represents the number of repeats with different random seeds in experiments. We refer to Appendix 7 for the derivation of the upper bound of CV. In our setting, this upper bound is $\sqrt{110-1} = 3$. In ASMC, the computational cost was fixed at $K = 1000$ and $\beta = 5$. In DSMC, we used the same number of particles, and the annealing scheme was set to $\phi_r = (r/R)^3$, where the total number of annealing parameters $R$ was fixed to be the one obtained from running ASMC with $K = 1000$ and $\beta = 5$ for a tree with 10 taxa.

Figure 4.5 displays the CV for ASMC and DSMC as a function of the number of taxa. The error bars in the figure represent 95% confidence intervals. The CV of DSMC increases faster than ASMC as the number of taxa gets larger than 15. It gradually converges to the upper bound of CV as the number of taxa reaches 35. The CV of ASMC increases more slowly as the number of taxa increases.

4.8.4 Comparison of Model Selection by Annealed SMC versus Stepping Stone

In this section, we compare the performance of ASMC and Stepping Stone on a Bayesian model selection task. We simulated 20 unrooted tree of 10 taxa using a uniform distribution for the tree topology and branch lengths generated from an exponential distribution with rate 10. A total of sixty data sets of DNA sequences of length 500 were generated using each of the simulated tree and the following three evolutionary models: JC69, K2P, and GTR+Γ. The parameter $\kappa$ in the K2P
model was set to 2.0. In the GTR+Γ model, a symmetric Dirichlet distribution with parameters (10, 10, 10, 10) was used to generate the base frequencies, and a symmetric Dirichlet with parameters (10, 10, 10, 10, 10, 10) was used to generate the GTR relative rates in the rate matrix. The discrete gamma distribution with 4 categories was used to convey among-site rate heterogeneity, with the gamma shape parameter drawn from a Gamma distribution with parameters (2, 3).

For each data set, marginal likelihoods were estimated by ASMC and SS using three evolutionary models, JC69, K2P, and GTR, respectively. In ASMC, we used $K = 1000$ and $\beta = 4$. The total number of iterations in SS was set to the product of the number of particles and number of iterations in ASMC. Table 4.2 shows the Bayesian model selection results. Both ASMC and SS choose the correct model for all of the 20 data sets generated from the JC69 and K2P model, respectively. For the data generated from GTR+Γ, SMC chooses the closest model, GTR, 18 times out of 20, while SS only chooses GTR 15 times out of 20.

Table 4.2: Comparison of model selection by ASMC and SS based on the Bayes factor.

<table>
<thead>
<tr>
<th>Method</th>
<th>Model</th>
<th>Data generated from</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>JC69</td>
</tr>
<tr>
<td>ASMC</td>
<td>JC69</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>K2P</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>GTR</td>
<td>0</td>
</tr>
<tr>
<td>SS</td>
<td>JC69</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>K2P</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>GTR</td>
<td>0</td>
</tr>
</tbody>
</table>

4.8.5 Comparison of Tree Distance Metrics

In this section, we compare the quality of reconstructed phylogenies using synthetic data. We simulated one unrooted tree, the reference tree, with 50 taxa and then generated one data set of DNA sequences of length 2000 from this tree. The ASMC was run with $\beta = 6$ and $K = 100$. The MCMC algorithm was initialized with a random tree from the prior distribution. To make a fair
comparison, we set the number of MCMC iterations to be no less than $K_{SMC} R_{SMC}$. We discarded 20% of the MCMC chain as “burn-in”. Table 4.3 summarizes the iteration numbers, the log likelihood of the consensus tree and tree distance metrics from running ASMC and MCMC. Although the computational cost of MCMC is set to about twice as high as ASMC, the log-likelihood of the consensus tree from ASMC is much higher than that from MCMC. In addition, ASMC achieves much lower RF and KF distances to the reference tree. Further, to confirm that both ASMC and MCMC can converge to the same posterior distribution, MCMC was rerun with a better starting value, namely the consensus tree obtained after running ASMC. This run of MCMC is denoted as MCMC2 in Table 4.3. The computational cost of MCMC2 is set the same as the ASMC algorithm. This time MCMC achieved similar consensus tree log-likelihood and tree distance metrics compared to ASMC, which supports that MCMC is indeed “trapped” in a sub-space.

Table 4.3: Comparison of tree distance metrics using ASMC and MCMC.

<table>
<thead>
<tr>
<th>Method</th>
<th>$R$</th>
<th>$K$</th>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASMC</td>
<td>54876</td>
<td>100</td>
<td>ConsensusLogLL</td>
<td>-72787.99</td>
</tr>
<tr>
<td>ASMC</td>
<td>54876</td>
<td>100</td>
<td>BestSampledLogLL</td>
<td>-72826.17</td>
</tr>
<tr>
<td>ASMC</td>
<td>54876</td>
<td>100</td>
<td>PartitionMetric</td>
<td>0</td>
</tr>
<tr>
<td>ASMC</td>
<td>54876</td>
<td>100</td>
<td>RobinsonFouldsMetric</td>
<td>0.70623</td>
</tr>
<tr>
<td>ASMC</td>
<td>54876</td>
<td>100</td>
<td>KuhnerFelsenstein</td>
<td>0.00990</td>
</tr>
<tr>
<td>MCMC</td>
<td>1.0E+07</td>
<td>100</td>
<td>ConsensusLogLL</td>
<td>-72833.82</td>
</tr>
<tr>
<td>MCMC</td>
<td>1.0E+07</td>
<td>100</td>
<td>PartitionMetric</td>
<td>0</td>
</tr>
<tr>
<td>MCMC</td>
<td>1.0E+07</td>
<td>100</td>
<td>RobinsonFouldsMetric</td>
<td>0.92031</td>
</tr>
<tr>
<td>MCMC</td>
<td>1.0E+07</td>
<td>100</td>
<td>KuhnerFelsenstein</td>
<td>0.03138</td>
</tr>
<tr>
<td>MCMC2</td>
<td>5.49E+06</td>
<td>100</td>
<td>ConsensusLogLL</td>
<td>-72784.86</td>
</tr>
<tr>
<td>MCMC2</td>
<td>5.49E+06</td>
<td>100</td>
<td>PartitionMetric</td>
<td>0</td>
</tr>
<tr>
<td>MCMC2</td>
<td>5.49E+06</td>
<td>100</td>
<td>RobinsonFouldsMetric</td>
<td>0.73644</td>
</tr>
<tr>
<td>MCMC2</td>
<td>5.49E+06</td>
<td>100</td>
<td>KuhnerFelsenstein</td>
<td>0.01066</td>
</tr>
</tbody>
</table>

4.8.6 Influence of Number of Threads, $\beta$, and $K$

The runtime of the ASMC is dependent on the the number of threads, as well as on the tuning parameters $\beta$ and $K$. In this section, we focus on investigating the effects of these factors on ASMC. We simulated an unrooted tree with 30 taxa at the leaves, and then generated DNA sequences of length 1500.

Next, Figure 6.5 displays the computing time versus number of threads for an implementation of ASMC where the proposal step is parallelized. The error bars represent the 95% confidence intervals based on 100 runs. We used $K = 1000$ and $\beta = 2$ for each number of threads. The results indicate that by increasing the number of cores, the speed of the ASMC algorithm can be increased notably.

In Table 4.4, we compare the performance of ASMC algorithm as a function of $K$, with $\beta$ fixed at 5. We chose four different particle values $K = 100, 300, 1000, 3000$. The marginal likelihood estimates improve as $K$ increases.
We also compared the performance of ASMC algorithm as a function of $\beta$, with $K = 1000$. We selected four distinct $\beta$ values, $\beta = 3, 4, 5, 5.3$. As expected, the marginal likelihood estimates improve when $\beta$ increases. The likelihood of the consensus trees and tree distance metrics provided by these two experiments are displayed in Appendix 8. In practice, a value of $\beta$ close to 5 is recommended as the default value.

Table 4.4: Comparison of adaptive SMC algorithm with different numbers of particles and $\beta$.

<table>
<thead>
<tr>
<th>$K$ ($\beta = 5$)</th>
<th>$\log(Z)$</th>
<th>$\beta$ ($K = 1000$)</th>
<th>$\log(Z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>-28,288.5 (-28,283.9, -28,293.7)</td>
<td>3</td>
<td>-28,524.8 (-28,466.2, -28,641.0)</td>
</tr>
<tr>
<td>300</td>
<td>-28,283.5 (-28,281.1, -28,287.9)</td>
<td>4</td>
<td>-28,312.2 (-28,304.5, -28,328.8)</td>
</tr>
<tr>
<td>1000</td>
<td>-28,280.5 (-28,278.3, -28,283.5)</td>
<td>5</td>
<td>-28,280.5 (-28,278.3, -28,283.5)</td>
</tr>
<tr>
<td>3000</td>
<td>-28,279.3 (-28,278.1, -28,280.4)</td>
<td>5.3</td>
<td>-28,279.5 (-28,278.7, -28,280.5)</td>
</tr>
</tbody>
</table>

4.8.7 Trade-off between $R$ and $K$

We conducted an experiment to investigate, for a given amount of computation, the relative importance of $R$ and $K$ in improving the quality of the posterior distribution inferred by annealed SMC. We used DSMC with a cubic annealing scheme. We selected values for the tuning parameter $K$ (100, 300, 1000, 3000, 10000), and for each value of $K$, a corresponding value for $R$ such that the total computation cost $K \cdot R$ is fixed at $10^6$. We simulated one unrooted tree of 15 taxa, and generated one data set of DNA sequences. Sequence length was set to 300. Figure 4.7 displays the marginal likelihood estimates, and KF metric provided by DSMC with different $K$ values when the total computational budget ($K \cdot R$) is fixed. This result indicates that for a given amount of computation, a relatively small $K$ and a large $R$ is optimal. However, the value of $K$ cannot be too small, as an extremely small $K$ necessarily leads to a large Monte Carlo variance.

4.8.8 Analysis of Subsampling SMC

Subsampling SMC, detailed in Appendix 1, can be used to speed up the SMC algorithms at the cost of decreasing the accuracy of estimation. The idea is to divide the data, the sites of biological sequences in our case, into batches, and only use a subset of the data in the intermediate distributions.
Figure 4.7: Performance of deterministic SMC algorithm on a fixed computational budget ($K \cdot R = 10^6$). We select 5 values of $K$, 100, 300, 1000, 3000, 10000, from left to right on X-axis.

Figure 4.8: Ratio of cost (subsampling/annealing) versus the batch size.

In this section, we evaluate the impact of the batch size (the number of sites of biological sequences in each batch), denoted $b_s$, on the speed of the algorithm and the posterior approximation.

In a first experiment, we analyzed the relative computational cost of subsampling SMC with respect to annealed SMC for different batch sizes. We simulated an unrooted tree with 10 taxa, and then generated DNA sequences of length 6000. The annealing parameter sequence $\phi_r$, $r = 0, 1, \ldots, R$, was chosen by running adaptive ASMC using $\beta = 4$ and $K = 100$. The computational cost in this subsection is measured by the total number of sites involved in computing the unnormalized posterior and the weight update function. For example, in this simulation study, the total number of annealing parameters in adaptive ASMC is 2318, and the number of sites involved in each SMC iteration is 6000. Using the fact that the likelihood for particles evaluated at iteration $r - 1$ can be used to evaluate the weight update function at iteration $r$, the total cost for ASMC is $2318 \cdot 6000 = 1.39 \times 10^7$. Figure 4.8 displays the ratio of computational cost (subsampling/annealing) versus the batch size. The cost ratio increases slowly when we increase the batch size from 1 to 100.

We investigated the performance of subsampling SMC with different batch sizes, $b_s = 1, 10, 100, 1000, 6000$ in terms of phylogenetic tree inference. We used $K = 100$ and ran the subsampling SMC algorithm 10 times for each value of $b_s$. The schedule $\phi_r$ used to compute the annealing parameter $\psi(s, \phi_r)$ in subsampling SMC was obtained by running adaptive annealed SMC once using $\beta = 4$ and $K = 100$. Figure 4.9 displays the performance of the subsampling algorithm with different
As expected, there is a trade-off between the computational cost and accuracy of most metrics: for all metrics except the partition metric, subsampling produces lower quality approximations at a lower cost. However, if the user only require a reconstruction of the tree topology, the partition metric results provide an example where subsampling is advantageous.

### 4.8.9 Comparison of ASMC and Combinatorial SMC (CSMC)

We compared the performance of the annealed SMC and the combinatorial SMC algorithm (CSMC) (Wang et al., 2015) for three different kinds of trees: clock, relaxed clock, and nonclock. The clock trees were simulated by assuming that the waiting time between two coalescent events is exponentially distributed with rate 10. The relaxed clock trees were obtained by perturbing the branch length of clock trees. More specifically, we modified each branch of length \( l \) by adding to it a noise randomly sampled from \( \text{Unif}(-0.3l, 0.3l) \). The nonclock trees were simulated with uniformly distributed tree topologies and exponentially distributed branch lengths with rate 10.

For each type of phylogenetic tree, we simulated 10 trees with 10 leaves. The JC69 evolutionary model was used to generate sequences of length 500. Three data sets were generated for each tree. We ran the annealed SMC, debiased adaptive annealed SMC (DASMC), and CSMC, respectively, for each data set three times with different random seeds. In the annealed SMC, \( \beta \) was set to 5, and \( K=100 \); in CSMC, the number of particles was set to 100,000.

Figure 4.10 shows the boxplots of log likelihood of the consensus trees, three tree distance metrics from the true trees, and computing time (in milliseconds) obtained from running the three algorithms for clock trees (top), relaxed clock trees (middle), and nonclock trees (bottom). Note that we ran CSMC for a longer time to favour this reference method. CSMC performs well for clock trees and relaxed clock trees, while the annealed SMC works for all of the three types of trees and clearly outperforms CSMC for non-clock trees.
Figure 4.10: Comparison of adaptive SMC algorithms with CSMC for three types of simulated trees: clock, relaxed clock, nonclock (from top to bottom).
4.9 Real datasets

We analyzed two difficult real data sets from TreeBASE: M336 and M1809 in Table 1 of Lakner et al. (2008). M336 contains DNA sequences of length 1949 for 27 species. In M1809, there are 59 species and the length of each DNA sequence is 1824. We compared the marginal likelihood estimates, log-likelihood of the consensus tree, and tree distance metrics provided by ASMC and MrBayes (with the default setting) with the same computational budget. The reference trees used to compute tree distances are based on at least six independent long MrBayes parallel tempering runs provided by Lakner et al. (2008). Convergence to the posterior in these “reference runs” was established with high confidence in that previous work. Note that the comparison handicaps ASMC as the set of tree moves in MrBayes is a superset of those used in ASMC. The evolutionary model we consider in real data analysis is the JC69 model.

4.9.1 Dataset M336

We used $K = 500$ and $\beta = 5.3$ for the ASMC algorithm. The log marginal likelihood estimated from ASMC is $-7103.73$, which is higher than the log marginal likelihood provided by MrBayes using Stepping Stone ($-7114.04$). Table 4.5 displays the log-likelihood of the consensus tree and tree distance metrics provided by ASMC and MrBayes. In the table, $R$ represents the number of annealing parameters in ASMC and the total number of MCMC iterations in MrBayes respectively. The log-likelihood of the consensus tree estimated from ASMC is slightly lower than MrBayes. The RF and KF metrics estimated from MrBayes are slightly higher than ASMC. The majority rule consensus tree provided by ASMC and MrBayes are identical, and coincide with the reference tree. Figure 4.11 displays the estimated majority-rule consensus trees and the clade posterior probabilities provided by ASMC and MrBayes. Most clades posterior probabilities provided by ASMC and MrBayes are close. ASMC provides lower posterior support for some clades, which is consistent with the hypothesized superior tree exploration provided by ASMC on a fixed budget.

Table 4.5: Comparison of running ASMC and MrBayes for M336 from TreeBASE.

<table>
<thead>
<tr>
<th>Method</th>
<th>$R$</th>
<th>$K$</th>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASMC</td>
<td>15706</td>
<td>500</td>
<td>ConsensusLogLL</td>
<td>-6892.16</td>
</tr>
<tr>
<td></td>
<td>15706</td>
<td>500</td>
<td>BestSampledLogLL</td>
<td>-6901.31</td>
</tr>
<tr>
<td></td>
<td>15706</td>
<td>500</td>
<td>PartitionMetric</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>15706</td>
<td>500</td>
<td>RobinsonFouldsMetric</td>
<td>0.01269</td>
</tr>
<tr>
<td></td>
<td>15706</td>
<td>500</td>
<td>KuhnerFelsenstein</td>
<td>5.55E-06</td>
</tr>
<tr>
<td>MrBayes</td>
<td>8.0E+06</td>
<td></td>
<td>ConsensusLogLL</td>
<td>-6889.52</td>
</tr>
<tr>
<td></td>
<td>8.0E+06</td>
<td></td>
<td>PartitionMetric</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8.0E+06</td>
<td></td>
<td>RobinsonFouldsMetric</td>
<td>0.01832</td>
</tr>
<tr>
<td></td>
<td>8.0E+06</td>
<td></td>
<td>KuhnerFelsenstein</td>
<td>2.25E-5</td>
</tr>
</tbody>
</table>
Figure 4.11: The majority-rule consensus trees for the M336 dataset estimated by (a) ASMC and (b) MrBayes. The numbers on the trees represent the clade posterior probabilities (number 100 is omitted). Most clades posterior probabilities provided by ASMC and MrBayes are close. ASMC provides lower posterior support for some clades, which is consistent with the hypothesized superior tree exploration provided by ASMC on a fixed budget.
### 4.9.2 Dataset M1809

We used $K = 1000$ and $\beta = 5$ for the ASMC algorithm. The log marginal likelihood estimated from ASMC is $-37,542.25$, the one estimated by MrBayes using Stepping Stone is $-37,335.73$. Table 4.6 displays the tree metrics provided by ASMC and MrBayes. The log-likelihood of the consensus tree provided by ASMC is higher than the one from MrBayes, and PM, RF, KF metrics estimated from ASMC are lower.

Table 4.6: Comparison of running ASMC and MrBayes for M1809 from TreeBASE.

<table>
<thead>
<tr>
<th>Method</th>
<th>$R$</th>
<th>$K$</th>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASMC</td>
<td>17639</td>
<td>1000</td>
<td>ConsensusLogLL</td>
<td>-36,972.513</td>
</tr>
<tr>
<td></td>
<td>17639</td>
<td>1000</td>
<td>BestSampledLogLL</td>
<td>-36,991.443</td>
</tr>
<tr>
<td></td>
<td>17639</td>
<td>1000</td>
<td>PartitionMetric</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>17639</td>
<td>1000</td>
<td>RobinsonFouldsMetric</td>
<td>0.13741</td>
</tr>
<tr>
<td></td>
<td>17639</td>
<td>1000</td>
<td>KuhnerFelsenstein</td>
<td>3.95E-4</td>
</tr>
<tr>
<td>MrBayes</td>
<td>1.76E+07</td>
<td></td>
<td>ConsensusLogLL</td>
<td>-36,996.13</td>
</tr>
<tr>
<td></td>
<td>1.76E+07</td>
<td></td>
<td>PartitionMetric</td>
<td>16.0</td>
</tr>
<tr>
<td></td>
<td>1.76E+07</td>
<td></td>
<td>RobinsonFouldsMetric</td>
<td>0.513285</td>
</tr>
<tr>
<td></td>
<td>1.76E+07</td>
<td></td>
<td>KuhnerFelsenstein</td>
<td>0.01137</td>
</tr>
</tbody>
</table>

### 4.10 Conclusion and discussion

The annealed SMC algorithm discussed in this paper provides a simple but general framework for phylogenetic tree inference. Unlike previous SMC methods in phylogenetics, annealed SMC considers the same state space for all the intermediate distributions. As a consequence, many conventional Metropolis-Hastings tree moves used in the phylogenetic MCMC literature can be utilized as the basis of SMC proposal distributions. Since MCMC tree moves are available for a large class of trees, including non-clock trees as well as strict and relaxed clock models, the annealed SMC method is automatically applicable to a wide range of phylogenetic models. It should also be relatively easy to incorporate the proposed ASMC into existing phylogenetic software packages that implement MCMC algorithms, such as MrBayes, RevBayes or BEAST.

The annealed SMC algorithm has two adaptive mechanisms, dynamic resampling and adaptive determination of annealing parameters, to make the algorithm efficient while requiring less tuning. Dynamic resampling based on ESS is a common practice in the SMC literature. Devising the annealing parameter sequence is a relatively newer practice (Del Moral et al., 2012). The annealing parameter sequence can be determined dynamically based on the conditional ESS criterion. Since the particle weights of the current iteration only depend on the previous particles, there is negligible computational cost for finding annealing parameters.

The consistency of annealed SMC discussed in the theoretical results section holds when the number of particles $K$ goes to infinity. However, $K$ cannot in practice be made arbitrarily large as the memory requirements scale linearly in $K$. In contrast, increasing the number of intermediate
distributions $R$ (in our adaptive algorithm, by increasing $\beta$) does not increase memory consumption. We conjecture that consistency for large $R$ but fixed $K$ also holds, in the sense of having the marginal distribution of each particle at the last iteration converging to the posterior distribution. We have explored the relative importance between $K$ and $R$ with fixed computational budgets using simulations. These results suggest that increasing $R$ and $K$ improves the approximation at different rates, with increasing $R$ giving bigger bang to the buck. Assuming that our conjecture on the convergence in $R$ holds true, in the regime of very large $R$ and fixed $K$, the particle population at the last iteration can be conceptualized as $K$ independent Monte Carlo samples from the true posterior distribution (in particular we conjecture that the weights will convergence to a uniform distribution and hence to naive Monte Carlo based on independent exact samples). We remind the reader that the power of independent exact Monte Carlo is that the variance does not depend on the dimensionality of the problem. Hence if $R$ is sufficiently large, a lower bound for $K$ can therefore be obtained by selecting $K$ large enough so that for independent samples $X_i$ with distribution $\pi$ the variance of the Monte Carlo average $(1/K) \sum_{k=1}^{K} f(X_i)$ is sufficiently small. Here is a concrete example: suppose we have a test function $f$ of interest, for example an indicator function on a fixed clade, with unknown posterior support $p = \int f(x)\pi(x) \, dx$. We should take $K$ large enough so that the Monte Carlo average will have a 95% Monte Carlo confidence interval having a width of no more than say $\min\{p, 1 - p\}/10$. For $p \leq 1/2$, this yields $K \geq (p/10)^{-2}(z^*)^2\text{Var}_\pi f \approx 384(1 - p)/p$, where $z^* \approx 1.96$ is the 95% critical value. For example, if the clade of interest is believed from a test run to be highly uncertain, $p \approx 1/2$, then in the large $R$ regime, at the very minimum $K = 400$ particles should be used. The value of $K$ should also be sufficiently large to accommodate the number of parallel cores available, and also to ensure that the adaptive annealing scheme is stable (i.e. that a further increase in $K$ results in a qualitatively similar annealing schedule). See also Olsson and Douc (2017) for more sophisticated schemes for estimating the Monte Carlo variance of SMC algorithms.

Importantly, annealed SMC provides an efficient way to estimate the marginal likelihood, which is still a challenging task in Bayesian phylogenetics. We have also reviewed other marginal likelihood estimation methods, including Stepping Stone and Linked Importance Sampling. Our annealed SMC algorithm enjoys advantageous theoretical properties. The main property that justifies the use of the annealed SMC is the unbiasedness of its marginal likelihood estimate. In addition, the unbiasedness of the marginal likelihood estimate can be used to test implementation correctness of the algorithm. Our simulation studies have shown that ASMC can give a similar marginal likelihood estimate as the one obtained from the ASMC with the same but deterministic annealing parameter sequence (debiased ASMC). With the same computing budget, ASMC has been demonstrated to result in more accurate estimates. Moreover, the ASMC algorithm requires less tuning than the other methods considered. Both LIS and SS need a predetermined annealing parameter sequence, which is often inconvenient to choose in practice. ASMC leads to a more stable estimate for the marginal likelihood compared to the other methods considered.

MCMC moves often come with tuning parameters. For example, proposal distributions typically have a bandwidth parameter which needs to be tuned (Roberts et al., 1997). To improve the per-
formance of annealed SMC, it would be possible to use automatic tuning of proposal distributions within SMC algorithms, as proposed in Zhou et al. (2016).

A second future direction would be to investigate modifications in the specification of the sequence of intermediate distributions. For example, Fan et al. (2010) proposed an alternative to the prior distribution to replace $\pi_0$. The same choice could be used within our framework. In another direction, it may be possible to combine the construction of Dinh et al. (2017) with ours to handle online problems via standard moves: instead of integrating the new taxon with all of its sites unannealed (which requires specialized proposals), it may be beneficial to anneal the newly introduced site.

In terms of empirical comparisons, it would be interesting to expand the set of metrics, models and datasets used to compare the algorithms. For example, in addition to the tree distance metrics used in this article, geodesic tree distance (Billera et al., 2001) is also an important metric to compare distances between phylogenetic trees. The GTP software (Owen and Provan, 2011) allows easy calculation of the geodesic tree distance.

We have investigated the subsampling SMC algorithm for “tall data” phylogenetic problems. The annealing parameters $\psi(s, \phi_r)$ of the subsampling SMC is derived from the annealing parameter sequence $\phi_r$ of the adaptive ASMC without subsampling. This choice was made for implementation convenience, and there is no reason why the two optimal sequences of distributions should coincide. To improve the algorithm performance, one direction is therefore to design an adaptive scheme tailored to the subsampling version. One challenge is that pointwise evaluation of the adaptation function $g(\phi)$ is more expensive in the subsampling setup, with a cost that grows with $\phi$. Bayesian optimization might be useful in this context. Another use of the subsampling arises in situations where the sampling algorithm is not constructed using an accept-reject step. For example, conjugate Gibbs sampling on an augmented target distribution (Lartillot, 2006) is used by PhyloBayes (Lartillot et al., 2009) to efficiently sample the evolutionary model parameters. It is not clear how the conjugate Gibbs sampling step can be modified to accommodate the annealed distribution in Equation (10) for $\phi < 1$. On the other hand, conjugate sampling is directly applicable to intermediate distributions that consist in taking subsets of sites. This sequence of distributions could be used to handle conjugate Gibbs sampling not only in annealed SMC but also in the context of parallel tempering or any other sequence of measure based method. One last line of work is to combine control variates to annealed SMC to reduce the variance of the likelihood estimator, a general strategy that has been very successful in other subsampling work (Bardenet et al., 2017).
4.11 APPENDIX 1

4.11.1 Construction of Intermediate Distributions for Subsampling SMC

Let us decompose the unnormalized posterior distribution as

\[ \gamma_1(x) = p(x) \prod_{s=1}^{\#S} p(y_s|x), \]

where \( x \) refers to the phylogenetic tree and evolutionary parameter of interest, \( s \) is an index for one batch of sites from a biological sequence, and \( \#S \) represents the total number of batches. Each batch contains one or more sites of the biological sequence; we denote the number of sites in each batch by \( b_s \).

Consider the annealing parameter sequence \( 0 = \phi_0 < \phi_1 < \cdots < \phi_R = 1 \). We define the sequence of intermediate distributions for subsampling as follows:

\[ \gamma_{\phi_r}(x) = p(x) \prod_{s=1}^{\#S} p(y_s|x)^{\psi(s, \phi_r)}, \]

where

\[ \psi(s, \phi_r) = \begin{cases} 
1 & \text{if } \phi_r \geq s/\#S, \\
0 & \text{if } \phi_r \leq (s - 1)/\#S, \\
\#S \cdot \phi_r - (s - 1) & \text{otherwise.} 
\end{cases} \]

The subsampling SMC algorithm is a more general version of the annealed SMC algorithm. If we define \( \#S = 1 \) in \( \gamma_1(x) \), then the sequence of intermediate distributions of subsampling SMC is exactly the same as the intermediate distributions of the annealed SMC. In this case, the computational cost of subsampling SMC is exactly the same as the annealed SMC. Another extreme case is that \( \#S = n \), in which case we sequentially incorporate the sites of sequence one by one.

4.12 APPENDIX 2

4.12.1 Theoretical Foundations of Annealed SMC

In this section, we review the construction of Del Moral et al. (2006), which is the basis for our work. See also Wang et al. (2015) for a similar construction tailored to a phylogenetic setup.

The corresponding sequence of unnormalized distributions are denoted by \( \{\gamma_r\}_{1}^{R} \). The annealed SMC can be obtained by defining an auxiliary sequence of distributions that admit the distribution of interest, \( \pi_r(x_r) \), as the marginal of the latest iteration

\[ \tilde{\pi}_r(x_r) = \pi_r(x_r) \prod_{j=1}^{r-1} L_j(x_{j+1}, x_j), \]
where $L_j(x_{j+1}, x_j)$ is an auxiliary “backward” Markov kernel with $\int L_j(x_{j+1}, x_j) \, dx_j = 1$. We never sample from $L_j$, rather its role is to allow us to derive weight updates that yield a valid SMC algorithm.

The idea is then to apply standard SMC (i.e. SMC for product spaces such as state space models) to this auxiliary sequence of distributions, $\tilde{\pi}_1, \tilde{\pi}_2, \ldots, \tilde{\pi}_R$. The resulting sampler has a weight update given by

$$w(x_{r-1}, x_r) \propto \frac{\tilde{\pi}_r(x_r)}{\tilde{\pi}_r(x_{r-1})} \frac{1}{K_r(x_{r-1}, x_r)} \frac{\pi_r(x_r)}{\pi_{r-1}(x_{r-1})} K_r(x_{r-1}, x_r),$$

which is different from the one in a standard SMC.

When $K_r$ satisfies global balance with respect to $\pi_r$, a convenient backward Markov kernel that allows an easy evaluation of the importance weight is

$$L_{r-1}(x_r, x_{r-1}) = \frac{\pi_r(x_{r-1}) K_r(x_{r-1}, x_r)}{\pi_r(x_r)}.$$

This choice is a properly normalized backward kernel, $\int L_{r-1}(x_r, x_{r-1}) \, dx_{r-1} = 1$: this follows from the assumption that $K_r$ satisfies global balance with respect to $\pi_r$. With this backward kernel, the incremental importance weight becomes

$$w(x_{r-1}, x_r) = \frac{\gamma_r(x_r)}{\gamma_{r-1}(x_{r-1})} \frac{L_{r-1}(x_r, x_{r-1})}{\pi_r(x_{r-1}) K_r(x_{r-1}, x_r)} \frac{1}{\pi_r(x_r) K_r(x_{r-1}, x_r)} \frac{\pi_r(x_{r-1}) K_r(x_{r-1}, x_r)}{\pi_r(x_r)} \frac{1}{\pi_r(x_{r-1})} \frac{1}{\gamma_{r-1}(x_{r-1})} \frac{\pi_r(x_{r-1}) K_r(x_{r-1}, x_r)}{\pi_r(x_r)} \frac{1}{\pi_r(x_{r-1})}.$$

### 4.12.2 General Estimates of Marginal Likelihood

In Section Basic Annealed SMC Algorithm we describe the estimator for marginal likelihood in a simplified setting, i.e. without adaptation. Here we describe the marginal likelihood estimator in full generality.

Recall that we denote the marginal likelihood by $Z$ for simplicity. With a slight abuse of notation, we use $K_r(x_{r-1}, \cdot)$ to denote the proposal distribution for $x_r$ in this section.

Let us start by rewriting the normalization constant of the first intermediate distribution as

$$Z_1 = \int \frac{\gamma_1(x_1)}{K_1(x_1)} K_1(x_1) \, dx_1 = \int w_1(x_1) K_1(x_1) \, dx_1,$$

where $K_1(\cdot)$ is the proposal distribution for $x_1$. 
Correspondingly, an estimate of $Z_1$ is

$$Z_{1,K} = \frac{1}{K} \sum_{k=1}^{K} w_{1,k}. $$

Similarly, we can rewrite the ratio of the normalization constants of two intermediate distributions as

$$\frac{Z_r}{Z_{r-1}} = \frac{\int \gamma_r(x_r)dx_r}{\int \gamma_{r-1}(x_{r-1})dx_{r-1}} = \frac{\int \gamma_r(x_r)dx_r}{\gamma_{r-1}(x_{r-1})/\pi_{r-1}(x_{r-1})} = \int \frac{\gamma_r(x_r)}{\gamma_{r-1}(x_{r-1})} \pi_{r-1}(x_{r-1})dx_r = \int \frac{\gamma_r(x_r)\pi_{r-1}(x_{r-1})K_r(x_{r-1}, x_r)}{\pi_{r-1}(x_{r-1})K_r(x_{r-1}, x_r)} dx_r = \int w_r(x_r)\pi_{r-1}(x_{r-1})K_r(x_{r-1}, x_r)dx_r. $$

Straightforwardly, an estimate of $Z_r/Z_{r-1}$ is provided by

$$\hat{Z}_r = \frac{1}{K} \sum_{k=1}^{K} w_{r,k}. $$

Since the estimate of the marginal likelihood can be rewritten as

$$Z \equiv Z_R = Z_1^{R} \prod_{r=2}^{R} \frac{Z_r}{Z_{r-1}}, $$

an estimate of the marginal likelihood $Z$ is

$$\hat{Z}_{R,K} = \prod_{r=1}^{R} \left( \frac{1}{K} \sum_{k=1}^{K} w_{r,k} \right) = \prod_{r=1}^{R} \left( \frac{1}{K} \sum_{k=1}^{K} \{p(y|x_{r-1,k})\}^{\phi_{r-1}} \right), \quad (19)$$

which can be obtained from an SMC algorithm readily. If resampling is not conducted at each iteration $r$, an alternative form is provided by

$$\hat{Z}_{R,K} = \prod_{j=1}^{t_{R-1}+1} \left( \sum_{k=1}^{K} W_{n_{j-1,k}} \prod_{m=n_{j-1}+1}^{n_j} \{p(y|x_{m-1,k})\}^{\phi_m-\phi_{m-1}} \right), \quad (20)$$

where $n_j$ is the SMC iteration index at which we do the $j$th resampling, $t_{R-1}$ is the number of resampling steps between 1 and $R - 1$.

### 4.12.3 Unbiasedness, Consistency and Central Limit Theorem for Annealed SMC

Here we provide more information on the theoretical properties discussed in Section Properties of Annealed Sequential Monte Carlo.
Theorem 1 (Unbiasedness): For fixed $0 = \phi_0 < \phi_1 < \cdots < \phi_R = 1$, $\hat{Z}_{R,K}$ is an unbiased estimate of $Z$.

$$\mathbb{E}(\hat{Z}_{R,K}) = Z.$$ 

This result is well known in the literature, although many statements of the result are specialized to SMC for state space models (Doucet and Johansen, 2009). The results in Theorem 7.4.2 of Del Moral (2004) provide a very general set of conditions which includes the annealed SMC algorithm presented here. However, the theoretical framework in Del Moral (2004) being very general and abstract, we outline below an alternative line of argument to establish unbiasedness of phylogenetic annealed SMC.

First, by the construction reviewed in Section Theoretical Foundations of Annealed SMC, we can transform the sequence of distributions on a fixed state space, $\pi_r(x_r)$, into a sequence of augmented distributions $\tilde{\pi}_r(x_r)$ on a product space admitting $\pi_r(x_r)$ as a marginal. We now apply Theorem 2 of Andrieu et al. (2010) with the distribution $\pi_r(x_{1:n})$ in this reference set to $\tilde{\pi}_r(x_r)$ in our notation. To be able to use Theorem 2, we only need to establish the “minimum assumptions” 1 and 2 in Andrieu et al. (2010). Assumption 1 is satisfied by the fact that valid MCMC proposals are guaranteed to be such that $q(x,x') > 0 \iff q(x',x) > 0$. Assumption 2 holds since we use multinomial resampling. Next, since the conditions of Theorem 2 of Andrieu et al. (2010) hold, we have the following result from the proof of Theorem 2 in Appendix B1 of Andrieu et al. (2010):

$$\frac{\tilde{\pi}^N(k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1})}{q^N(k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1})} = \frac{\hat{Z}^N(\bar{x}_1, \ldots, \bar{x}_p)}{Z},$$

and hence

$$Z \tilde{\pi}^N(k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1}) = \hat{Z}^N(\bar{x}_1, \ldots, \bar{x}_p) q^N(k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1}).$$

Now taking the integral on all variables $k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1}$ with respect to the reference measure $\mu$ associated to $\tilde{\pi}^N$, we obtain:

$$Z \int \tilde{\pi}^N(k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1}) \, d\mu(k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1})$$

$$= \int \hat{Z}^N(\bar{x}_1, \ldots, \bar{x}_p) q^N(k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1}) \, d\mu(k, \bar{x}_1, \ldots, \bar{x}_p, a_1, \ldots, a_{p-1}).$$

The left-hand side is just $Z$ since $\tilde{\pi}^N$ is a density with respect to $\mu$. For the right-hand side, note that $q^N$ is the law of the full set of states produced by the particle filter, hence the right-hand side is just $\mathbb{E}[\hat{Z}_{R,K}]$ in our notation. This concludes the proof.

Next, we discuss consistency results. In the SMC literature, they are generally available both in the $L^2$ convergence and almost sure convergence flavours. We cover the $L^2$ case here and refer to Del Moral et al. (2006) for almost sure consistency results.
Theorem 2 (Consistency): Assume there is a constant $C$ such that $|f| \leq C$ and $w_{r,k} \leq C$ almost surely. For a fixed $\phi_r$ ($r = 1, \ldots, R$), the annealed SMC algorithm provides asymptotically consistent estimates:

$$\sum_{k=1}^{K} W_{r,k} f(x_{r,k}) \to \int \pi_r(x) f(x) \, dx \text{ as } K \to \infty,$$

where the convergence holds in the $L^2$ norm sense.

The result can be deduced from Proposition 5 in Wang et al. (2015) as follows. Assumption 3 in Wang et al. (2015) holds since MCMC proposals satisfy $q(x, x') > 0 \iff q(x', x) > 0$. Assumption 4 holds since the support of the prior coincides with support of the posterior.

Note that the assumption that the weights are bounded is not valid for general tree spaces. However, if the branch lengths are assumed to be bounded then the space is compact and the assumption therefore holds in that setting.

Finally, we turn to the central limit theorem.

Theorem 3 (Central Limit Theorem): Under the integrability conditions given in Theorem 1 of Chopin (2004), or Del Moral (2004), section 9.4, pages 300 – 306,

$$K^{1/2} \left[ \sum_{k=1}^{K} W_{r,k} f(x_{r,k}) - \int \pi_r(x) f(x) \, dx \right] \to N(0, \sigma_r^2(f)) \text{ as } K \to \infty,$$

where the convergence is in distribution. The form of asymptotic variance $\sigma_r^2(f)$ depends on the resampling scheme, the Markov kernel $K_r$ and the artificial backward kernel $L_r$. We refer readers to Del Moral et al. (2006) for details of this asymptotic variance.

4.13 APPENDIX 3

4.13.1 MCMC Proposals for Bayesian Phylogenetics

In this paper, we used the proposals $q_i^r$ defined as follow:

1. $q_1^1$: the multiplicative branch proposal. This proposal picks one edge at random and multiply its current value by a random number distributed uniformly in $[1/a, a]$ for some fixed parameter $a > 1$ (controlling how bold the move is) (Lakner et al., 2008).

2. $q_2^1$: the global multiplicative branch proposal that proposes all the branch lengths by applying the above multiplicative branch proposal to each branch.

3. $q_3^1$: the stochastic NNI proposal. We consider the nearest neighbor interchange (NNI) (Jow et al., 2002) to propose a new tree topology.

4. $q_4^1$: the stochastic NNI proposal with resampling the edge that uses the above NNI proposal in (3) and the multiplicative branch proposal in (1) for the edge under consideration.
5. $q^5$: the Subtree Prune and Regraft (SPR) move that selects and removes a subtree from the main tree and reinserts it elsewhere on the main tree to create a new tree.

Note that here we only describe the MCMC kernels for phylogenetic trees. To sample evolutionary parameters $\theta$, one can use simple proposals such as symmetric Gaussian distributions, or more complex ones, see for example Zhao et al. (2016).

### 4.14 APPENDIX 4

#### 4.14.1 Comparison of Resampling Strategies

In this section we compare the performance of adaptive annealed SMC with three different resampling thresholds. The first resampling threshold is $\epsilon_1 = 0$. In this case, the particles are never resampled. The second resampling threshold is $\epsilon_2 = 1$, in which case resampling is triggered at every iteration. The third resampling threshold is $\epsilon_3 = 0.5$. The resampling method we used in all our experiments was the multinomial resampling scheme. We simulated one unrooted tree of 15 taxa, and generated one data set of DNA sequences of length 200. The tree simulation setup was the same as in Section Simulation Studies. We ran adaptive annealed SMC algorithm 20 times with the three resampling thresholds described above. We used $rCESs_f = 0.99999$ and $K = 100$. Figure 4.12 demonstrates the advantage of resampling triggered by a threshold of $\epsilon_3 = 0.5$ over the other two choices by displaying the marginal likelihood estimates, log likelihood of the consensus tree and tree metrics provided by adaptive annealed SMC using three different $\epsilon$’s. The log marginal likelihood estimate and log likelihood of the consensus tree provided by adaptive annealed SMC using $\epsilon_3 = 0.5$ are higher and admit smaller variation. The PF, RF and KF metrics provided by adaptive annealed SMC using $\epsilon_3 = 0.5$ are lowest. Therefore the threshold 0.5 has been used in the rest of the paper.

### 4.15 APPENDIX 5

#### 4.15.1 Review of Particle Degeneracy Measures

The two adaptive schemes in ASMC, adaptively conducting resampling and the automatically construction of the annealing parameter sequence, rely on being able to assess the quality of a particle

Figure 4.12: Comparison of three resampling thresholds, 0, 0.5 and 1.
approximation. For completeness, we provide some background in this section on the classical notation of Effective Sample Size (ESS) and of conditional ESS (CESS), a recent generalization which we use here (Zhou et al., 2016). The notion of ESS in the context of importance sampling (IS) or SMC is distinct from the notion of ESS in the context of MCMC. The two are related in the sense of expressing a variance inflation compared to an idealized Monte Carlo scheme but they differ in the details. We will assume from now on that ESS refers to the SMC context.

We will also use a slight variation of the derivation of ESS and CESS where the measures obtained are normalized to be between zero and one (some hyper-parameters of the adaptive algorithms are easier to express in this fashion). We use the terminology relative (conditional) ESS to avoid confusion.

The fundamental motivation of (relative and/or conditional) ESS stems from the analysis of the error of Monte Carlo estimators. Recall that for a given function of interest \( f \) (think for example of \( f \) being an indicator function on a clade),

\[
I = \int \pi_r(\text{d}x)f(x) \approx \sum_{k=1}^{K} W_{r,k}f(x_{r,k}) =: \hat{I}.
\]

The quantity on the right hand side is a random variable (with respect to the randomness of the SMC algorithm), \( \hat{I} \), and we can think about it as an estimator of the deterministic quantity on the left hand side, \( I \). Moreover the right hand side is a real-valued random variable so we can define its mean square error, which can be further decomposed as a variance term and a squared bias term. For SMC algorithms, the variance term dominates as the number of particles goes to infinity (Del Moral, 2004). For this reason, we are interested in estimates of the variance of \( \hat{I} \) across SMC random seeds, \( \text{Var}_{\text{SMC}}[\hat{I}] \). However, the variance of \( \hat{I} \) depends on the choice of function \( f \), which is problem dependent, and we would like to remove this dependency. The first step is to consider a notion of relative variance, comparing to the variance we would obtain from a basic Monte Carlo scheme \( \hat{I}^* \) relying on iid exact samples \( x_1^*, \ldots, x_K^* \sim \pi \), \( \text{Var}_{\text{MC}}[\hat{I}^*] = \text{Var} \left[ \frac{1}{K} \sum_{k=1}^{K} f(x_k^*) \right] \). To make further progress, we will make approximations of the ratio \( \text{Var}_{\text{MC}}[\hat{I}^*]/\text{Var}_{\text{SMC}}[\hat{I}] \).

To understand these approximations, let us start with a simplified version of Algorithm 12, where the function NextAnnealingParameter returns the value 1.0. In this setting, no resampling occurs, and the algorithm reduces to an importance sampling algorithm (more specifically, it reduces to a single iteration of the Annealed Importance Sampling (AIS) algorithm (Neal, 2001)). Importance sampling is easier to analyze since the individual particles are independent and identically distributed, allowing us to summarize the behaviour based on one particle, say \( k = 1 \). If we assume further that (A1) \( \gamma_i = \pi_i \), i.e. that the normalization constant is one, then a classical argument by Kong (1992)
based on the Delta method yields

\[
\frac{\text{Var}_{\text{MC}}[\hat{I}^*]}{\text{Var}_{\text{SMC}}[\hat{I}]} = \frac{\text{Var}_{\text{MC}}[\hat{I}^*]}{\text{Var}_{\pi_0}[\hat{I}]} \approx \frac{1}{1 + \text{Var}_{\pi_0}[\hat{w}_{1,1}]},
\]

\[
= \frac{1}{\mathbb{E}_{\pi_0}[(\hat{w}_{1,1})^2]},
\]

where we used the fact that in this simple setting the distribution of one proposed particle is just \(\pi_0\), so \(\text{Var}_{\text{SMC}}[\cdot] = \text{Var}_{\pi_0}[\cdot]\) and in the last line,

\[
\mathbb{E}_{\pi_0}[\hat{w}_{1,1}] = \int \pi_0(x_{0,1}) \frac{\pi_1(x_{0,1})}{\pi_0(x_{0,1})} dx_{0,1} = 1.
\]

In general, assumption (A1) does not hold, i.e. the normalization constant is not one, so for a general one-step AIS algorithm we get instead the approximation:

\[
\frac{\text{Var}_{\text{MC}}[\hat{I}^*]}{\text{Var}_{\text{SMC}}[\hat{I}]} \approx \left( \mathbb{E}_{\pi_0} \left[ \left( \frac{\pi_1(x_{0,1})}{\pi_0(x_{0,1})} \right)^2 \right] \right)^{-1}
\]

\[
= \left( \mathbb{E}_{\pi_0} \left[ \left( \frac{\gamma_1(x_{0,1})}{\gamma_0(x_{0,1})} \right)^2 \right] \right)^{-1}
\]

\[
= \left( \frac{Z_1}{Z_0} \right)^2 \left/ \mathbb{E}_{\pi_0} \left[ \left( \frac{\gamma_1(x_{0,1})}{\gamma_0(x_{0,1})} \right)^2 \right] \right. \].
\]

Generalizing the notation of this section into a general SMC setup, \(\pi_1\) here plays the role of the current iteration, and \(\pi_0\), of the previous iteration. However, since \(\pi_0\) is not known in this case, we plug-in a particle approximation \(\hat{\pi}_0 = \sum_{k=1}^{K} W_{0,k} \delta_{x_{0,k}}\) to get:

\[
\frac{\text{Var}_{\text{MC}}[\hat{I}^*]}{\text{Var}_{\text{SMC}}[\hat{I}]} \approx \left( \sum_{k=1}^{K} W_{0,k} \gamma_1(x_{0,k}) \right)^2 \left/ \sum_{k=1}^{K} W_{0,k} \gamma_1(x_{0,k}) \right. \].
\]

The effect of this additional approximation is that it makes our estimator over-optimistic, by ignoring the error of the approximation \(\hat{\pi}_0\) of \(\pi_0\). It is nonetheless a useful tool to assess the degradation of performance over a small number of SMC iterations.

Finally, since the ratio of normalization constants is also unknown, we also need to estimate it. Based on a particle approximation of Equation (14), we obtain:

\[
\frac{\text{Var}_{\text{MC}}[\hat{I}^*]}{\text{Var}_{\text{SMC}}[\hat{I}]} \approx \left( \sum_{k=1}^{K} W_{0,k} \frac{\gamma_1(x_{0,k})}{\gamma_0(x_{0,k})} \right)^2 \left/ \sum_{k=1}^{K} W_{0,k} \left( \frac{\gamma_1(x_{0,k})}{\gamma_0(x_{0,k})} \right)^2 \right. \].
\]

This quantity is called the relative conditional ESS (rCESS), Equation (11). Having a high rCESS value is a necessary but not sufficient condition for a good SMC approximation. If it is low during some SMC iteration, especially an iteration close to the final iteration, then with high probability
most of the particles will have very small or zero weights, which will lead to a collapse of the quality of the annealed SMC algorithm.

4.15.2 Comparison of Relative ESS (rESS) and Relative CESS (rCESS)

In earlier work on adaptive SMC methods, the function NextAnnealingParameter was implemented using a different criterion based on rESS instead of rCESS. Later, Zhou et al. (2016) argued that rCESS was more appropriate. Here we confirm that this is also the case in a phylogenetic context. We provide two experiments. In the first experiment, we simulated one unrooted tree of 10 taxa, and generated one data set of DNA sequences and each sequence has length 100. The setup of tree simulation is the same as Section Simulation Studies. We ran adaptive annealed SMC algorithm in two schemes: (a) $rCESS_r = 0.99$; (b) $rESS_r = 0.99$. The one based on $rESS_r$ only differs in the way NextAnnealingParameter is implemented; shown in Algorithm 14. We used $K = 1000$ particles. Resampling of particles was triggered when $rESS < 0.5$. Figure 4.13 demonstrates the advantage of using rCESS over rESS in adaptive annealed SMC. The annealing parameter difference $(\phi_r - \phi_{r-1})$ increases smoothly in the rCESS scheme, while in the rESS scheme there are big gaps in annealing parameter increment after doing resampling, then the consecutive annealing parameter change decreases gradually until the next resampling time. The number of iterations $R$ for adaptive annealed SMC using rESS is much larger than using rCESS.

Algorithm 14 Alternative NextAnnealingParameter procedure (sub-optimal)

1: **Inputs:** (a) Particle population from previous SMC iteration $(x_{r-1,:}, w_{r-1,:})$; (b) Annealing parameter $\phi_{r-1}$ of previous SMC iteration; (c) A degeneracy decay target $\alpha \in (0, 1)$.
2: **Outputs:** automatic choice of annealing parameter $\phi_r$.
3: Initialize the function $\tilde{g}$ assessing the particle population quality associated to a putative annealing parameter $\phi$:

\[
\tilde{g}(\phi) = \left( \frac{\sum_{k=1}^{K} W_{r-1,k} P(y|x_{r-1,k})^{\phi - \phi_{r-1}}}{\sum_{k=1}^{K} (W_{r-1,k} P(y|x_{r-1,k})^{\phi - \phi_{r-1}})^2} \right)^2.
\]
4: if $\tilde{g}(1) \geq \alpha \tilde{g}(\phi_{r-1})$ then
5: return $\phi_r = 1$.
6: else
7: return $\phi_r = \phi^* \in (\phi_{r-1}, 1)$ such that $\tilde{g}(\phi^*) = \alpha \tilde{g}(\phi_{r-1})$ via bisection.

In our second experiment, we compared the performance of adaptive annealed SMC using rCESS and rESS in terms of tree metrics and marginal likelihood. We simulated one unrooted tree of 15 taxa, and generated one data set of DNA sequences. Each sequence has length 200. The tree simulation setup is the same as Section Simulation Studies. We ran adaptive annealed SMC algorithm 20 times with $rCESS_r = 0.999$ and $rESS_r = 0.978$ respectively. The number of particles was set to $K = 500$. Under this setting, the computational costs of the two schemes are quite similar. The numbers of annealing parameters ($R$) selected via rCESS and rESS are similar. Figure 4.14 displays the marginal likelihood estimates, consensus likelihood and tree metrics provided by adaptive SMC using rCESS.
Figure 4.13: Comparison of rCESS (left) and rESS (right) in terms of rESS as a function of $r$ (top) and $\phi_r - \phi_{r-1}$ as a function of $\phi_r$ (bottom).

Figure 4.14: Comparison of adaptive annealed SMC using rCESS and rESS in terms of estimating the log marginal likelihood, the log likelihood of the consensus tree, tree distance metrics, and the number of SMC iterations ($R$).
and rESS. The log marginal likelihood estimates and log consensus likelihoods provided by adaptive SMC using rCESS are higher and have lower variability. The PF, RF and KF metrics provided by the two schemes are quite close, while the metrics provided by rCESS scheme have lower variability.

4.16 APPENDIX 6

4.16.1 Estimates of Marginal Likelihood from LIS

We described the LIS procedure as follows:

1. Sample an index \( v_0 \) randomly from \( \{1, 2, \ldots, N\} \), and sample \( x_{0,v_1} \sim \pi_0(\cdot) \).
2. For \( d = 0, 1, \ldots, D \), sample \( N \) states from \( \pi_d \) as follows:
   (a) If \( d > 0 \): sample an index \( v_d \) from \( \{1, 2, \ldots, N\} \), and set \( x_{d,v_d} = x_{d-1,v_d} \).
   (b) For \( k = v_d + 1, \ldots, N \), sample \( x_{d,k} \) from the forward kernel \( x_{d,k} \sim K_d(x_{d,k-1}, \cdot) \).
   (c) For \( k = v_d - 1, \ldots, 1 \), sample \( x_{d,k} \) from the backward kernel \( x_{d,k} \sim L_d(x_{d,k+1}, \cdot) \).
   (d) If \( d < D \), sample \( \mu_d \) from \( \{1, 2, \ldots, N_d\} \) according to the following probabilities:
      \[
p(\mu_d|x_d) = \frac{\gamma_{d-1}x_{d,\mu_d}}{\gamma_d(x_{d,\mu_d})} \sqrt{\frac{1}{N} \sum_{k=1}^N \frac{\gamma_{d-1}x_{d-1,k}}{\gamma_d(x_{d,k})}} /
      \sqrt{\frac{1}{N} \sum_{k=1}^N \frac{\gamma_{d-1}x_{d-1,k}}{\gamma_d(x_{d,k})}}.
      \]
      and set \( x_{d+1} \) to \( x_{d,\mu_d} \).
3. Compute the likelihood estimate
   \[
   \hat{Z}_{LIS} = \prod_{d=1}^D \left[ \frac{1}{N} \sum_{k=1}^N \frac{\gamma_{d-1}x_{d-1,k}}{\gamma_d(x_{d,k})} \right]^{N_d} \frac{1}{N} \sum_{k=1}^N \frac{\gamma_{d-1}x_{d-1,k}}{\gamma_d(x_{d,k})}.
   \]

Note that if the backward kernel is reversible, then the forward kernel is the same as backward kernel. In this paper, we use the MCMC kernel as backward and forward kernels in LIS.

4.17 APPENDIX 7

4.17.1 Derivation of Upper Bound of CV

\[
CV = \frac{sd(\hat{Z})}{\mathbb{E}(\hat{Z})} = \frac{\sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{Z}_i - \frac{1}{n} \sum_{i=1}^n \hat{Z}_i)^2}}{\frac{1}{n} \sum_{i=1}^n \hat{Z}_i} = \sqrt{n} \sqrt{\frac{\sum_{i=1}^n (\hat{Z}_i - \frac{1}{n})^2}{\sum_{i=1}^n \hat{Z}_i}}.
\]
Figure 4.15: Comparison of adaptive SMC algorithm with different numbers of particles, from left to right $K = 100, 300, 1000, 3000$.

Figure 4.16: Comparison of adaptive SMC algorithm with different $\beta$, from left to right $\beta = 3, 4, 4.3, 5, 5.3$ (x-axis). Here $R$ is the total number of SMC iterations.

For non-negative $Z_i$, the CV is maximized when $\frac{\hat{Z}_i}{\sum_{i=1}^{n} Z_i} = 1$ for some $i$, and 0 for the rest. In this extreme case, $\hat{Z}_i$ is much larger than the rest. The upper bound of the CV can be simplified to $\sqrt{n - 1}$.

4.18 APPENDIX 8

4.18.1 Tuning of $\beta$ and $K$

In Figure 4.15, we compare the performance of ASMC algorithm as a function of $K$, with $\beta$ fixed at 5. We used four different numbers of particles $K = 100, 300, 1000, 3000$. Both the marginal likelihood estimate and tree metrics improve as $K$ increases. Figure 4.16 displays the performance of ASMC algorithm as a function of $\beta$, with $K = 1000$. $R$ is the total number of SMC iterations. We used five distinct $\beta$ values, $\beta = 3, 4, 4.3, 5, 5.3$. The marginal likelihood estimates and tree metrics improve as $\beta$ increases; they tend to be stable after $\beta$ reaches 5. A larger value of $\beta$ can improve the performance of ASMC more significantly than an increase in $K$. 
4.19 APPENDIX 9

4.19.1 Comparison of ASMC, DASMC, LIS and SS for large $K$

In this experiment, we focus on evaluating the marginal likelihood estimates using ASMC, DASMC, LIS and SS with a shared, large computational budget. We simulated an unrooted tree of 4 taxa, generated one data set of DNA sequences of length 10. Every algorithm for each data set was repeated 50 times with different random seeds. We set $\beta = 2$ and $K = 200000$. The setup of DASMC, LIS and SS is the same as Section Comparison of marginal likelihood Estimates.

Figure 4.17 shows the comparison of the performance of the four algorithms in terms of the marginal likelihoods in the log scale. The mean log marginalized likelihood estimates provided by ASMC, DASMC, LIS and SS are quite close. The variance of estimates for ASMC and DASMC is smaller than LIS and SS.
Chapter 5

Particle Gibbs Sampling for Bayesian Phylogenetic inference

5.1 Introduction

Phylogeny reconstruction method is to recover the evolutionary history of biological species or other entries. It has long been attractive to researchers in comparative linguistics (Dunn et al., 2005; Mace and Holden, 2005; Pagel, 2009; Atkinson and Gray, 2005), biology (Harvey et al., 1991; Huelsenbeck et al., 2001; Nei and Kumar, 2000), and archaeology (Cavalli-Sforza et al., 1988; O’Brien et al., 2001). In this article, we are interested in reconstructing the evolutionary relationship, represented by a phylogenetic tree, for different species.

A phylogenetic tree is latent, and typically estimated by using the biological sequences (e.g. DNA sequences) observed at tips of the tree. There is a rich literature on phylogenetic tree reconstruction. Bayesian approaches are extremely popular for phylogenetic inference. In Bayesian phylogenetics (Lemey et al., 2009; Drummond and Suchard, 2010; Huelsenbeck and Ronquist, 2001; Ronquist and Huelsenbeck, 2003; Ronquist et al., 2012; Suchard and Redelings, 2006), the goal of these methods is to compute a posterior over a phylogenetic tree space. It is generally not possible to obtain an explicit expression for this posterior as the exact calculation involves integrating over all possible trees. The standard inference algorithm for Bayesian phylogenetics is Markov chain Monte Carlo methods.

Many user-friendly software packages have been developed for implementing MCMC for phylogenetic inference, such as MrBayes (Ronquist et al., 2012), BEAST (Drummond and Rambaut, 2007), and BAiL-Phy (Suchard and Redelings, 2006). At each MCMC iteration, local changes of a tree are proposed (Lakner et al., 2008; Jow et al., 2002; Allen and Steel, 2001). A large change of a tree proposed by MCMC has high probability of being rejected. Hence, this collection of MCMC samples often leads to poor mixing. For large sequencing data, MCMC samples are computationally expensive as the evaluation of an unnormalized posterior is computationally intensive.

Wang et al. (2015) proposed a combinatorial sequential Monte Carlo (CSMC) for Bayesian phylogenetic inference. This method has been shown to be a good alternative to Markov chain
Monte Carlo. In CSMC, a flexible proposal distribution is introduced to construct non-clock trees. In addition, a backward kernel in CSMC is proposed to correct the overcounting issue in non-clock tree inference, as conventional SMC algorithms applied to non-clock tree reconstruction would favor trees that can be constructed in multiple ways. We provide an example to illustrate the overcounting issue in Figure 5.2 panel (a). However, CSMC fails to serve as an efficient tree proposal in the particle Gibbs framework for estimating the parameter in nucleotide substitution model (Andrieu et al., 2010), due to the simple and restricted propagation step. This limitation motivates us to propose a more elaborate CSMC algorithm to improve the mixing of a particle Gibbs sampler.

In this work, we propose a novel CSMC algorithm, which we call the revert-merge CSMC. The proposal distribution of revert-merge CSMC is more flexible than the naive version of CSMC algorithm. In Bayesian phylogenetics, the proposal distribution is important for exploring the complex tree posterior distribution. For instance, Fourment et al. (2018) have investigated using different tree proposals in SMC and found that a good proposal is essential in exploring posterior of trees. The revert-merge CSMC is easy to parallelize by allocating samples into different cores, which is more scalable in inferring posterior trees for large sequence data. A novel backward kernel is proposed to correct the overcounting issue in revert-merge CSMC. We further show that consistency properties of tree estimators are guaranteed under weak conditions.

The main advantage of using this new CSMC is that we can jointly estimate the phylogenetic tree and the associated evolutionary parameter in a particle Gibbs sampler (PGs) scheme. We use the conditional version of the proposed revert-merge CSMC to design an efficient high-dimensional tree proposal distribution in PGs. Unlike the naive version of CSMC, our proposed method allows us to conduct ancestor sampling (Lindsten et al., 2014) to improve the mixing of PGs. We conduct a series of simulation studies to evaluate the quality of tree reconstruction using the two CSMCs. Particle Gibbs with our CSMC estimates the tree more accurately. Interacting particle Markov chain Monte Carlo (Rainforth et al., 2016) is more efficient than particle Gibbs sampler with fixed computational budget.

Sequential Monte Carlo (SMC) algorithm are popular for inference in state-space models (Doucet et al., 2001; Liu, 2001) and can be applied to more general settings (Del Moral et al., 2006). There is a growing body of literature on phylogenetic tree reconstruction based on sequential Monte Carlo methods. Several SMC approaches (Teh et al., 2008; Görür and Teh, 2009; Bouchard-Côté et al., 2012; Görür et al., 2012) have been proposed to estimate clock trees and have been demonstrated to be good alternatives to MCMC methods. Wang et al. (2015) proposed a combinatorial sequential Monte Carlo, which our algorithm builds on, that addresses the limitations of previous SMC methods for phylogenetic trees. SMC algorithms have also been applied to online phylogenetic inference scenarios, in which the taxonomic data arrive sequentially in an online pattern. Dinh et al. (2017) focused on the theoretical framework for the online phylogenetic inference via SMC approaches. Fourment et al. (2018) investigated the importance of ‘guided’ proposal distributions in online phylogenetic tree inference. In addition, Everitt et al. (2016) has explored a combination of reversible jump methods with phylogenetic trees targeting the spaces of varying dimension. Finally, several
SMC algorithms for inference in intractable evolutionary models have been proposed (Hajiaghayi et al., 2014; Smith et al., 2017). Wang et al. (2019) proposed an annealed sequential Monte Carlo with adaptive determination of intermediate target distributions in the general SMC framework (Del Moral et al., 2006).

The rest of this chapter is organized as follows. In Section 5.2, we provide background knowledge for Bayesian phylogenetics and introduce notations. In Section 5.3, we introduce our new algorithm for phylogenetic tree reconstruction. In Section 5.4, we introduce joint estimation of phylogenetic tree and nucleotide substitution parameter in particle Gibbs sampler framework. In Section 5.6 and Section 5.7, we use numerical experiments to show the effectiveness of our new method. We conclude in Section 5.8. Our implementation of the proposed method and experimental setups and results are available upon request.

5.2 Background and notation

In this section, we introduce background, notation, and a review of methods for Bayesian phylogenetic inference followed by reviewing Markov chain Monte Carlo algorithm.

5.2.1 Background

We are interested in the evolutionary history among different species inferred from biological sequences. For example, our observation could be DNA sequences for different species, composed of sequences of the four nucleotides \{A, C, G, T\} (Fig 5.1 (d)). Phylogenetic trees are an acyclic graphs, with a set of vertices and edges. Figure 5.1 displayed several phylogenetic tree examples, including (a) an ultrametric tree; (b) a rooted non-binary tree; (c) an unrooted binary tree. In this article, we emphasize binary tree reconstruction. We are interested in the tree topology and the set of branch lengths.

5.2.2 Notation

We denote our observed sequence data by $y$. Let $X$ be a set of observed taxa, a phylogenetic $X$-tree $t$ represents the relationship among observed taxa via a tree topology and a set of branch lengths. We let $\theta$ denote the parameter in a nucleotide substitution model. The prior distribution of $\theta$
and \( t \) are denoted by \( p(\theta) \) and \( p(t|\theta) \) respectively. The likelihood of data \( y \) given \( t \) and \( \theta \) is denoted by \( p(y|t, \theta) \). The joint posterior of \( t \) and \( \theta \) is denoted by \( \pi(t, \theta) \), the posteriors of \( t|\theta \) and \( \theta|t \) are denoted by \( \pi(t|\theta) \) and \( \pi(\theta|t) \). We introduce the notation \( \gamma(\cdot) \) to denote the unnormalized version of \( \pi(\cdot) \). We list all notations at the end of this article.

### 5.2.3 Bayesian phylogenetic inference

In Bayesian phylogenetics, our objective is to estimate the posterior distribution of \( t \) and \( \theta \),

\[
\pi(\theta, t) = \frac{p(y|\theta, t)p(t|\theta)p(\theta)}{p(y)}.
\]

Here \( p(y) = \int \int p(y|\theta, t)p(t|\theta)p(\theta) \, d\theta \, dt \) is the normalizing constant.

With site independence assumption, the likelihood function \( p(y|\theta, t) \) can be evaluated by Felsenstein pruning (Felsenstein, 1973, 1981), involving the calculation of the probability of nucleotide mutation given a fixed amount of evolution (i.e. the branch length). We use a continuous-time Markov chain (CTMC) to model the evolution of each site along each branch of \( t \). There is rich literature about phylogenetic nucleotide substitution models, such as Jukes-Cantor (JC) model (Jukes et al., 1969), the Kimura 2-parameter (K2P) model (Kimura, 1980) and the general time reversible (GTR) model (Rodriguez et al., 1990). The evolutionary model considered in this article is the K2P model. The rate matrix of the CTMC for K2P model is

\[
Q = \begin{pmatrix}
-\pi_C & \kappa \pi_G & \pi_T \\
\pi_A & -\pi_G & \kappa \pi_T \\
\kappa \pi_A & \pi_C & -\pi_T \\
\pi_A & \kappa \pi_C & \pi_C
\end{pmatrix}.
\]

Here the stationary state frequencies \( \pi_A = \pi_C = \pi_G = \pi_T = 0.25 \), and \( \kappa \) represents the ratio of transition to transversion, which is the only parameter in the K2P evolutionary model \( \theta = \kappa \).

A common assumption in Bayesian phylogenetics is that the priors for \( \theta \) and \( t \) are independent, i.e. \( p(t|\theta) = p(t) \). A common prior over non-clock trees consists of a uniform distribution over topologies and a product of independent exponential distributions with rate \( \lambda \) over the branch lengths. A commonly used prior for parameter \( \kappa \) in K2P model is an exponential distribution with rate \( \mu_0 \).

The exact evaluation of the normalized posterior \( \pi(\theta, t) \) requires computing the normalizing constant \( p(y) \), which is generally intractable in phylogenetics.

### 5.2.4 Markov chain Monte Carlo

Markov chain Monte Carlo algorithms are typically used to approximate \( \pi(t, \theta) \) when we are able to evaluate \( \gamma(t, \theta) \) pointwise. Under weak conditions, Markov chain Monte Carlo is guaranteed to converge to the posterior distribution asymptotically. We sample from \( \pi(t|\theta) \) and \( \pi(\theta|t) \) iteratively until convergence is achieved.
Since $\pi(\theta|t)$ does not admit a closed form, we use one Metropolis-Hastings step to sample $\theta$. We let $q(\cdot|\theta)$ denote the proposal distribution given $\theta$, and sample $\theta^* \sim q(\cdot|\theta)$. The MH ratio for accepting a newly proposed $\theta^*$ is

$$
\alpha(\theta \rightarrow \theta^*|t) = \min \left(1, \frac{p(y|\theta^*, t)p(\theta^*) q(\theta|\theta^*)}{p(y|\theta, t)p(\theta) q(\theta^*|\theta)} \right).
$$

(2)

Algorithm 15 provides a detailed description of the MH algorithm for $\theta$.

Algorithm 15 The accept-reject Metropolis Hasting algorithm for $\theta$

1: **Inputs:** (a) Prior over evolutionary parameter $p(\theta)$; (b) Likelihood function $p(y|t, \theta)$; (c) One phylogenetic tree sample $t$; (d) MH proposal $q(\cdot|\theta)$.
2: **Outputs:** $\theta^* \sim \pi(\theta|t)$.
3: Propose a new evolutionary parameter, $\theta^* \sim q(\cdot|\theta)$.
4: Compute the Metropolis-Hastings ratio $\alpha(\theta \rightarrow \theta^*|t)$ according to Eq. (2).
5: Simulate $u \sim U(0, 1)$.
6: if $u < \alpha(\theta \rightarrow \theta^*|t)$ then
7: $\theta^* = \theta^*$.
8: else
9: $\theta^* = \theta$.

We also use MH algorithm to sample new trees $t$. The MCMC proposals for $t$ involve the multiplicative branch proposal (Lakner et al., 2008), the stochastic nearest neighbor interchange (NNI) proposal (Jow et al., 2002) and subtree prune and regraft (SPR) move (Allen and Steel, 2001). Given evolutionary parameter $\theta$, the MH ratio for accepting a newly proposed $t^*$ is

$$
\alpha(t \rightarrow t^*|\theta) = \min \left(1, \frac{p(y|\theta, t^*)p(t^*) q(\theta|\theta^*)}{p(y|\theta, t)p(\theta) q(t^*|\theta)} \right).
$$

(3)

In this article, we consider the K2P nucleotide substitution model, in which the only unknown parameter is $\kappa$. A commonly used prior for parameter $\kappa$ is $\kappa \sim \text{Exp}(\mu_0)$, and the proposal $q(\kappa^*|\kappa) = mk$. Here $\mu_0$ is a hyper-parameter in exponential prior distribution, and $m \sim \text{Unif}(\frac{1}{a}, a)$ with a prefixed tuning parameter $a > 0$. The acceptance probability $\alpha$ for MH algorithm can be simplified as

$$
\alpha(\kappa \rightarrow \kappa^*|t) = \min \left(1, \frac{p(y|\kappa^*, t)}{p(y|\kappa, t)} \cdot e^{\text{max}(1-m)} \cdot m \right).
$$

5.2.5 Limitations of MCMC

There are several drawbacks for MCMC. First, the posterior distribution of phylogenetic tree is usually a complex and multimodal distribution (Lakner et al., 2008). It is challenging to design a very efficient MCMC with good mixing. Often only small moves are allowed at every MCMC iteration as large moves may lead to a high rejection rate. Second, the rapid development of sequencing technology makes large data sets more accessible. The evaluation of the unnormalized posterior for large sequencing data sets is expensive, and MCMC requires long runs to guarantee convergence and
Figure 5.2: (a) overcounting issue in CSMC; (b) overcounting issue in revert-merge CSMC; (c) overcounting issue in the revert step of revert-merge CSMC.

thus is not very scalable to large sequence data sets. Third, in recent decades, little improvement has been made in typical MCMC proposals for phylogenetic inference.

5.3 Phylogenetic tree inference

In this section, we assume that the parameter $\theta$ in the nucleotide substitution model in known. Our interest is on the posterior inference over phylogenetic trees $\pi(t)$. We first review combinatorial sequential Monte Carlo, and then we our new approach.

5.3.1 Combinatorial sequential Monte Carlo

Combinatorial sequential Monte Carlo (Wang et al., 2015) is an SMC algorithm for general tree inference based on a graded partially ordered set on an extended combinatorial space. The essential idea of the CSMC algorithm is to introduce a sequence of $R$ intermediate states to construct the target phylogenetic tree $t$ incrementally. These $R$ intermediate states are typically graded from ‘simple’ to ‘complex’. The CSMC algorithm sequentially approximates these intermediate distributions efficiently. The last intermediate distribution is the posterior of tree $\pi(t)$.

We use $s_1, s_2, \ldots, s_R$ to denote the sequence of intermediate states. We call state $s_r$ a partial state of rank $r$. For example, a partial state of rank $r$ can be a phylogenetic forest over the observed taxa, defined as a set of $R - r + 1$ phylogenetic trees. We use the notation $S_r$ to denote the set of partial states of rank $r$, and define $S = \bigcup S_r$. We use the notation $|s|$ to denote the number of trees in a forest $s$. Recall that our interest is in inferring $\pi(t)$ with $\gamma(t)$. Here $\gamma(t)$ denotes an unnormalized version of $\pi(t)$. A natural extension for $\gamma$ into forests is to take a product over the trees in the forest $s$ as follows:

$$\gamma(s) = \prod_{(t_i, X_i) \in s} \gamma_{\gamma(X_i)}(t_i). \quad (4)$$

where $t_i$ denotes one subtree $i$ in forest $s$, $X_i$ denotes the taxa of $t_i$ and $\gamma(X_i)$ denotes the data associated with $X_i$. 75
CSMC algorithms iterate between resampling, propagation and re-weighting to propose samples of rank \( r \) from samples of rank \( r - 1 \). We let \( s_{r-1,k} \) denote the \( k \)-th particle for rank \( r - 1 \) after propagation. First, we resample \( K \) times from the empirical posterior \( \pi_{r-1}(s) = \sum_{k=1}^{K} W_{r-1,k} \delta_{s_{r-1,k}}(s) \) and denote the particles after resampling by \( \tilde{s}_{r-1,k} \), \( k = 1, 2, \ldots, K \). Second, we propose new samples through sampling a pair of subtrees to merge (i.e. pick the pair of trees in \( \tilde{s}_{r-1} \) uniformly at random among the \( \binom{|\tilde{s}_{r-1}|}{2} \) pairs) and sample the length of a new branch. Finally, we compute a weight update function for each particle with the following formula

\[
    w_{r,k} = w(\tilde{s}_{r-1,k}, s_{r,k}) = \frac{\gamma(s_{r,k})}{\gamma(\tilde{s}_{r-1,k})} \cdot \frac{v^-_{r,k}(\tilde{s}_{r-1,k})}{v^+_{r-1,k}(s_{r,k})},
\]

where \( \nu^- \) is a backward kernel to correct an overcounting problem in non-clock tree inference, \( \nu^+ \) is the forward kernel. Figure 5.2 (a) displays the overcounting issue for CSMC in non-clock tree inference. As shown in the figure, one intermediate state \( S_r \) may have multiple ancestors \( S_{r-1} \), which may lead to an inconsistent estimator. Wang et al. (2015) compared CSMC with MCMC, and showed that CSMC can be orders of magnitude faster than MCMC in terms of converge to the tree posterior.

As we will show later in numerical experiments, CSMC is not efficient in PG frameworks for estimating nucleotide substitution parameters and for inferring phylogenetic trees. The main reason for this is the restricted proposal distribution. Figure 5.3 shows an example illustrating the constraints. If a clade \((A, B)\) exists in partial state \( \tilde{s}_{r-1} \), we cannot propagate samples of partial states \( s_r \) without this clade. This motivates us to propose improvements to CSMC.

### 5.3.2 Revert-merge combinatorial sequential Monte Carlo

We propose a revert-merge CSMC, with a more sophisticated proposal to explore the tree space. The proposed algorithm is built upon the naive version of CSMC framework. In revert-merge CSMC, we introduce \( R \) intermediate target distributions. The same setups and notations are used in section **Combinatorial sequential Monte Carlo**.

Figure 5.4 presents an overview of the revert-merge CSMC algorithmic framework. The revert-merge CSMC algorithm sequentially approximates \( \pi(s_r) (r = 2, 3, \ldots, R) \). The algorithm is initialized at rank \( r = 1 \) by initializing the list with \( K \) copies of the least partial state \( s_1 \) (a list of taxa). Given a
Figure 5.4: An overview of the revert-merge CSMC framework. A set of partial states is kept at each SMC iteration. A positive-valued weight is associated with each partial state. The algorithm iterates the following three steps: (i) resample to prune particles with small weights, (ii) propose a new partial state through the revert and merge proposal, and (iii) compute the weights for new particles.
list of weighted particles of the partial state $s_{r-1}$, the revert-merge CSMC algorithm performs the following three steps to approximate $\pi(s_r)$: resampling, propagation and re-weighting.

**Resampling:** First, we conduct a resampling step to resample $K$ particles from the empirical distribution $\pi_{r-1}(s) = \sum_{k=1}^{K} W_{r-1,k} \delta_{s_{r-1,k}}(s)$ and denote the resampled particles by $\tilde{s}_{r-1,1}, \tilde{s}_{r-1,2}, \ldots, \tilde{s}_{r-1,K}$. The resampling step prunes particles with low weights. It is well known that SMC algorithms suffer from the path degeneracy issue (Doucet et al., 2001). This issue occurs when the discrepancy between two successive distributions of partial states is large. Conducting a resampling step can help alleviate this issue. For example, we could use multinomial resampling, or more advanced resampling schemes such as stratified resampling or residual resampling (Douc and Cappé, 2005).

A list of equally weighted samples is obtained after performing the resampling step. To make the algorithm more efficient, we only resample when the relative effective sample size (rESS) falls below a threshold. The rESS is defined as

$$\text{rESS}(W_r) = \left( K \sum_{k=1}^{K} W_k^2 \right)^{-1},$$

which represents the percent of ‘perfect’ samples in the approximated distribution. The intuition that we do not perform resampling at each iteration $r$ is that particles with small weights at the current iteration may become important particles later. The resampling step increases the asymptotic variance of SMC estimators (Chopin, 2002). Hence, resampling triggered by rESS is preferable.

**Propagation:** Second, we propagate a new particle of rank $r$, denoted by $s_{r,k}$ from each of the resampled particle $\tilde{s}_{r-1,k}$ ($\tilde{s}_{r-1,k} = s_{r-1,k}$ if we do not conduct resampling at rank $r-1$), using a proposal distribution $\nu_{\tilde{s}_{r-1,k}}: S \to [0,1]$. The probability of proposing a new state $s_{r,k}$ from the current state $\tilde{s}_{r-1,k}$ is denoted by $\nu_{\tilde{s}_{r-1,k}}(s_{r,k})$. There are three steps in propagating $s_{r,k}$:

1. Find the reverted state of the current partial state $\tilde{s}_{r-1,k}$, denoted by $\varrho(\tilde{s}_{r-1,k})$.
2. We first pick a pair of trees in $\varrho(\tilde{s}_{r-1,k})$ uniformly at random among the $\left\lfloor \frac{|\varrho(\tilde{s}_{r-1,k})|}{2} \right\rfloor$ pairs, then sample the length of the new branch. This state is denoted by $\int(\tilde{s}_{r-1,k})$.
3. We pick a pair of trees in $\int(\tilde{s}_{r-1,k})$ uniformly at random among the $\left\lfloor \frac{|\int(\tilde{s}_{r-1,k})|}{2} \right\rfloor$ pairs, and sample the length of the new branch. This step finishes the propagation of $s_{r,k}$.

We define $\tilde{s}_{r-1,k}$ to be the parent state of $s_{r,k}$, and define $\varrho(\tilde{s}_{r-1,k})$ to be the reverted state of $\tilde{s}_{r-1,k}$. There is no strict definition for ‘parent’ state or ‘reverted’ state. We define in this way just for the purpose of differentiating the two states.
Figure 5.5 displays an example of proposing \(s_{r,k}\) from \(\tilde{s}_{r-1,k}\) using the revert-merge proposal. In Step 1, another possible reverted state other than its real reverted state could be generated through the same topology. We save the information about how the current state was generated, and so it can only have one reverted state.

**Re-weighting:** Finally, we compute a weight for each of these new particles. The weight update function for the proposed revert-merge CSMC is

\[
w_{r,k} = w(\tilde{s}_{r-1,k}, s_{r,k}) = \frac{\gamma(s_{r,k})}{\gamma(\tilde{s}_{r-1,k})} \cdot \frac{\nu_{s_{r,k}}^- (\tilde{s}_{r-1,k})}{\nu_{s_{r-1,k}}^+ (s_{r,k})}.
\]

where \(\nu^-\) is a backward kernel to correct an overcounting problem, and \(\nu^+\) is a forward kernel. The normalized version of weight function is represented by \(W_{r,k}\). Figure 5.2 (b) illustrates the overcounting issue in revert-merge CSMC. The computation of the backward kernel and the forward kernel in revert-merge CSMC is more complicated than the naive version of CSMC, since there are three steps in the revert and merge proposal. To simplify the calculation, we also decompose the forward and backward kernels into three steps.

\[
\begin{align*}
\nu_{s_{r,k}}^+(s_{r,k}) &= \nu_{s_{r-1,k}}^+(q(\tilde{s}_{r-1,k})) \cdot \nu_{\tilde{s}_{r-1,k}}^+ (f(\tilde{s}_{r-1,k})) \cdot \nu_{\tilde{s}_{r-1,k}}^+ (s_{r,k}), \\
\nu_{s_{r,k}}^- (\tilde{s}_{r-1,k}) &= \nu_{s_{r-1,k}}^- (f(\tilde{s}_{r-1,k})) \cdot \nu_{s_{r-1,k}}^- (q(\tilde{s}_{r-1,k})) \cdot \nu_{\tilde{s}_{r-1,k}}^- (\tilde{s}_{r-1,k}).
\end{align*}
\]

The weight update function can be rewritten as

\[w(\tilde{s}_{r-1,k}, s_{r,k}) = w(\tilde{s}_{r-1,k}, q(\tilde{s}_{r-1,k})) \cdot w(\tilde{s}_{r-1,k}, f(\tilde{s}_{r-1,k})) \cdot w(f(\tilde{s}_{r-1,k}), s_{r,k}).\]

where

\[
\begin{align*}
w(\tilde{s}_{r-1,k}, q(\tilde{s}_{r-1,k})) &= \frac{\gamma(q(\tilde{s}_{r-1,k}))}{\gamma(\tilde{s}_{r-1,k})} \cdot \frac{\nu_{q(\tilde{s}_{r-1,k})}^+ (\tilde{s}_{r-1,k})}{\nu_{\tilde{s}_{r-1,k}}^+ (q(\tilde{s}_{r-1,k}))}, \\
w(q(\tilde{s}_{r-1,k}), f(\tilde{s}_{r-1,k})) &= \frac{\gamma(f(\tilde{s}_{r-1,k}))}{\gamma(q(\tilde{s}_{r-1,k}))} \cdot \frac{\nu_{f(\tilde{s}_{r-1,k})}^- (\tilde{s}_{r-1,k})}{\nu_{f(\tilde{s}_{r-1,k})}^- (q(\tilde{s}_{r-1,k}))}, \\
w(f(\tilde{s}_{r-1,k}), s_{r,k}) &= \frac{\gamma(s_{r,k})}{\gamma(f(\tilde{s}_{r-1,k}))} \cdot \frac{\nu_{s_{r,k}}^- (f(\tilde{s}_{r-1,k}))}{\nu_{f(\tilde{s}_{r-1,k})}^- (s_{r,k})}.
\end{align*}
\]

In order to evaluate \(w(\tilde{s}_{r-1,k}, s_{r,k})\), we compute \(w(\tilde{s}_{r-1,k}, q(\tilde{s}_{r-1,k}))\), \(w(q(\tilde{s}_{r-1,k}), f(\tilde{s}_{r-1,k}))\) and \(w(f(\tilde{s}_{r-1,k}), s_{r,k})\) respectively. The details for the selection the forward \((\nu^+)\) and the backward \((\nu^-)\) kernels, and the concrete forms of the weight update function will be described in section A phylogenetic tree example using revert-merge CSMC. If resampling is not conducted at rank \(r-1\), the weight update function admits the following form

\[
w_{r,k} = W_{r-1,k} \cdot \frac{\gamma(s_{r,k})}{\gamma(\tilde{s}_{r-1,k})} \cdot \frac{\nu_{s_{r,k}}^- (\tilde{s}_{r-1,k})}{\nu_{s_{r-1,k}}^- (s_{r,k})}.
\]

79
We construct a discrete positive measure using a list of weighted particles at rank \( r \),

\[
\pi_{r,K}(s) = \sum_{k=1}^{K} W_{r,k} \delta_{s_{r,k}}(s), \quad \text{for all } s \in S,
\]

where \( \delta_s \) is the Kronecker delta function. In the end, we obtain a Monte Carlo approximation \( \pi_{R,K} \) of \( \pi(t) \). Algorithm 16 describes the revert-merge CSMC algorithm.

**Assumption 1.** For all \( s, s' \in S \), \( \nu^+_s(s') = 0 \) implies \( \nu^-_s(s') = 0 \).

**Proposition 1.** Under assumption 1, the revert-merge CSMC provides asymptotically consistent estimates. We have

\[
\sum_{k=1}^{K} W_{R,k} \phi(s_{R,k}) \rightarrow \int \pi_{R,K}(s) \phi(s) \, ds \quad \text{as} \quad K \to \infty,
\]

where the convergence is in \( L^2 \) norm, \( \phi \) is a target function under mild conditions. For example, \( \phi \) is a bounded function. We refer readers to Wang et al. (2015) for the proof of this proposition.

---

**Algorithm 16 : Revert-merge combinatorial sequential Monte Carlo**

1. **Inputs:** (a) Prior over partial states \( p(s) \); (b) Likelihood function \( p(y|s, \theta) \).
2. **Outputs:** Approximation of the posterior distribution, \( \sum_{k=1}^{K} W_{r,k} \delta_{s_{r,k}}(\cdot) \approx \pi(\cdot) \).
3. Initialize SMC iteration index: \( r \leftarrow 1 \).
4. **for** \( k \in \{1, \ldots, K\} \) **do**
   5. Initialize particles with the least partial state.
   6. Initialize weights to unity: \( W_{1,k} \leftarrow 1 \).
7. **for** rank \( r \in [2, \ldots, R] \) **do**
   8. if \( r \text{ESS}(W_{r-1,\cdot}) < \epsilon \) **then**
      9. Resample the particles.
      10. **for** \( k \in \{1, \ldots, K\} \) **do**
        11. Reset particle weights: \( \tilde{W}_{r-1,k} = 1/K \).
   12. **else**
      13. **for** \( k \in \{1, \ldots, K\} \) **do**
      14. \( \tilde{W}_{r-1,k} = W_{r-1,k}, \tilde{s}_{r-1,k} = s_{r-1,k} \).
   15. **for** all \( k \in \{1, \ldots, K\} \) **do**
      16. Sample particles \( s_{r,k} \sim \nu^+_{s_{r-1,k}}(\cdot) \), using one revert move to find \( \varphi(\tilde{s}_{r-1,k}) \), and two merge moves to propose \( \int (\tilde{s}_{r-1,k}) \) and \( s_{r,k} \).
      17. Compute pre-resampling normalized weights:
      \[
      W_{r,k} \propto \tilde{W}_{r-1,k} \cdot \frac{\gamma(s_{r,k}) \cdot \nu^-_{s_{r-1,k}}(\tilde{s}_{r-1,k})}{\gamma(\tilde{s}_{r-1,k}) \cdot \nu^+_{s_{r-1,k}}(s_{r,k})}.
      \]
18. **Construct**
   \[
   \pi_{r,K}(s) = \frac{1}{K} \sum_{k=1}^{K} W_{r,k} \delta_{s_{r,k}}(s).
   \]

Revert-merge CSMC uses a more sophisticated proposal to explore tree space. This provides a benefit to the algorithm as it can more easily explore the tree posterior. At the same time, it leads
to a higher computational cost. In the naive version of CSMC, there is only one step in proposing particles for new partial states and computing the incremental weight function, while in the revert-merge CSMC method, there are three steps in propagating new state and re-weighting. We store the information of the reverted states to reduce the computational cost.

5.3.3 A phylogenetic tree example using revert-merge CSMC.

In this section, we construct a detailed example of our proposed CSMC algorithm for estimating ultrametric trees with continuous branch lengths. We introduce the proposal distribution, and the weighting function, including the forward kernel \( \nu^+ \) and backward kernel \( \nu^- \). We let \( s \) denote the current partial state. We first use one revert move to find the reverted state of \( s \), \( \varrho(s) \). This step is deterministic, hence \( \nu^+(\varrho(s)) = 1 \). The overcounting issue for this step originates from two sources, tree topology and branch length (refers to Figure 5.2 (c)). The number of possible reverted states for \( \varrho(s) \) is the number of possible pairs of trees in \( \varrho(s) \). If we use \( \Delta h^- \) to denote the incremental height of the forest we revert, the backward probability is

\[
\nu^-(\varrho(s)) = \left( \frac{|\varrho(s)| \times 2}{2} \right)^{-1} \left( \frac{|\varrho(s)|}{2} \right) \cdot \exp(-\lambda \Delta h^-),
\]

where \( \lambda \) denotes the rate of the exponential distribution that proposed the incremental height of forests (\( \Delta h^- \)) in the second merge step of the previous partial state. In the revert step, we assume that the subtree being split is \( (t_m, X_m), (t_1, X_1) \) and \( (t_2, X_2) \) are subtrees is split to form \( (t_m, X_m), (t_1, X_1), (t_2, X_2) \),

\[
\varrho(s) = s \cup \{(t_1, X_1), (t_2, X_2)\} \setminus \{(t_m, X_m)\}.
\]

The weight update function for this step admits the following form

\[
w(s, \varrho(s)) = \frac{p(y(X_1)\theta, t_1^-)p(y(X_2)\theta, t_2^-)}{p(y(X_m)\theta, t_m^-)}.
\]

Then we apply two merge steps to propagate a new partial state \( s' \). In each merge step, we randomly choose a pair of subtrees to merge and strictly increasing the height of the forest. The overcounting issue in merge steps is resolved by the assumption of strict increase in height. In the first merge step, we assume that the subtrees being merged are \( (t_1, X_1) \) and \( (t_2, X_2) \), and that the merged subtree connecting \( (t_1, X_1) \) and \( (t_2, X_2) \) is \( (t_m, X_m) \).

\[
\bar{f}(s) = \varrho(s) \cup \{(t_m, X_m)\} \setminus \{(t_1, X_1), (t_2, X_2)\}.
\]

The increment of the height of the forest (\( \Delta h \)) is proposed from an exponential distribution with rate \( \lambda_{\varrho(s)} \) (the prior distribution). The density of the proposal \( \nu^+ \) is

\[
\nu^+_{\varrho(s)}(\bar{f}(s)) = \left( \frac{|\varrho(s)| \times 2}{2} \right)^{-1} \left( \frac{|\varrho(s)|}{2} \right) \cdot \exp(-\lambda_{\varrho(s)} \Delta h). \]
The weight update function for the first merge step can be simplified as

\[ w(g(s), \hat{f}(s)) = \frac{p(y(X_m)|\theta, t_m)}{p(y(X_1)|\theta, t_1)p(y(X_2)|\theta, t_2)}. \]

The proposed state \( \hat{f}(s) \) is a sibling (or twin) of the state \( s \). In step 3, we assume \((t^*_m, X^*_m)\) is the merged subtrees connecting \((t^*_1, X^*_1)\) and \((t^*_2, X^*_2)\).

\[ s' = \hat{f}(s) \cup \{(t^*_m, X^*_m)\}\backslash\{(t^*_1, X^*_1), (t^*_2, X^*_2)\}. \]

The increment of the height of the forest is also sampled from an exponential distribution with rate \( \lambda_{\hat{f}(s)} \) (the prior distribution). The incremental weight function for the two merge steps can be written as

\[ w(\hat{f}(s), s') = \frac{p(y(X^*_m)|\theta, t^*_m)}{p(y(X^*_1)|\theta, t^*_1)p(y(X^*_2)|\theta, t^*_2)}. \]

We describe a non-clock tree example in Appendix 1.

5.4 Joint estimation of phylogenetic tree and evolutionary parameter

5.4.1 Particle Gibbs Sampler

We have introduced posterior inference over phylogenetic tree spaces, using the revert-merge CSMC algorithm, assuming the parameter \( \theta \) is known. In a more realistic scenario, \( \theta \) is also an unknown parameter, which requires us to estimate \( \theta \) given data.

We study the particle Gibbs sampler, a Gibbs-type algorithm that iterates between sampling \( t \) and \( \theta \). Given a tree \( t \), we use one Metropolis-Hastings step to sample \( \theta \) from \( \pi(\theta) \) (see Algorithm 15). Given \( \theta \), a conditional revert-merge CSMC algorithm is used to approximate \( \pi(t|\theta) \). The conditional revert-merge CSMC is described in Algorithm 19 of Appendix 2. The main difference between the revert-merge CSMC algorithm and the conditional version is that in the latter, one of the particle trajectories is pre-specified, which is called the reference trajectory. This reference trajectory cannot be pruned in the resampling step. The resulting Markov kernel of PGs will leave the target distribution invariant for an arbitrary number of particles used in the conditional revert-merge CSMC. We apply the conditional revert-merge CSMC (Algorithm 19) in PGs scheme to estimate \( \theta \) and \( t \), summarized in Algorithm 17. Without loss of generality, we assume the first particle trajectory \( s_{1,R,1} \) to be the reference trajectory, denoted by \( s^*_{1,R} \). In PGs, we first use one Gibbs step to update the parameter \( \theta \), and then conditional on this \( \theta \), we sample a particle trajectory from the approximated posterior of phylogenetic forests by running the conditional revert-merge CSMC. This sampled trajectory will be the reference trajectory of the conditional revert-merge CSMC in the next particle Gibbs iteration. We iterate these two steps until the convergence is achieved.
We sample a

Algorithm 17: Particle Gibbs sampler
1: **Inputs:** (a) Prior over partial states \( p(s) \) and evolutionary parameter \( p(\theta) \); (b) Likelihood function \( p(y|s, \theta) \); (c) One pre-specified particle trajectory for partial states \( s_{1:R}^* \); (d) MH proposal for \( \theta \), \( q(\cdot|\theta) \).
2: **Outputs:** Approximated posterior distribution \( \tilde{p}(\theta, t) \).
3: Initialize MCMC iteration index \( i \leftarrow 0 \).
4: Initialize \( \theta(0) \) to an arbitrary value.
5: Initialize the reference trajectory of the conditional revert-merge CSMC (Algorithm 19) to \( s_{1:R}^* \). (Without generality we could set \( s_{1:R,1}(0) = s_{1:R,1}^* \).)
6: Run the conditional revert-merge CSMC algorithm targeting \( \pi(t|\theta(0)) \), sample \( s_{1:R}^* \sim \tilde{p}(\cdot|\theta(0)) \).
7: for \( i = 1, 2, \ldots, N \) do
8: Sample \( \theta(i) \sim p(\theta|y, t = s_{1:R}^*) \) using the accept-reject Metropolis Hastings algorithm.
9: Update the reference trajectory to \( s_{1:R}^* \) and run the conditional revert-merge CSMC algorithm (Algorithm 19) targeting \( \pi(t|\theta(i)) \), sample \( s_{1:R}^* \sim \tilde{p}(\cdot|\theta(i)) \).
10: Burn in a proportion (e.g. 50%) of the PGs chain.

5.4.2 Particle Gibbs Sampler with Ancestor Sampling

In particle Gibbs, the reference trajectory is kept intact throughout the revert-merge CSMC algorithm. This may lead to slow mixing of PGs algorithm when path-degeneracy exists, even though the Markov kernel leaves the target distribution invariant.

In this section, we investigate particle Gibbs with ancestor sampling (PGAS) (Lindsten et al., 2014) to improve the mixing of particle Gibbs samplers. The basic idea of PGAS is to include an ancestor sampling step in the conditional revert-merge CSMC algorithm to update the reference trajectory. The path degeneracy issue is inevitable in revert-merge CSMC. If the reference trajectory is updated though the ancestor sampling step, the particle system will degenerate to a new trajectory other than the reference trajectory.

We turn to the implementation of the ancestor sampling step. Without loss of generality, we let \( s_{1:R,1} \) be the reference trajectory. For rank \( r \geq 3 \), we consider the part of the reference trajectory \( s_{r:R,1} \) ranging from the current partial state \( r \) to the final state \( R \). The objective is to connect the partial reference trajectory \( s_{r:R,1} \) to one of the historical trajectories \( \{s_{1:r-1,i}\}_{i=1}^N \). We use notation \( a_r \in \{1, 2, \ldots, N\} \) to denote the index of ancestor sampling trajectory for partial state of rank \( r \).

We assign each index \( a_r \in \{1, 2, \ldots, N\} \) a weight function, which admits the form

\[
P(a_r = k) \propto w_{r-1,k} \cdot \frac{\gamma(s_{1:r-1,k}, s_{r:R,1})}{\gamma(s_{1:r-1,k})} \propto w_{r-1,k} \cdot \frac{\gamma(s_{r-1,k}, s_r)}{\gamma(s_{r-1,k})}.
\]

We sample \( a_r \) with probability \( P(a_r = k) \). The invariance property for the kernel of PGAS still holds. We described the PGAS in Algorithm 18. The following assumption describes conditioning on \( P(a_r = k) \).

**Assumption 2.** For all \( s_{r-1,k}, s_r, \in S, \nu_{s_{r-1,k}}^+(s_r) = 0 \) implies \( P(a_r = k) = 0 \).
We use an example to illustrate the benefits of using the proposed revert-merge CSMC algorithm in the ancestor sampling step. In Figure 5.6, the trajectory in grey color represents the reference trajectory. To sample the ancestor of the reference trajectory at rank \( r = 3 \), we evaluate \( P(a_r = k) \) for \( k = 1, \ldots, K \). Using assumption 2, \( P(a_3 = 1) = 0 \) and \( P(a_3 = 3) = 0 \) in PGAS with naive version of CSMC as \( \nu^r_{s_{1,1}}(s_{2,2}) = 0 \) and \( \nu^r_{s_{1,3}}(s_{2,2}) = 0 \). A new partial state can only be proposed by randomly choosing a pair of subtrees to combine. Hence, \( s_r \) can connect to \( s_{r-1} \) if and only if \( s_{r-1} \) is the parent of \( s_r \). As a result, the sampled ancestor is still the partial reference trajectory \( s_{1:r-1} \). If the revert-merge proposal is used in PGAS, \( P(a_3 = 1) > 0 \) and \( P(a_3 = 3) > 0 \).

### 5.5 Interacting particle Markov chain Monte Carlo

Another particle Gibbs type algorithm, interacting particle Markov chain Monte Carlo (IPMCMC) (Rainforth et al., 2016), is considered to improve the mixing of a particle Gibbs sampler. In IPMCMC, a pool of standard and conditional revert-merge CSMC algorithms are interacted to design an efficient proposal for tree posterior. The interaction of conditional and standard revert-merge CSMCs is achieved by communicating their marginalized likelihoods. The algorithmic description of IPMCMC is displayed in Appendix 3. The nodes of standard and conditional revert-merge CSMC can be allocated into different cores to achieve parallelization.

### 5.6 Simulation studies

In this section, we evaluate the quality of tree reconstruction using the revert-merge CSMC algorithm. We generate ultrametric trees assuming the branch length is exponentially distributed with rate \( 10 \cdot (|s_r|/2) \), where \( s_r \) denotes partial states with rank \( r \). Given the simulated tree, we simulate
Algorithm 18: Conditional revert-merge CSMC with ancestor sampling

1: **Inputs:** (a) Prior over partial states $p(s)$; (b) Likelihood function $p(y|s, \theta)$; (c) One pre-specified particle trajectory for partial states $s_{1,R}^*$.  
2: **Outputs:** Approximation of the posterior distribution, $\sum_k W_{R,k} \delta_{s_{R,k}}(\cdot) \approx \pi(\cdot)$.  
3: Initialize SMC iteration index: $r \leftarrow 1$.  
4: for $r \in \{1, \ldots, R\}$ do  
5: Initialize $s_{1,r} = s_{1,R}^*$.  
6: for $k \in \{1, \ldots, K\}$ do  
7: Initialize particles with the least partial state $\bot$.  
8: Initialize weights to unity: $W_{1,k} \leftarrow 1/K$.  
9: for $r \in \{2, \ldots, R\}$ do  
10: if rESS($W_{r-1,\cdot}$) $< \epsilon$ then  
11: Resample the particles with index $k = 2, 3, \ldots, K$. Note that we do not resample particles in the reference trajectory.  
12: for $k \in \{1, \ldots, K\}$ do  
13: Reset particle weights: $\tilde{W}_{r-1,k} = 1/K$.  
14: else  
15: for $k \in \{1, \ldots, K\}$ do  
16: $W_{r-1,k} = W_{r-1,k}; \tilde{s}_{r-1,k} = s_{r-1,k}$.  
17: if $r \geq 3$ then  
18: for $k \in \{1, \ldots, K\}$ do  
19: Compute the probability of connecting the partial reference trajectory $s_{r,R,1}$ to one of the history trajectories $s_{1:r-1,k}$,  
20: We sample ancestor index $a_r = k$ with probability $P(a_r = k)$.  
21: for $k \in \{1, \ldots, K\}$ do  
22: if $k > 1$ then  
23: Sample particles $s_{r,k} \sim \nu_{\tilde{s}_{r-1,k}}^-(\cdot)$, using one revert move to find $q(\tilde{s}_{r-1,k})$, and two merge moves to propose $\tilde{f}(\tilde{s}_{r-1,k})$ and $s_{r,k}$.  
24: Compute pre-resampling normalized weights:  
25: Construct  
26: Construct

$$
\pi_{r,K}(s) = \frac{1}{K} \sum_{k=1}^K W_{r,k} \delta_{s_{r,k}}(s).
$$
DNA sequences using the K2P model with parameter $\theta = 2.0$. Site independence is assumed. We use *majority-rule consensus tree* to summarize the weighted samples of phylogenetic trees (Felsenstein, 1981). We mainly use the normalized version of the Robinson Foulds (RF) metric (Robinson and Foulds, 1979), and the partition metric (PM), also known as the symmetric difference or topology only RF metric (Robinson and Foulds, 1981) to measure the quality of the estimated tree. Stratified resampling is conducted to resample particles when the rESS falls below 0.5.

### 5.6.1 Comparison of CSMC and Revert-Merge CSMC

In this study, we first emphasize a comparison of the naive version of CSMC and the proposed revert-merge CSMC in terms of computational speed. We also illustrate the computational gains from parallelization of CSMC algorithms. We simulated 50 ultrametric trees, each with 20 leaves, with 1000 nucleotides on each leaf. We ran two CSMC algorithms using 10000 particles on all data sets, varying the number of threads. Table 6.2 indicates that the computational speed of revert-merge CSMC is lower than that of the naive version of CSMC. This is expected as the naive version of CSMC proposal is simpler, while in our new proposal we have to use one move to find the reverted state and merge twice to propose the new partial state. The weight update function in revert-merge CSMC algorithm is also more complicated. When we increase the number of threads, the computing time of both methods decreases. As we have alluded earlier MCMC algorithms are not very adaptable to large scale sequencing data. The parallelization of the revert-merge CSMC is an important advantage over MCMC. The propagation and weighting steps can be easily parallelized by allocating samples over different cores.

Table 5.1: Runtime (second) of CSMC and CSMC-RM, with mean and 95% confidence intervals, for 1 to 4 threads.

<table>
<thead>
<tr>
<th>#Threads</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSMC</td>
<td>30.42 (30.15, 30.69)</td>
<td>17.94 (17.68, 18.19)</td>
<td>17.52 (17.16, 17.89)</td>
<td>16.36 (16.18, 16.55)</td>
</tr>
<tr>
<td>CSMC-RM</td>
<td>42.73 (42.49, 42.98)</td>
<td>24.55 (24.31, 24.79)</td>
<td>23.67 (23.29, 24.05)</td>
<td>22.07 (21.97, 22.17)</td>
</tr>
</tbody>
</table>

We did another experiment to evaluate the quality of tree reconstruction of two CSMC algorithms. We simulated 15 ultrametric trees, with the number of leaves taken from the set \{10, 20, 40\} (5 trees of each size), and one dataset for each tree. The number of nucleotides on each leaf is 1000. We used the RF metric is used to measure the distance between the estimated consensus tree and the simulated reference tree. In the first experiment, we compared two CSMC algorithms as a function of number of particles. The number of particles for revert-merge CSMC is \{1 \cdot 10^3, 3 \cdot 10^3, 1 \cdot 10^4, 3 \cdot 10^4\}. As we have alluded in the previous experiment, the computational cost of revert-merge CSMC is higher with a fixed particle number. We selected four levels for the number of particles for naive version of CSMC, \{1.5 \cdot 10^3, 4.5 \cdot 10^3, 1.5 \cdot 10^4, 4.5 \cdot 10^4\}, to guarantee the computational budget allocated for two methods are close. For each simulated tree, each algorithm was repeated 10 times with
different random seeds. Figures 5.7 displays the RF metric of the revert-merge CSMC (‘CSMC-RM’ in the figure) and the naive version of CSMC algorithms as a function of the number of particles across different numbers of taxa. The RF metric decreases with incremental numbers of particles. For trees with a small number of taxa (e.g. 10), the tree construction quality of CSMC is better than revert-merge CSMC. For larger trees, the RF metric provided by the revert-merge CSMC is lower than the naive version of CSMC with a fixed computational budget. The gap in the RF metric between two CSMCs widens as the value of taxa increases.

![Figure 5.7: Comparison of CSMC and revert-merge CSMC as a function of number of (1000) particles. The x-axis represents number of a thousand particles. The y-axis represents the normalized RF metric. The four levels from left to right for revert-merge CSMC are $1 \cdot 10^3$, $3 \cdot 10^3$, $1 \cdot 10^4$ and $3 \cdot 10^4$ respectively. The four levels from left to right for naive version of CSMC are $1.5 \cdot 10^3$, $4.5 \cdot 10^4$, $1.5 \cdot 10^4$, $4.5 \cdot 10^4$ respectively. The y-axis represents the Robinson Fouolds metric.](image)

**5.6.2 Comparison of particle Gibbs samplers**

In this section, we focus on joint estimation of evolutionary parameter $\theta$ and phylogenetic tree $t$ in the framework of particle Gibbs sampler.

The computational cost of particle Gibbs samplers is a linear function of the number of particles $K$ and number of MCMC iterations $N$. Increasing both $K$ and $N$ can improve the mixing of particle Gibbs sampling. We first use one experiment to investigate the relative importance of $K$ and $N$ in improving the mixing of PGs with a fixed computational cost. We simulated 1 ultrametric tree with 15 leaves, and one data set with 500 nucleotides for each leaf. The evolutionary model we used to simulate DNA sequences is the K2P model with parameter $\theta = 2.0$. The log-likelihood of majority-rule consensus tree and RF metric are used to measure the distance between the estimated consensus tree and the reference tree we simulated. We fixed the computational budgets at $K \cdot N = 2 \cdot 10^6$, and choose three levels of $(K, R)$: (1000, 2000), (2000, 1000), (4000, 500). We repeat the PGs sampling with revert-merge proposals 10 times. Figure 5.8 displayed the RF metric and log-likelihood of majority-rule consensus tree provided by PGs with revert-merge proposals. With a higher value of $K$ (and lower number of MCMC iterations), the log-likelihood of majority-rule consensus tree increases and the RF metric decreases. This indicates that with a fixed computational cost, high values of $K$ are more important than the number of PGs iterations.
We conduct another experiment to investigate the performance of two CSMCs as efficient designers of tree posterior in PGs and IPMCMC, as a function of the number of particles. We also investigate the performance of PGAS with revert-merge CSMC. The initial reference trajectory $s^*_{1,R}$ is simulated through the proposal distribution of the CSMC algorithm. The initial value of $\theta$ is randomly sampled from $Unif(1, 3)$. We simulated 5 ultrametric trees, with 15 leaves, and one data set for each tree. The number of nucleotides on each leaf is 500. The evolutionary model we used to simulate DNA sequence is the K2P model with parameter $\theta = 2$.0. The log-likelihood of \textit{majority-rule consensus tree} and RF metric are used to measure the distance between the estimated consensus tree and the reference tree we simulated. We set the total number of nodes for running conditional CSMC and CSMC algorithms to be twice as the number of nodes running conditional CSMC ($M = 2P$), and set $M = 4$. The computational cost of IPMCMC is a linear function of $M$. In order to fix computational budgets for IPMCMC and PGs, we set $K$ for PGs and PGAS to be 4 times as large as that of IPMCMC. We also set $K$ for the naive version of CSMC to be three times as large as revert-merge CSMC. The other setups are the same as for the previous experiment. We chose three levels of $K$ for PGs with the revert-merge CSMC proposal: $K = 200, 500, 1000$. The number of MCMC iterations were fixed at 4000 iterations. Figure 5.9 displays the comparison of PGs and IPMCMC with naive version of CSMC (IPGS, PGS) and revert-merge CSMC (IPGS-RM, PGS-RM) as a function of number of particles. We generated another two panels in the second row to provide a better comparison between PGs, IPMCMC and PGAS with revert-merge proposals as the results displayed in the upper panels cannot clearly show the difference between methods. For both PGs and IPMCMC with the naive version of CSMC, the log-likelihood of \textit{majority-rule consensus tree} and RF metric do not improve if we increase $K$. PGs, IPMCMC and PGAS with revert-merge CSMC performs better in terms of log-likelihood and RF metric. If we increased $K$, the log-likelihoods increase and RF metrics decrease. The log-likelihood and RF metric provided by PGs, IPMCMC and PGAS with revert-merge CSMC are close.
5.7 Real data analysis

In this section, we analyze real data sets of DNA sequences. The evolutionary model we used in real analysis was the K2P model, in which the only unknown parameter is the ratio of transition to transversion, denoted by $\theta$. We make assumption that trees ($t$) are ultrametric, and consider the joint estimation of $t$ and $\theta$ in the framework of particle Gibbs sampler. The criteria we used to evaluate the tree construction quality is the log-likelihood function of majority-rule consensus tree.

5.7.1 Primate dataset

The first real data set we analyze is a set of DNA sequences for nine primates (Brown et al., 1982). In each DNA sequence, there are 888 sites. As we have investigated in Simulation Studies, with a fixed computational budget, the number of particles ($K$) is more important than the number of MCMC iterations ($R$) in improving the mixing of algorithm. We fix the number of MCMC iterations $R = 5000$, and vary $K$ to investigate the estimation by IPMCMC-RM and PGs. Table 5.2 displays the log-likelihood of consensus tree provided by IPMCMC-RM and PGs with different number of particles. We select four levels of $K$ for PGs, $K = 1000, 2000, 5000, 10000$. We set the total number of nodes for running conditional CSMC and CSMC algorithms to be twice as the number of nodes running conditional CSMC ($M = 2P$), and set $M = 4$. The computational cost of IPMCMC is a
linear function of $M$. In order to fix computational budgets for IPMCMC and PGs, we set $K$ for PGs 4 times as large as each work of IPMCMC. For each algorithm, we repeat 10 times.

Table 5.2 displays the log-likelihood for consensus tree PGs and IPMCMC-RM (mean and standard deviation), with different number of particles, each case is repeated by 10 times with different initialization of evolutionary parameters and trees. The mixing of PG chains is poor even with $K = 10000$, as multiple chains with different initializations do not converge to the same posterior distribution. The mixing of IPMCMC-RM is good when we increase the value of $K$. The log-likelihood gets higher and the standard deviation of log-likelihood decreases when we increase $K$. The IPMCMC-RM chain mixes well when $K = 10000$, as multiple chains with different initializations converge to the same posterior distribution. The mean and standard deviation for the posterior mean of $\theta$ for 10 replications provided by IPMCMC-RM are 3.68 and 0.0094, respectively. For one run of IPMCMC-RM, the posterior mean and 95% credible interval of $\theta$ are 3.69 and (3.23, 4.23) respectively. Figure 5.10 displays the traceplot of $\theta$ for one run IPMCMC-RM, the yellow dashed line is the posterior mean of $\theta$.

Table 5.2: Log-likelihood for consensus tree provided by PGs and IPMCMC-RM, with mean and standard deviation, with varying number of particles.

<table>
<thead>
<tr>
<th>$K$</th>
<th>1000</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGs</td>
<td>-8424.1 (594.5)</td>
<td>-8247.0 (723.2)</td>
<td>-8333.8 (789.5)</td>
<td>-7930.9 (549.9)</td>
</tr>
<tr>
<td>IPMCMC-RM</td>
<td>-5626.8 (1.9)</td>
<td>-5611.4 (1.7)</td>
<td>-5580.6 (1.3)</td>
<td>-5580.6 (1.3)</td>
</tr>
</tbody>
</table>

In addition, we run PGs-RM and PGAS using $K = 10000$ and $R = 5000$. The majority-rule consensus tree provided by IPMCMC-RM, PGs-RM and PGAS are the same, as displayed in Figure 5.11.
Figure 5.11: Consensus tree of the primate data set provided by PGs-RM, IPMCMC-RM and PGAS.

### 5.7.2 Cichlid Fishes dataset

The second data set we analyze is the African cichlid fish data (Kocher et al., 1995; Cheon and Liang, 2008), which consists of 12 species from two tribes (Ectodini and Lamprologini). We compare the performance of PGs, PGs-RM, IPMCMC-RM and PGAS with a fixed computational budget. The number of MCMC iterations is $R = 5000$, and the number of particles is $K = 5000$. The setup for IPMCMC-RM is the same as Primate dataset ($K = 1250$ in each worker of IPMCMC-RM). Figure 5.13 displays the majority-rule consensus trees provided by PGs and PGAS, with log-likelihood $-6560.31$ and $-4696.34$ respectively. The consensus tree provided by PGs-RM and IPMCMC-RM is consistent with PGs-RM. Figure 5.12 displays the traceplot of $\theta$ provided by one run PGAS. The posterior mean of $\theta$ is 8.47.

### 5.8 Conclusion

In this article, we propose a combinatorial sequential Monte Carlo method with a revert-merge proposal. Instead of randomly choosing a pair of trees to combine, we first use a revert step to find the parent of the current step, then in each merge step we randomly choose a pair of trees to combine. This proposal can benefit the exploration of tree posterior distribution. Our experimental results indicate that the revert-merge proposal can improve the performance of CSMC, and this improvement can be enlarged when the number of taxa increases. The framework of revert-merge CSMC is also easy to parallelize. As we have illustrated in our experiments, the computing speed gains with the
Figure 5.12: Traceplot of nucleotide substitution parameter θ provided by PGAS (Cichlid Fishes dataset).

Figure 5.13: Consensus tree of the Cichlid Fishes dataset provided by PGs (left) and PGAS (right).
increment of number of cores. This makes the proposed CSMC more scalable to large DNA data sets compared with traditional Bayesian methods, such as MCMC.

We have presented a particle Gibbs sampler, a hybrid of revert-merge CSMC and Gibbs sampler to estimate evolutionary parameters. We have demonstrated in many settings that the revert-merge CSMC can be more efficient as a high dimensional phylogenetic tree proposal in PGs. The path degeneracy issue in CSMC algorithms can cause strong autocorrelation between the evolutionary and partial states trajectory. To enhance the performance of the PGs, we also investigate interacting particle Markov chain Monte Carlo and particle Gibbs with ancestor sampling. In IPMCMC, several CSMCs and conditional CSMCs are communicated among nodes to reduce the autocorrelation and improve the mixing of MCMC chain. The performance of IPMCMC can be improved as the number of nodes increase. The parallelization among nodes can be easily achieved. We also demonstrated in our experiments that with the same computational cost, IPMCMC is preferable to PGs. In PGAS, an ancestor sampling step is implemented to reduce the correlation between two successive reference trajectories. Under the revert-merge proposal, one partial state will have more ancestors than the naive version, which makes the implementation of an ancestral sampling step possible.

There are several possible directions to refine this methodology. First, in the revert step, instead of finding the reverted state of the current state, we could randomly pick one subtree to revert. This step is more complex and more computationally expensive than the proposed revert step, but it could create more possible ancestors than the current revert-merge proposal. A second direction could be the incorporation of MCMC moves in revert-merge CSMC. Wang et al. (2019) combined MCMC and SMC in the general SMC framework. Their experimental results indicated the importance of MCMC moves in SMC. Chopin (2004) studied the asymptotic variance of SMC estimator under different resampling schemes, and displayed that advanced resampling schemes can reduce the variance of estimators. A final line of future work is to propose a computationally efficient resampling scheme, which is both unbiased and admits lower asymptotic variance for discrete tree space (Fearnhead and Clifford, 2003).

5.9 APPENDIX 1

5.9.1 Revert-merge CSMC for non-clock tree

We introduce a weight update function for the non-clock tree case. We use the same notation as in the ultrametric tree example. The proposal in the non-clock tree case is similar to the ultrametric case except for the branch length. For the proposal over branch lengths in non-clock trees, we consider two cases based on the number of trees in the forest. If there are exactly two trees in the forest before the propagation (in the last iteration of CSMC), we propose a single length ($b_1$) distributed according to an exponential distribution with rate $\lambda$ (the prior distribution). Otherwise if there are more than two trees, we propose two independent branch lengths $b_1, b_2$ according to an exponential distribution with rate $\lambda$ (the prior distribution).
The weight update function for the revert step admits the same form as the ultrametric tree case

\[ w(s, q(s)) = \frac{p(y(X_1) | \theta, t_1) p(y(X_2) | \theta, t_2)}{p(y(X_m) | \theta, t_m)}. \]

In the merge steps as mentioned in the text, there exists an overcounting issue. The backward kernel \( \nu^- \) in the naive version of CSMC for non-clock trees can be applied to the merge steps in revert-merge CSMC. For the first merge step, the probability of the backward kernel \( \nu^- \) is

\[ \nu^-_q(f(s)) = \frac{1}{\sum_{(t_i, X_i) \in f(s)} I[|X_i| > 1]} \]

where \( I[|X| > 1] \) denotes the number of non-trivial tree in forest \( t_i \). The probability of the proposal \( \nu^+ \) is

\[ \nu^+_q(f(s)) = \left( \frac{|q(s)|}{2} \right)^{-1} \lambda_q^{(|q(s)| = 2)} \cdot \exp(-\lambda_q b_1) \times \left( 1[|q(s)| = 2] + \lambda_q \exp(-\lambda_q b_2) 1[|q(s)| > 2] \right) \]

We skip the formula for the probability of backward and forward kernel of the second merge step since it is basically the same as the first one. The incremental weight function for the merge step can be written as

\[ w(q(s), f(s)) = \frac{p(y(X_1) | \theta, t_1) p(y(X_2) | \theta, t_2)}{p(y(X_m) | \theta, t_m)} \cdot \frac{1}{\sum_{(t_i, X_i) \in f(s)} I[|X_i| > 1]}. \]

\[ w(f(s), s') = \frac{p(y(X_1) | \theta, t_1^*) p(y(X_2) | \theta, t_2^*)}{p(y(X_m^*) | \theta, t_m^*)} \cdot \frac{1}{\sum_{(t_i^*, X_i^*) \in s'} I[|X_i^*| > 1]}. \]

5.10 **APPENDIX 2**

In this section, we provide algorithmic description of the conditional version of the revert-merge CSMC algorithm.

5.11 **APPENDIX 3**

5.11.1 Algorithmic description of Interacting Particle Markov chain Monte Carlo

Algorithm 20 summarizes the IPMCMC algorithm. For the sampling of \( \theta \), we use the Metropolis-Hasting algorithm described in Algorithm 15. This step is same as PGs. The updating of \( t \) involves communication between several standard and conditional revert-merge CSMC algorithms. These algorithms are referred to as ‘nodes’, and each of these nodes are assigned indices \( m \in \{1, 2, \ldots, M\} \). Among these \( M \) nodes, we run conditional revert-merge CSMC among \( P \) nodes, and the index of these conditional nodes are denoted by \( c_j, j = 1, \ldots, P \). Among the rest of the \( M - P \) nodes, we run a standard revert-merge CSMC algorithm. Superscripts are introduced to represent the index of nodes. For example, \( s_{r,k}^m \) denotes the \( k \)th particle for the partial state of rank \( r \) of the \( m \)th node. We introduce
Algorithm 19: Conditional revert-merge CSMC algorithm

1: **Inputs:** (a) Prior over partial states $p(s)$; (b) Likelihood function $p(y|s, \theta)$; (c) One pre-specified particle trajectory for partial states $s^*_{1,R}$.

2: **Outputs:** Approximation of the posterior distribution, $\sum_k W_{R,k} \delta_{s_{R,k}}(\cdot) \approx \pi(\cdot)$.

3: Initialize SMC iteration index: $r \leftarrow 1$.

4: for $r \in \{1, \ldots, R\}$ do

5: Initialize $s_{1,r} = s^*_{r}$.

6: for $k \in \{1, \ldots, K\}$ do

7: Initialize particles with the least partial state $\bot$.

8: Initialize weights to unity: $W_{1,k} \leftarrow 1/K$.

9: for $r \in \{2, \ldots, R\}$ do

10: if $\text{rESS}(W_{r-1,k}) < \epsilon$ then

11: Resample the particles with index $k = 2, 3, \ldots, K$. Note that we do not resample particles in the reference trajectory.

12: for $k \in \{1, \ldots, K\}$ do

13: Reset particle weights: $\tilde{W}_{r-1,k} = 1/K$.

14: else

15: for $k \in \{1, \ldots, K\}$ do

16: $\tilde{W}_{r-1,k} = W_{r-1,k}; \tilde{s}_{r-1,k} = s_{r-1,k}$.

17: for $k \in \{1, \ldots, K\}$ do

18: if $k > 1$ then

19: Sample particles $s_{r,k} \sim \nu^+_{\tilde{s}_{r-1,k}}(\cdot)$, using one revert move to find $\omega(\tilde{s}_{r-1,k})$, and two merge moves to propose $f(\tilde{s}_{r-1,k})$ and $s_{r,k}$.

20: Compute pre-resampling normalized weights:

$$W_{r,k} \propto \tilde{W}_{r-1,k} \cdot \frac{\pi(s_{r,k}) \cdot \nu^+_{\tilde{s}_{r-1,k}}(\tilde{s}_{r-1,k})}{\pi(\tilde{s}_{r-1,k}) \cdot \nu^+_{\tilde{s}_{r-1,k}}(s_{r,k})}.$$ 

21: Construct

$$\pi_{r,K}(s) = \frac{1}{K} \sum_{k=1}^K W_{r,k} \delta_{s_{r,k}}(s).$$
the notation $\hat{Z}$ to denote the marginalized likelihood estimates. The marginalized likelihood for node $m$ can be written as

$$\hat{Z}^m = \prod_{r=2}^{R} \frac{1}{K} \sum_{k=1}^{K} w_{r,k}^m.$$ 

After computing the marginalized likelihood for each node, we sample new index for each node $c_j$ according to

$$P(c_j = m| c_{1:P(j)}) = \hat{\zeta}_j^m,$$

$$\hat{\zeta}_j^m = \frac{\hat{Z}^m 1_{m \in c_{1:P(j)}}}{\sum_{m=1}^{M} \hat{Z}^m 1_{m \not\in c_{1:P(j)}}}.$$ 

Then we sample the new reference trajectory for node $c_j$ with

$$P(b_j = k|c_j) = w_{k,R}^{c_j}.$$ 

The updated reference trajectory for node $c_j$ is $s_{1:R}^{c_j,*} = s_{b_j,1:R}^{c_j}$. The switch of nodes is the essential part of IPMCMC.

5.12 APPENDIX 4

List of Symbols
Algorithm 20: Interacting particle Markov chain Monte Carlo

1: **Inputs:** (a) Prior over partial states \( p(s) \) and evolutionary parameter \( p(\theta) \); (b) Likelihood function \( p(y|s, \theta) \); (c) Pre-specified particle trajectories for nodes; (d) MH proposal for \( \theta \), \( q(\cdot|\theta) \); (e) number of nodes \( M \) and number of conditional nodes \( P \).
2: **Outputs:** Approximated posterior distribution \( \hat{p}(\theta, t|y) \).
3: Initialize MCMC iteration index \( i \leftarrow 0 \).
4: Initialize \( \theta(0) \) to an arbitrary value.
5: Initialize an index \( m = \{1, 2, \ldots, M\} \) to each node.
6: for \( m \in \{c_1, c_2, \ldots, c_P\} \) do
7: Run the revert-merge CSMC algorithm targeting \( p(t|y, \theta(i - 1)) \).
8: for \( m = 1, 2, \ldots, M \) do
9: Run the conditional revert-merge CSMC algorithm targeting \( p(t|y, \theta(i - 1)) \), conditional on the reference trajectory \( s_{c_j}^{\ast, \ast} \).
10: Update the marginalized likelihood \( \hat{Z}^m = \prod_{r=2}^{R} \frac{1}{K} \sum_{k=1}^{K} w_{r,k}^m \).
11: Update index \( c_j \) by sampling with probability
\[
P(c_j = m | c_{1:P}, j) = \hat{\zeta}_j^m, \]
\[
\hat{\zeta}_j^m = \frac{\hat{Z}^m 1_{m \in c_{1:P}, j}}{\sum_{n=1}^{M} \hat{Z}^n 1_{n \notin c_{1:P}, j}}.
\]
12: Update index \( b_j \) with probability
\[
P(b_j = k | c_j) = w_{k,R}^{c_j}.
\]
13: Update the reference trajectory \( s_{1,R}^{c_j} = s_{b_j,1,R}^{c_j} \).
14: Randomly sample a reference trajectory \( s_{1,R}^{\ast} \) from \( \{s_{1,R}^{c_j}\}_{j=1}^{P} \).
15: Sample \( \theta(i) \sim p(\theta|y, t = s_{1,R}^{\ast}) \) using the accept-reject Metropolis Hastings algorithm.
16: Burn in a proportion of the IPMCMC chain. \( \triangleright \) For example, we burn in 50% of the chain in our experiment.

97
\( y \) observed biological sequence

\( X \) a set of observed taxa

\( t \) a phylogenetic \( X \)-tree

\( \theta \) parameter in the nucleotide substitution model

\( p(\theta) \) the prior distribution of \( \theta \)

\( p(t) \) the prior distribution of \( t \)

\( p(y|\theta) \) the likelihood of data

\( \pi(\cdot) \) normalized posterior distribution of \( \cdot \)

\( \gamma(\cdot) \) unnormalized posterior distribution of \( \cdot \)

\( q(\cdot) \) proposal distribution for \( \cdot \)

\( \alpha \) MH acceptance ratio

\( \kappa \) ratio of transition to transversion in K2P model

\( k \) index of particle

\( K \) total number of particles

\( r \) index of rank

\( R \) largest rank value

\( i \) index of MCMC iteration

\( N \) total number of MCMC iterations

\( s_r \) partial state with rank \( r \)

\( s_{r,k} \) \( k \)-th particle for partial state with rank \( r \)

\( \tilde{s}_{r,k} \) \( k \)-th particle for partial state of rank \( r \) after resampling

\( |s| \) number of trees in forest

\( g(\cdot) \) the reverted state of \( \cdot \)

\( f(\cdot) \) the merge state of \( \cdot \)

\( w_{r,k} \) unnormalized weights for \( k \)-th particle of partial state rank \( r \)

\( W_{r,k} \) normalized version of \( w_{r,k} \)

\( \nu^- \) a backward kernel

\( \nu^+ \) a forward kernel

\( \text{rESS} \) relative effective sample size

\( a_r \) index of ancestor sampling trajectory for partial state of rank \( r \)

\( m \) index of node in IPMCMC

\( M \) total number of nodes in IPMCMC
Chapter 6

A Bayesian nonparametric approach to genome-wide association studies

6.1 Introduction

Multidrug-resistant tuberculosis (MDR-TB) is a major concern for disease control (Grandjean et al., 2017). Developing new therapies for tuberculosis (TB) patients is challenging and expensive. Multidrug-resistance in TB is caused by genetic variations in genes that encode drug targets and drug-converting enzymes (Coll et al., 2014). Understanding these effects is critical for improving treatment for MDR-TB patients, but population stratification (in which genetic variates correlate with structure in geographical or socioeconomic indicators), and noisy genotyping, confounds genetic studies (Price et al., 2006; Zhang et al., 2010). In this work, we improve the control provided by linear mixed models, and report applications on TB data and simulated data.

Genome-wide association studies (GWAS) are designed to identify the genetic variants affecting phenotypes of interest such as multidrug-resistance in tuberculosis (Price et al., 2006; Zhang et al., 2010). Classic approaches to GWAS rely on linear association tests to quantify the relationship between phenotypes and genotypes. Linear association tests can be confounded by population structure, leading to false positives, spurious associations, and inflated $p$-values (Novembre et al., 2008). The genealogy of tuberculosis populations typically exhibit strong clade structure (Earle et al., 2016; Cordero and Polz, 2014), with geographically widespread lineages, and so GWAS are vulnerable to population stratification.

Linear mixed models (LMMs) use the genetic similarity among the samples as a fixed effect. This controls for confounding from population structure, leading to improved false discovery rates. In Kang et al. (2010), the model EMMAX was proposed, which computes the variance component in linear mixed models in an efficient way. Recently, factored spectrally transformed linear mixed models (FaST-LMM) were introduced Lippert et al. (2011); Listgarten et al. (2013), with running time and memory costs that scale linearly in cohort size. This allows LMMs to be scaled to large cohorts Consortium et al. (2007). In Zhou and Stephens (2014); Yang et al. (2011); Dahl et al. (2016), models were developed for computationally efficient linear mixed effects model with multivariate
phenotypes. The efficiency of FaST-LMM methods were improved by subsetting the genetic variants examined, so that a set of maximally independent genetic variants are considered Listgarten et al. (2012).

All existing LMM methods assume that the matrix specifying the genetic similarity among the samples is known (i.e., through an empirical genetic similarity matrix, or a kinship matrix derived from a pedigree). For large cohorts of human genotypes, there is often low uncertainty about estimated genetic similarity matrices. However, for some studies, such as bacterial studies, in which small numbers of samples are present, or for which genotyping is sparse and noisy, uncertainty about the genetic similarity matrix may degrade the quality of LMM results.

In this work, we propose a full Bayesian nonparametric treatment of the LMM for genome-wide association studies, using phylogenetic trees to control for population structure. Bayesian nonparametric methods are used in machine learning and include posterior inference for models in which the parameter space is organised as a combinatorial structure, such as a tree (Görür and Teh, 2009). Classical methods for GWAS with phylogenetic trees (such as BEAST (Bouckaert et al., 2014) and SEER (Puranen et al., 2018)) control population structure either by using a single consensus tree found by averaging sampled phylogenies, or by performing one controlled GWAS for each sampled phylogeny and averaging the results.

A Bayesian view of the LMM implies a conditional dependence between the phenotype and the LMM parameters (demonstrated in Figure 6.1). The dependence arises from a non-multiplicative link between the LMM parameters and the genetic similarity matrix. In our proposed tree-based linear mixed model (which we refer to as the TreeLMM), we derive sampling updates for the phylogenies and also the LMM parameters conditioned on the phenotype. We combine these updates to create a chain that respects this dependence and targets the TreeLMM posterior.

We develop a sequential Monte Carlo (SMC) framework (Doucet et al., 2001; Wang et al., 2019) for TreeLMM inference. We also propose an algorithm to compute the expected value of the genetic similarity matrix, given an estimated phylogeny. Simulations show that the true positive rates and false discovery rates of TreeLMM outperform both existing linear regression methods and LMM methods in which the genetic similarity matrix is estimated empirically. We apply this method to MDR-TB in a GWAS of 469 TB subjects in a population from Lima, Peru (Grandjean et al., 2017).

### 6.2 Methods

#### 6.2.1 Linear mixed effects models for genome-wide association studies

Genetic sequences are often described by a series of genome locations at which a mutation is observed in the ancestry of the subjects. We consider single nucleotide polymorphisms (SNPs), locations at which a single DNA basepairs can appear with more than one form (or, allele). This sequence is known as the genotype. The most common form is known as the major allele, and the less common forms are the minor alleles. We begin this section with an exposition of the linear mixed model (Kang et al., 2010; Lippert et al., 2011). Let the study subject indices be $i = 1, 2, \ldots, N$, and let
Figure 6.1: Graphical structure of TreeLMM. The phenotype $y$ is a collider (Pearl, 2000). Conditioning on $y$ and genotype $G$ induces dependence among the effect size $\beta$, the phylogeny $t$, and the environmental and genetic variance ($\sigma^2_g$ and $\sigma^2_e$). Plate diagram for a) TreeLMM, and b) TreeLMM conditioned on genotype and phenotype.

The SNP indices be $m = 1, 2, \ldots, M$. Let $y = (y_1, y_2, \ldots, y_N)'$ denote a column vector of phenotypes ($y_i \in \mathbb{R}$), and let $G = [G_1, G_2, \ldots, G_M]$ denote genotype data observed at the $M$ SNPs, with $G_m$ denoting a column vector of alleles for the $m$-th SNP for all $N$ subjects. For details on bacterial genetics we refer the reader to Earle et al. (2016) and Coll et al. (2014). Let $G_{im} = 0$ and $G_{im} = 1$ encode the events that subject $i$ has the major allele or the minor allele at SNP $m$ respectively.

The LMM is a mixed effects model for association between SNP $G_m$ and the phenotype. Independent LMMs may be applied at each SNP as follows:

$$y = G_m \beta_m + b_m + \varepsilon_m. \quad (1)$$

Here $\beta_m$ is the effect size of the fixed effect of the $m$-th SNP, $\varepsilon_m$ is the random error vector, with $\varepsilon_m \sim \text{MVN}(0, \sigma^2_e I)$, and $b_m$ is the random effect of the $m$-th SNP, with $b_m \sim \text{MVN}(0, \sigma^2_e \psi)$, and MVN(0, $\Sigma$) is the multivariate normal distribution with mean 0 and covariance $\Sigma$. The genetic similarity matrix $\psi$ measures the genetic relatedness among different subjects. This is a $N \times N$ positive semi-definite matrix, and an empirical estimate of $\psi$ is given by (Kang et al., 2010):

$$\psi_{ij} = \frac{1}{M} \sum_{m=1}^{M} \frac{(G_{im} - \mu_m)(G_{jm} - \mu_m)}{\sigma^2_m}. \quad (2)$$

Here $\mu_m = \frac{1}{N} \sum_{i=1}^{N} G_{im}$, $\sigma^2_m = \mu_m (1 - \mu_m)$ are the empirical mean and variance (respectively) of the genotypes of the $N$ subjects at the $m$-th SNP.

While previous LMM work approximates $\psi$ by (2), the true value of $\psi$ is unknown. A realisation of $\psi$ is implied deterministically by a phylogenetic tree for the $N$ subjects. We denote this tree by $t$.

In the next subsection we introduce TreeLMM, treating the tree $t$ as a latent and unobserved variable.

6.2.2 The tree-based linear mixed model

The covariance matrix of the random effects for the $m$-th SNP is the positive-definite matrix $\psi(t)$, which depends on a phylogenetic tree $t$ encoding the ancestry of the sample. Conditioning on $y$ and $G$ induces dependence between $\beta$ and $t$. Hence joint estimation of LMM parameters and the
phylogeny \( t \) is different from a procedure in which first \( t \) is estimated, and then LMM parameters are inferred, conditioned on \( t \). We consider joint inference on the phylogeny \( t \) and the LMM parameters. Following the derivation of the LMM in Section 2.1, the likelihood for the parameters in our model is as follows:

\[
\mathbb{P}(y, G|t, \beta, \sigma^2_e, \sigma^2_g) = \mathbb{P}(G|t) \prod_{m=1}^{M} \mathbb{P}(y_m|\beta_m, \sigma^2_e, \sigma^2_g, \psi(t), G_m).
\]  

(3)

Here \( y_1 = y_2 = \ldots = y_m = y \). We add subscripts \( m \) to the phenotype in each linear mixed model to emphasize that we test each SNP separately (Zhou and Stephens, 2012). Equation 3 is a product of \( M \) linear mixed models linked through a common phylogeny. In Figure 6.1 we provide a plate diagram describing the graphical structure of TreeLMM (hyperparameters are not shown, and observed variables are indicated by grey circles).

To construct the hierarchical Bayesian framework, we place prior distributions over the parameters. We set the prior on the fixed effects \( \beta = \{\beta_m\}_{m=1,...,M} \) to be a normal distribution with hyper-parameters \( \mu_m, \sigma^2_m \). We set the prior for the variance parameters \( \sigma^2_e \) and \( \sigma^2_g \) to be inverse-gamma distributions with hyper-parameters \( a_0, b_0 \) and \( g_0, h_0 \), respectively. We place a uniform prior \( |\tau|^{-1} \) on the phylogenetic tree topology \( \tau \) (here \( |\tau| \) denotes the number of ranked tree topologies), and a product of independent exponential distributions with rate \( \lambda_0 \) over the lengths of the branches \( |e_d| \) of the phylogeny (i.e., \( t \) is a non-clocked, unrooted binary tree). The induced hierarchical model is described by the following scheme:

\[
y|\beta_m, b_m, \sigma^2_g \sim \text{MVN}(G_m \beta_m + b_m, \sigma^2_g I), \quad \beta_m \sim \text{N}(\mu_m, \sigma^2_m), \quad m = 1, \ldots, M, \tag{4}
\]

\[
b_m|\sigma^2_e, \psi(t) \sim \text{MVN}(0, \sigma^2_e \psi(t)), m = 1, \ldots, M, \quad \sigma^2_e \sim \text{IG}(a_0, b_0), \tag{5}
\]

\[
\sigma^2_g \sim \text{IG}(g_0, h_0), \quad |\tau| \sim |\tau|^{-1}, \quad |e_d| \sim \exp(\lambda_0), \quad d = 1, \ldots, 2N - 3. \tag{6}
\]

We are interested in posterior inference over \( \Omega = (\beta, b, \sigma^2_e, \sigma^2_g, t) \). We apply (4,5,6) separately to each of the \( M \) SNPs, and link the variance parameters, yielding the following joint posterior distribution:

\[
\pi(\beta, b, \sigma^2_e, \sigma^2_g, t) \propto \pi_0(\sigma^2_e, \sigma^2_g, \beta, t) \mathbb{P}(G|t) \prod_{m=1}^{M} \mathbb{P}(b_m|\sigma^2_e, \psi(t)) \mathbb{P}(y_m|\beta_m, b_m, \sigma^2_g, G_m). \tag{7}
\]

6.2.3 Phylogenetic tree reconstruction

Trees can be sampled conditioned on genotype data (Wang et al., 2019). The leaves of a sampled tree \( t \) are the subjects in the study. Each interior node of \( t \) represents the latest common ancestor of the two children of that node, and the branch lengths are the evolutionary distance between nodes.

A phylogeny \( t \) represents the relationships among \( N \) taxa through the tree topology \( \tau \) and the set of branch lengths \( e = (e_1, e_2, \ldots, e_{2N-3}) \). We assume that the phylogeny \( t \) is an unobserved latent variable. The tree \( t \) can be estimated by conditioning on \( G \), the genotype, and through assumptions
about the mutation model which are used during the calculation of the likelihood of the genotype $G$ given a phylogeny $t$.

We assume that the observed SNPs are independent (as is commonly assumed for TB sequences (Coll et al., 2014)), and so the likelihood of $G$ given $t$ can be evaluated by using Felsenstein pruning (Felsenstein, 1973, 1981): the branch lengths denote the evolutionary time between two nodes, and a continuous-time Markov chain (Hajiaghayi et al., 2014; Zhao et al., 2016) models the evolution of each site along each branch of the phylogeny $t$. We assume the Jukes-Cantor mutation model (Jukes et al., 1969), with base frequencies and mutation rates all constant.

### 6.2.4 Expected genetic similarity matrices from phylogenies

![Figure 6.2: Segregation by an unobserved SNP on edge $e_d$. Computation of the expected mean $\mu_{e_d}$, and variance $\sigma^2_{e_d}$, for an unobserved allele involves integration over edges $e_d$, and examination of the connected components of the tree formed by removing edge $e_d$.](image)

Here $\mu$, and $\sigma^2$ are the mean and variance of the alleles. To compute (8) we integrate the location of an unobserved SNP over $t$. The weights of this integral over each branch are given by $|e_d|/|t|$ and values of $G_i$, $\mu$ and $\sigma^2$ are determined by the location of the SNP. Here $|e_d|$ is the branch length of $e_d$, and $|t|$ is the sum of the branch lengths of all edges in the tree.

The values $G_{ie_d}$, $\mu_{e_d}$, $\sigma^2_{e_d}$ are found by considering a mutation on the edge $e_d$, as follows: 1) Let $c_0$ and $c_1$ be the two connected components in the tree formed by removing $e_d$ from $t$. Assume the number of leaves in $c_0$ is greater than or equal to that of $c_1$. Each leaf in $c_0$ will have the major allele, and each leaf in $c_1$ will have the minor allele. Figure 6.2 shows an example of this operation. 2) $G_{ie_d}$ is 1 if leaf $i$ is in $c_1$ and $G_{ie_d}$ is 0 if leaf $i$ is in $c_0$. 3) $\mu_{e_d}$ is the number of leaves in $c_1$ divided by $N$, and $\sigma^2_{e_d}$ is the variance $\mu_{e_d}(1 - \mu_{e_d})$. 

$$
E[\psi_{ij}] = E\left[\frac{(G_i - \mu)(G_j - \mu)}{\sigma^2}\right] = \sum_{d=1}^{2N-3} \frac{|e_d|}{|t|} \cdot \frac{(G_{ie_d} - \mu_{e_d})(G_{je_d} - \mu_{e_d})}{\sigma^2_{e_d}}.
$$

(8)
6.2.5 TreeLMM inference with sequential Monte Carlo

We are interested in posterior inference of the LMM parameters and the phylogeny, $\Omega = (\beta, b, \sigma_g^2, \sigma_e^2, t)$. The posterior $\pi(\Omega)$ of $\Omega$ is:

$$\pi(\Omega) = \frac{\gamma(\Omega)}{Z} = \frac{\pi_0(\Omega)P(G, y|\Omega)}{Z}. \quad (9)$$

Here $\gamma(\Omega)$ is the unnormalized posterior and $Z = \int \pi_0(\Omega)P(G, y|\Omega)d\Omega$ is the normalizing constant. We consider an annealed (Neal, 2001; Del Moral et al., 2006) SMC algorithm for inferring the posterior. We define intermediate target SMC distributions $\gamma_r(\Omega)$ as follows:

$$\gamma_r(\Omega) = \pi_0(\sigma_g^2, \sigma_e^2, \beta, t) \left[ P(G|t) \prod_{m=1}^{M} P(b_m|\sigma_e^2, K(t)) \right]^{\phi_r} \prod_{m=1}^{M} P(y_m|\beta_m, b_m, \sigma_g^2, G_m). \quad (10)$$

Here $\phi_r (r = 1, \ldots, R)$ is a strictly increasing exponent with $\phi_1 = 0$ and $\phi_R = 1$. Small values of $\phi_r$ flatten the tree posterior and allows the sample to move easily among local modes. The samples gradually approach the invariant distribution $\pi(\Omega)$ as $\phi_r$ increases.

There are several advantages of our proposed SMC over MCMC methods with regards to local modes, mixing, and parallelization (Chopin, 2004; Del Moral et al., 2006; Wang et al., 2019).

6.2.6 Details of the SMC algorithm for TreeLMM inference

At each SMC iteration $r$, we keep a collection of $K$ SMC particles $\Omega_{r,k}$, associated with the weight $W_{r,k}$. We initialize the particles from the prior $\pi_0$. Our SMC algorithm iterates between $R$ propagation, weighting and resampling steps, approximating the target posterior $\pi(\Omega) \approx \sum_{k=1}^{K} \delta_{\Omega_{r,k}}(\Omega)$. The steps are as follows:

**Propagation step:** We propagate new particles $\Omega_{r,k}$ ($k = 1, \ldots, K$) through the $\pi_r$-invariant MCMC kernel $K_r(\Omega_{r-1,k}, \Omega_{r,k})$, with $\pi_r(\Omega) \propto \gamma_r(\Omega)$. The closed form full conditional distributions for $\beta, b, \sigma_e^2$ and $\sigma_g^2$ admitting $\pi_r(\Omega)$ as an invariant are listed in (1-4) in Appendix 2. We provide efficient sampling methods for updating $b_m$ and $\sigma_e^2$ using the Woodbury matrix identity and the matrix trace in Appendix 3.

The genealogy of TB populations typically exhibit strong clades structure (Earle et al., 2016; Cordero and Polz, 2014), and so for computational reasons we assume independent between clades. The clades are determined in a preprocessing step conducted on the genotypes. We compute $\psi_{ij}(t)$ for each clade according to Section 6.2.4, and assume $\psi_{ij}(t) = 0$ if $i$ and $j$ are in different clades.

We use a Metropolis-Hastings algorithm based on (Wang et al., 2019) to sample trees $t_r$. Details are described in Appendix 2.

**Weight update step:** We update the weighting function according to:

$$W_{r,k} \propto W_{r-1,k}w(\Omega_{r-1,k}, \Omega_{r,k}) = W_{r-1,k} \frac{\gamma_r(\Omega_{r,k})}{\gamma_{r-1}(\Omega_{r-1,k})} \frac{L_{r-1}(\Omega_{r,k}, \Omega_{r-1,k})}{K_r(\Omega_{r-1,k}, \Omega_{r,k})}. \quad (11)$$
Here $L_{r-1}(\Omega_r, \Omega_{r-1}, k)$ is a backward kernel (Del Moral et al., 2006) that allows the importance weights to be easily evaluated:

$$L_{r-1}(\Omega_r, \Omega_{r-1}, k) = \frac{\pi_r(\Omega_{r-1}, k) K_r(\Omega_{r-1}, \Omega_r, k)}{\pi_r(\Omega_r, k)}.$$  \hspace{1cm} (12)

With this backward kernel, the incremental importance weights are:

$$w(\Omega_{r-1}, k, \Omega_r, k) = \frac{\gamma_r(\Omega_{r-1}, k)}{\gamma_{r-1}(\Omega_{r-1}, k)} \left[ \prod_{m=1}^{M} \mathbb{P}(b_m(\tau_{r-1}, k) | \sigma^2_{e(\tau_{r-1}, k)}, \psi(\tau_{r-1}, k)) \right]^{\phi_r - \phi_{r-1}}.$$

**Resampling step:** When the relative effective sample size ($rESS$) (Wang et al., 2019) of the particles falls below a threshold $\epsilon$, we resample the particles according to their weights ($rESS$ is defined in Appendix 4).

### 6.3 Experiments

In our experiments, we use the following settings of the hyper-parameters and priors: $\mu_m = 0.0$, $\sigma^2_m = 100.0$, $a_0 = 1.0$, $b_0 = 1.0$, $g_0 = 1.0$, $h_0 = 1.0$. We assume a uniform prior for the tree topology, and we assume that the branch lengths follow exponential distributions with rates $\lambda_0 = 10.0$. In all conditions, we use a cubic annealing parameter scheme $\phi_r = (r/R)^3$, and we set the threshold for $rESS$ resampling to $\epsilon = 0.5$.

#### 6.3.1 Association study of MDR-TB in Lima, Peru

We carry out a genome-wide association study using the TreeLMM method to control for population structure for 469 tuberculosis subjects (of which 61 have multidrug-resistant strains) collected in Lima, Peru. These data were studied in (Grandjean et al., 2017), and in that work many homoplastic variant sites were identified to be significantly correlated, indicating *epistasis*. Our analysis further refines these results with a TreeLMM control for population structure. We removed genotypes with minor allele frequency below 0.02, yielding 5,560 SNPs. We compare LM, LMM and the TreeLMM. For the LMM, we use the empirical genetic similarity matrix, and run MCMC inference on the LMM parameters with 20,000 iterations. For TreeLMM, we run SMC algorithm with 100 particles and 6,000 intermediate target distributions. Our data is provided with class labels INH MONO, MDR, NO HAY, RESISTENTE and RIF MONO. We define our MDR phenotype $y$ by combining classes 1 and 2 for MDR, and the remaining classes for no-MDR.

We compare our methods to a classical linear regression GWAS with $t$-tests. This linear analysis identifies 25 genetic variants that are significantly associated with multidrug-resistance after Bonferroni (BF) correction with $p$-value $< 0.05/5,560$. For these 25 associated SNPs, we perform GWAS using the LMM and the TreeLMM. Figure 6.3 displays the Manhattan plot for these GWAS. The TreeLMM identifies 2 genetic variants are significantly associated (red squares) after BF correction.
Figure 6.3: A Manhattan plot of SNP association with MDR-TB phenotype: $p$-values are shown for linear association tests (grey circles), the linear mixed model with empirical genetic similarity matrix (blue triangles), and tree-based linear mixed model (red squares).

$(p$-values $< 0.05/5, 560).$ Both LMM and TreeLMM significantly correct the 25 hits found through linear regression, suggesting that many of these hits are due to population structure.

### 6.3.2 Simulation studies

We simulate datasets for three Scenarios: $A, B, C.$ For each Scenario, we simulate 50 replicate datasets, all with the same conditions. In Scenario $A$, we assume clade structure, for which each clade has $N = 30$ subjects, and the number of SNPs is $M = 2000$ (there are 5 clades in total); In Scenario $B, N = 80, M = 2000$; In Scenario $C, N = 15, M = 100.$ In Scenarios $B$ and $C$, we assume no clade structure.

For these simulated datasets, we generate sets of random unrooted trees, including the topology and branch lengths, as reference trees. The tree topologies are simulated from a uniform distribution, and branch lengths are drawn from an exponential distribution with rate 10.0. For Scenario $A$, these trees are simulated independently for each clade, whereas for Scenarios $B$ and $C$, a single tree is simulated for all subjects.

We simulate the genotypes $G,$ and add noise to $G,$ representing genotyping error, by flipping 2% of the sites for each subject, randomly. We standardize the genotypes, and uniformly choose one SNP to be significant. We compute a ground-truth genetic similarity matrix given the reference trees, according to Section 6.2.4. We simulate the phenotype though the LMM described in Section 6.2.1, with $\sigma^2 = 0.36, \sigma^2_g = 0.25,$ and effect size $\beta = 0.20.$ We assume no correlation among subjects in different clades.

We compared the true positive rate (TPR) and false discovery rate (FDR) of TreeLMM, LMM and LM in a task in which associated genetic variants are recovered. We examine the receiver operating characteristic (ROC) curves induced by the $p$-values for these models for simulated data in which
Table 6.1: AUC and TPR at FDR 0.05 for simulation Scenarios A and B. TreeLMM shows improved AUC and TPR in both Scenarios.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>AUC</th>
<th>LMM</th>
<th>TreeLMM</th>
<th>TPR (FDR = 0.05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.875</td>
<td>0.907</td>
<td>0.936</td>
<td>0.596 0.638 0.745</td>
</tr>
<tr>
<td>B</td>
<td>0.739</td>
<td>0.740</td>
<td>0.873</td>
<td>0.323 0.323 0.452</td>
</tr>
</tbody>
</table>

the ground truth is known. We also conduct simulation study to demonstrate the parallelization and consistency of the SMC algorithm.

**Effectiveness of TreeLMM**

We compare the performance of LM, LMM and our TreeLMM model using Scenarios described in Section 6.3.2. For TreeLMM, annealed SMC with a cubic annealing parameter scheme is used to estimate the model, with $K = 100$ and $R = 8,000$. We compute $p$-values for each SNP involved in the analysis (i.e. top 100 associated SNPs identified by LM in each replicate) for all three methods to plot the ROC curve. Figure 6.4 displays the ROC provided by these methods for datasets A (left panel) and B (right panel). The ROC curve of TreeLMM dominates those of LMM and LM at all FDR level in both Scenarios, and the area under the ROC curve (AUC) and the improvement at FDR=0.05 are listed in Table 6.1.

In Appendix 5, we describe speed improvements on the simulation Scenario C, gained through parallelization of our SMC methods for TreeLMM.

**Consistency of SMC**

We conduct experiments to compare the performance of SMC with variation of $K$ (panels 1, 2 of Figure 6.5, with $R = 200$) and with variation of $R$ (panels 3, 4 of Figure 6.5, with $K = 100$) using simulation Scenario A. Besides comparing ROC curve (i.e. based on top 100 associated
SNPs identified by LM in each replicate), we also performed some benchmarking on the quality of the inferred trees. To do so, we computed the majority-rule consensus tree (Felsenstein, 1981) to summarize the weighted phylogenetic tree samples obtained from SMC, and compared the Robinson-Foulds metric (Robinson and Foulds, 1981) between this tree and the ground truth. Figure 6.5 indicates a larger number of particles and intermediate target distributions improve the performance of TreeLMM both in terms of ROC and RF metric. This demonstrates the consistency of our SMC algorithm.

### 6.4 Results

In our genome-wide association study on multidrug-resistant TB, TreeLMM identifies reference 2211835 in gene mpt63, and reference 3448791 in gene accA3 as hits (Grandjean et al., 2017). Our simulation results (in Section 6.3.2) suggest that the TreeLMM has more power and lower FDR than TreeLMM, providing evidence that the two hits recovered by TreeLMM are more likely to be true positives and that the 23 hits found by linear association (and not TreeLMM) are likely to be spurious.

### 6.5 Discussion

We have developed a tree-based linear mixed effects model for genome-wide association studies, in which the genetic similarity matrix is induced by a phylogeny in a hierarchical nonparametric Bayesian framework. Our proposed method, the TreeLMM, allows us to integrate over the uncertainty in the LMM parameters and the phylogeny. These parameters are conditionally dependent, given the phenotype, and we propose an efficient SMC method to infer the posterior of the TreeLMM parameters.

Our simulations demonstrate the consistency of our methods, and improved false positive rates over the LMM. In our genome-wide association study on multidrug-resistant TB, the TreeLMM significantly corrects 25 associations found through linear regression.
6.6 APPENDIX 1: Computation of expected genetic similarity matrices

Algorithm 21 displays the pseudo-code for the computation of expected genetic similarity matrices.

**Algorithm 21 From a phylogeny to an expected genetic similarity matrix**

1: **Inputs:** An estimated phylogenetic tree \( t \) with a tree topology \( \tau \) and a set of branch length \( e = (e_1, e_2, \ldots, e_{2N-3}) \).
2: **Output:** An \( N \times N \) expected genetic similarity matrix \( \psi(t) \).
3: Initialize \( \psi_{ij}(t) \leftarrow 0 \).
4: Compute the total tree distance \( |t| \) by adding up branch length \( e \).
5: for \( e \in \{e_1, e_2, \ldots, e_{2N-3}\} \) do
6: Compute \( \mu_e \) and \( \sigma^2_e \)
7: for \( i \in \{1, 2, \ldots, N\} \) do
8: for \( j \in \{i, \ldots, N\} \) do
9: Set \( \psi_{ij}(t) \leftarrow \psi_{ij}(t) + \frac{|e|^{(G_{i,j} - \mu_e)(G_{j,i} - \mu_e)}}{\sigma^2_e} \). Set \( \psi_{ji}(t) \leftarrow \psi_{ij}(t) \).

6.7 APPENDIX 2: Updates for SMC algorithm

We propagate new samples \( \Omega_{r,k} \) \( (k = 1, \ldots, K) \) through \( \pi_r \)-invariant MCMC kernels \( K_r(\Omega_{r-1,k}, \Omega_{r,k}) \), with \( \pi_r(\Omega) \propto \gamma_r(\Omega) \).

6.7.1 Conditional distributions for LMM parameters

There exists closed form full conditional distributions for \( \beta, b, \sigma^2_g, \sigma^2_e \) that admit \( \pi_r(\Omega) \) as an invariant. These closed forms are as follows.

\[
\beta_m(r) | \beta_m(r), \sigma^2_g(r) \sim N(\mu_{\beta_m(r)}, \sigma^2_{\beta_m(r)}).
\] (13)
Here,

\[
\sigma^2_{\beta_{m(r)}} = \left( \frac{G'_m G_m}{\sigma^2_{g(r)}} + \frac{1}{\sigma^2_{m}} \right)^{-1}, \quad \mu_{\beta_{m(r)}} = \sigma^2_{\beta_{m(r)}} \left( \frac{G'_m (y_m - b_{m(r)})}{\sigma^2_{g(r)}} + \mu_m \frac{1}{\sigma^2_{m}} \right)
\]

\[
b_{m(r)}|\beta_{m(r)}, \sigma^2_{g(r)} \sim \text{MVN}(\mu_{b_{m(r)}}, \Sigma_{b_{m(r)}}).
\]

\[
\Sigma_{b_{m(r)}} = \left( \sigma^{-2}_{g(r)} I + \phi_r \psi(t_r)^{-1} \sigma^{-2}_{e(r)} \right)^{-1}, \quad \mu_{b_{m(r)}} = \Sigma_{b_{m(r)}} (y_m - G_m \beta_{m(r)})' \frac{1}{\sigma^2_{g(r)}}.
\]

\[
\sigma^2_{e(r)}|b_{m(r)} \sim \text{IG}(a^*_0, b^*_0).
\]

\[
a^*_0 = a_0 + \frac{NM}{2} + 1, \quad b^*_0 = b_0 + \phi_r \sum_{m=1}^{M} b'_{m(r)} \psi(t_r)^{-1} b_{m(r)} \frac{1}{2}.
\]

\[
\sigma^2_{g(r)}|b_{m(r)} \sim \text{IG}(g^*_0, h^*_0).
\]

\[
g^*_0 = g_0 + \frac{NM}{2} + 1, \quad h^*_0 = h_0 + \frac{1}{2} \sum_{m=1}^{M} \frac{(y_m - G_m \beta_{m(r)})'(y_m - G_m \beta_{m(r)} - b_{m(r)})}{2}.
\]

### 6.7.2 Metropolis algorithm for phylogenetic tree inference

There is no closed form posterior for the phylogeny \( t, \) and so we use a Metropolis-Hastings algorithm with a mixture kernel to propagate new trees \( t_r. \) The proposals \( q^j \) are defined as follow (Wang et al., 2019):

1. \( q^1: \text{multiplicative branch proposal}. \) This update picks one edge at random and multiplies the current value of the branch length by a random number distributed uniformly in \([1/a, a]\) for some fixed parameter \( a > 1. \) The fixed parameter \( a \) controls the boldness of the proposal (Lakner et al., 2008).

2. \( q^2: \text{global multiplicative branch proposal}. \) This update modifies all the branch lengths by applying the above multiplicative branch proposal to each branch.

3. \( q^3: \text{stochastic NNI proposal}. \) This update uses the nearest neighbor interchange (NNI) (Jow et al., 2002) to propose a new tree topology.

4. \( q^4: \text{stochastic NNI proposal with resampling the edge}. \) This update uses the above NNI proposal in 3 and the multiplicative branch proposal in 1 for the edge under consideration.

5. \( q^5: \text{Subtree Prune and Regraft (SPR) move}. \) We first pick an internal node uniformly at random, detach the subtree associated with the selected node. Then we uniformly pick an internal branch uniformly at random, re-attach the detached subtree to the pick branch (Bordewich and Semple, 2005).
We first randomly pick a proposal $q^*$, propose a new tree $t_r$ conditional on the current tree $t_{r-1}$, then we accept the proposed $t_r$ with the following acceptance probability

$$\alpha_r(t_{r-1}, t_r) = \min\left\{1, \frac{\pi_0(t_r)P(G|t_r)\prod_{m=1}^{M}\mathbb{P}(b_m(t_{r-1})|\sigma_{e(r-1)}, \psi(t_r)))^{\phi_t(q^*(t_{r-1}, t_r)}{\pi_0(t_{r-1})P(G|t_{r-1})\prod_{m=1}^{M}\mathbb{P}(b_m(t_{r-1})|\sigma_{e(r-1)}, \psi(t_{r-1)))^{\phi_t(q(t_r, t_{r-1})}}\right\}. \quad (21)$$

### 6.8 APPENDIX 3: Efficient sampling of random effects $b_m$ and $\sigma^2_e$ in SMC

#### 6.8.1 Sampling of the environmental effect $b_m$

The posterior distribution of $b_m$ is a multivariate normal distribution.

$$b_m(t_{r-1}) | \beta_{m(t_{r-1})}, \sigma^2_{e(r-1)}, \sigma^2_{g(r)} \sim MVN(\mu_{b_m(t_{r-1})}, \Sigma_{b_m(t_{r-1})}).$$

Here $\Sigma_{b_m(t_{r-1})} = (I\sigma^{-2}_{g(r)} + \phi_r \psi(t_r)^{-1}\sigma^{-2}_{e(r)})^{-1}$, $\mu_{b_m(t_{r-1})} = \Sigma_{b_m(t_{r-1})}(y - G\phi_{b_m(t_{r-1})})$. The Woodbury matrix identity is

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$ 

Here we let $A = I\sigma^{-2}_{g(r)}$, we conduct eigen-decomposition for $\psi(t_r) = QA^{-1}$, then $\psi(t_r)^{-1} = Q\Lambda^{-1}Q^{-1}$. We let $U = Q$, $V = Q^{-1}$ and $C = \Lambda^{-1}\sigma^{-2}_{e(r)}$, the formula can be simplified as

$$\Sigma_{b_m(t_{r-1})} = (I\sigma^{-2}_{g(r)} + Q\phi_r\Lambda^{-1}\sigma^{-2}_{e(r)}Q^{-1})^{-1}$$

$$= I\sigma^{-2}_{g(r)} - \sigma^{-4}_{g(r)}Q(\phi_r^{-1}\Lambda\sigma^{-2}_{e(r)} + I\sigma^{-2}_{g(r)})^{-1}Q^{-1}$$

$$= \sigma^{-2}_{g(r)}Q[I - (\phi_r^{-1}\Lambda\sigma^{-2}_{e(r)}/\sigma^{-2}_{g(r)} + I)^{-1}]Q^{-1}.$$ 

Hence, we first simulate $z_m \sim MVN(0, I)$, then let $b_m = \mu_{b_m(t_{r-1})} + \sigma_{g(r)}Q[I - (\phi_r^{-1}\Lambda\sigma^{-2}_{e(r)}/\sigma^{-2}_{g(r)} + I)^{-1}]^{1/2}z_m$.

#### 6.8.2 Sampling of the environmental variance $\sigma^2_e$

The sampling of $\sigma^2_e$ is computationally expensive. The computation of $b_0^*$ in Equation (17) involves the computation of $\sum_{m=1}^{M} b'_{m(t_{r-1})} \psi(t)^{-1} b_{m(t_{r-1})}/2$. The cost of computing this quantity naively is $O(2MN^2)$. If we use the following formula:

$$\sum_{m=1}^{M} b'_{m(t_{r-1})} \psi(t)^{-1} b_{m(t_{r-1})} = \frac{\psi(t)^{-1}(\sum_{m=1}^{M} b_{m(t_{r-1})}b'_{m(t_{r-1})})}{2},$$

then the computational cost is reduced to $O(\frac{MN^2}{2})$. In GWAS study, the number of SNPs included in the analysis is generally large. This transformation can reduce the computational cost for approximately four times in sampling $\sigma^2_e$.
6.9 APPENDIX 4: Resampling step of sequential Monte Carlo

SMC algorithms suffer from the path degeneracy issue Doucet and Johansen (2009). As \( r \) increases, the approximations of the marginal distributions \( \pi_r(\Omega) \) may collapse. Resampling algorithms are commonly used to alleviate path degeneracy issues in SMC, in which particle weights are dominated by a single path through the particles. Resampling steps increase the asymptotic variance of SMC estimators, and so often SMC practitioners only conduct resampling step when necessary. The standard measure of particle degeneracy used for this purpose is the relative effective sample size (rESS), defined as:

\[
\text{ESS}(W_r) = \left( K \sum_{k=1}^{K} W_{r,k}^2 \right)^{-1}.
\]

(22)

6.10 APPENDIX 5: Parallelization of SMC

We conduct an experiment to illustrate parallelisation of the propagation and weighting steps of our SMC inference. Table 6.2 displays the computing time for Scenario C against the number of threads used in the parallelisation. For each condition, we used \( K = 500 \) particles and \( R = 100 \) intermediate targets. The results indicate that increasing the number threads leads to a notable speed increase. These experiments were conducted on a 1.3 GHz Intel Core i5 processor.

Table 6.2: TreeLMM runtime (second), with 50%, 2.5% and 97.5% quantiles, for 1 to 4 threads.

<table>
<thead>
<tr>
<th>#Threads</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>50.1 (47.8, 80.5)</td>
<td>26.7 (24.5, 42.2)</td>
<td>23.6 (22.5, 38.0)</td>
<td>22.4 (21.0, 33.9)</td>
</tr>
</tbody>
</table>
Chapter 7

Adaptive semiparametric Bayesian differential equations via sequential Monte Carlo methods

7.1 Introduction

Nonlinear differential equations (i.e. nonlinear ordinary or delay differential equations) are commonly used in modelling dynamic systems in ecology, physics and engineering. Delay differential equations (DDEs) are described by equations $dx(t)/dt = g(x(t), x(t-\tau)|\theta)$. These are continuous time models for interactions between variables $x(t)$, and a time delay $\tau$. Ordinary differential equations (ODEs) are often presented by $dx(t)/dt = g(x(t)|\theta)$, which can be taken to be a special case of DDEs with $\tau = 0$. The form of $g(\cdot)$ is generally proposed by specialists with scientific intuition. For example, ecologists proposed the simple Lotka-Volterra model (Rosenzweig and MacArthur, 1963) to understand and predict the population of predators and preys in ecosystems. Given a concrete form of the function $g(x(t), x(t-\tau)|\theta)$, the parameters $\theta$ and $\tau$ are unknown and required to be estimated using $y(t)$ observed at some data points. We say that a DE is observed with measurement error. The observed $y(t)$ is often assumed to link with variables $x(t)$ though a linear regression such that $y(t) = x(t) + \epsilon$, where $\epsilon$ is measurement error. The estimation of parameters in DEs is of great interest and requires us to solve the DEs $dx(t)/dt = g(x(t), x(t-\tau)|\theta)$.

Most DE systems do not admit an analytic solution. One solution is to solve the DEs numerically (Butcher, 2016), for example by using the Euler method (Jain, 1979; Bulirsch and Stoer, 1966), the Exponential integrators (Hochbruck et al., 1998; Hochbruck and Ostermann, 2010) or the Runge-Kutta method (Jameson et al., 1981; Ascher et al., 1997). However, numerical DE solvers are computationally expensive, especially for DDEs. Various methods have been proposed to solve DEs more efficiently in recent decades. The idea of using smoothing splines to fit dynamic data was first proposed by Varah (1982). Ramsay and Silverman (2007), Poyton et al. (2006), Chen and Wu (2008) extend the idea of smoothing to a two-stage approach. In the first stage, spline coefficients are optimized by minimizing the sum of the squared distances between the data and the spline functions.
at the observation times. In the second stage, with the estimated spline coefficients, DE parameters are optimized by minimizing the residuals of DE models. The two-stage approach may lead to inconsistent estimates. Ramsay et al. (2007) proposed a generalized smoothing approach, called “parameter cascading”, based on data smoothing methods and a generalization of profiled estimation. In the proposed approach, the spline coefficients are treated as nuisance parameters. Their method iterates between optimizing the objective function with spline coefficients given parameter estimated so far, and optimizing the objective function with parameters given the estimated spline coefficients. This iterate is repeated until convergence is achieved. The parameter estimates are consistent and asymptotically normally distributed under mild conditions (Pang et al., 2017). There are several variates for the parameter cascading approach. Cao et al. (2011) proposed a robust algorithm to estimate measurements with outlier based on smoothing splines. Cao et al. (2012) proposed a method to estimate time-varying parameter in ODEs, in which the ODE parameters are also modelled by smoothing splines. Wang and Cao (2012) defined a semiparametric method with smoothing spline to estimate DDE parameters.

Using smoothing splines to model DEs are computationally efficient since we do not require to numerically solve DEs. Most methods based on data smoothing to estimate parameters of DEs are derived from a frequentist perspective, which means they are only able to provide point estimates for the DE parameters. Bayesian methods are of interest since they provide the uncertainty of parameters. Campbell and Steele (2012) propose a smooth functional tempering algorithm to conduct posterior inference for ODEs parameters. This idea originates from parallel tempering and model based smoothing. Zhang et al. (2017) propose a high dimensional linear ordinary differential equation (ODE) model to accommodate the directional interaction in brain areas. Parallelized schemes for Markov chain Monte Carlo have been proposed to estimate the model. Bhaumik et al. (2015) investigated a two-stage procedure to estimate the parameter by minimizing the penalized ODEs.

There are several lines of work involved in estimating DEs parameter from a Bayesian perspective based on numerical DE solver. Dass et al. (2017) proposed a two-step approach to approximate posterior distributions for parameters of interest. They first apply a numerical algorithm to solve ODEs, then integrate nuisance parameters using Laplace approximations. Bhaumik et al. (2017) proposed a modification of Bhattacharya et al. (2015) by directly considering the distance between the function in the nonparametric model and that obtained from a four stage Runge-Kutta (RK4) method. Calderhead et al. (2009) presented a novel Bayesian sampler to infer parameters in nonlinear delay differential equations, the derivatives and time delay parameters are estimated via Gaussian processes. To make the DE estimation more consistent, Dondelinger et al. (2013) proposed an adaptive gradient matching approach to jointly infer the hyperparameters of a Gaussian process as well as ODE parameters. Barber and Wang (2014) simplified previous approaches by proposing a more natural generative model of data using Gaussian process, which directly links state derivative information with system observations.

Standard sequential Monte Carlo (SMC) methods (Doucet et al., 2001, 2000; Liu and Chen, 1998) are popular approaches for estimating dynamic models (e.g. state space models). SMC methods
combine importance sampling and resampling algorithms. Under mild conditions, consistency properties and asymptotic normality hold (Chopin, 2004). Del Moral et al. (2006) proposed a general SMC framework, to sample sequentially from a sequence of probability distributions that are defined on a common space. Several SMC methods have been proposed to estimate parameters in ODE models. Zhou et al. (2016) presented an adaptive sequential Monte Carlo (SMC) sampling strategy to estimate the model and parameters jointly, they use an ODE model selection example to demonstrate the proposed algorithm. Lee et al. (2018) introduced additive Gaussian error into numerically solved ODE trajectory, and they proposed a particle filter to infer ODE parameters. In addition, Gaussian process has been used to avoid numerical integration. These works are based on numerically solving ODE models.

In this article, we propose a more efficient Bayesian framework to estimate non-linear DEs. First, we propose a Bayesian semi-parametric approach via the idea of collocation to solve DEs to avoid expensive numerical solver, especially DDEs. The parameters of interest include the DEs parameters and basis coefficients of smoothing spline functions. Secondly, the tuning parameter selector is chosen adaptively via Bayesian approach, which avoids expensive cross-validation. Thirdly, we propose a sequential Monte Carlo algorithm to effectively sample parameters with multiple isolated posterior modes and parameters with high-dimensionality. Finally, we propose an adaptive scheme to select intermediate target distributions. Our numerical experiments demonstrate the effectiveness of our algorithm in estimating both ODEs and DDEs cases.

The rest of this chapter is organized as follows. In Section 7.2, we construct a fully Bayesian framework for non-linear DEs. In Section 7.3, we introduce our new algorithm for Bayesian non-linear DEs problem. In Section 7.4 and Section 7.5, we use numerical experiments to show the effectiveness of our new method. We conclude in Section 7.6.

7.2 Hierarchical Bayesian differential equations

In this section, we introduce a hierarchical Bayesian structure for DE models. In Section 7.2.1, we introduce likelihood function for DEs. In Section 7.2.2, we construct a fully Bayesian structure for DE model. In Section 7.2.3, we introduce selection of tuning parameter $\lambda$.

7.2.1 DE models

We use $x(t) = (x_1(t), \ldots, x_l(t))^T$ to denote the DE variables, where $x_i(t)$ denotes the $i$-th DE variable, $l$ denotes the total number of DE variables. Each DE variable $x_i(t)$ is a dynamic process modelled with one differential equation

$$\frac{dx_i(t)}{dt} = g_i(x(t), x(t-\tau)|\theta),$$
$$x_i(0) = x_{i0},$$

(1)
where $\theta$ and $\tau$ denote the unknown parameters in the DE model, $\tau$ is the delay parameter in DDE model ($\tau = 0$ in ODE model), $x_i(0)$ is the initial condition for the $i$-th DE variable, which is also unknown and needs to be estimated. Delay differential equations (DDEs) are time-delayed systems, with delay parameter $\tau$. The time delay in DDEs consider the dependence of the present state of the DE variable based on its past state. In DDEs, $x_i(t - \tau) = x_{i0}$ for $t < \tau$. We refer readers to Section 7.4 for a more detailed description of DDE models.

We do not observe the DEs directly, instead we observe them with measurement error. We let $y_i = (y_{i1}, \ldots, y_{ij})'$ denote the observations for the $i$-th DE trajectory. The $j$-th observation of $y_i$ is assumed to be

$$y_{ij} \sim N(x_i(t_j|\theta, \tau, x_{i0}), \sigma_i^2),$$

where $x_i(t_j|\theta, \tau, x_{i0})$ denotes the DE solution given $\theta$, $\tau$ and initial condition $x_{i0}$.

The joint likelihood function of $\theta$, $\tau$, $x(0)$ and $\sigma_i^2$ admits the following form

$$L(\theta, \tau, x(0), \sigma_i^2) = \prod_{i=1}^{I} \prod_{j=1}^{J} \left(\sigma_i^2\right)^{-1/2} \exp\left\{-\frac{(y_{ij} - x_i(t_j|\theta, \tau, x(0)))^2}{2\sigma_i^2}\right\}. \quad (2)$$

We use a figure (see Figure 7.1 (b)) to show an example of the log-likelihood surface over DEs parameters $\theta$, and for the setup of this model we refer to Section 7.5.1. The log-likelihood surface for $\theta$ has multiple isolated modes, and it is very sensitive to different parameter values.

![Log-likelihood surface for DE model](image)

Figure 7.1: (a) Graphical representation of DEs, (b) Log-likelihood surface for a DE model.

### 7.2.2 A fully Bayesian structure for DE model

Numerically solving DEs is computationally very intensive, especially for DDE models. In this article, we propose to solve the differential equation model by penalized smoothing. We represent the $i$-th DE function $x_i(t)$ as a linear combination of $L_i$ B-spline basis functions $\Phi(t) = (\phi_1(t), \phi_2(t), \ldots, \phi_{L_i}(t))'$ (see Figure 7.2 for an example of cubic B-spline functions (Ramsay,
Figure 7.2: The thirteen B-spline basis functions defined on \([0, 1]\) with degree three and nine equally spaced knots.

2004; De Boor, 1972),

\[ x_i(t) = \Phi(t)'c_i, \]

where \(c_i\) denotes the basis coefficients. The initial condition for the \(i\)-th DE function is \(x_i(0) = \Phi(0)'c_i\).

One advantage of using smoothing spline functions to model DE trajectories is that we can avoid the estimation of initial condition \(x(0)\). Figure 7.1 (a) represents the graphical structure for the proposed DE model. The unknown parameters in our DE model include spline coefficients \(c_i\), the delay time parameter \(\tau\) (which is known in ODE with \(\tau = 0\)), the DE parameter \(\theta\), and variance parameter \(\sigma_i^2\).

In Bayesian smoothing approaches, we give \(x(t)\) a prior density proportional to the “partially improper” Gaussian process (Berry et al., 2002). We let \(\lambda\) denote the smoothing penalty parameter. Given \(\lambda\), the prior distribution for \(x(t)\) is

\[
p(x(t)|\theta, c, \tau) \propto \exp\left\{-\frac{\lambda}{2} \sum_{i=1}^{I} \int_{t_i}^{t_f} \left[ \frac{dx_i(s)}{ds} - g(x(s), x(s-\tau)|\theta) \right]^2 ds\right\},
\]

\[
\propto \exp\left\{-\frac{\lambda}{2} \sum_{i=1}^{I} \int_{t_i}^{t_f} \left[ \frac{d\Phi(s)'}{ds}c_i - g_i(\Phi(s)', \Phi(s-\tau)'c|\theta) \right]^2 ds\right\},
\]

where \(\Phi(s-\tau) = \Phi(0)\) if \(s < \tau\). This prior distribution measures how well the estimated DE variables \(\hat{x}(t)\) satisfy the DE system. Details on selecting proper \(\lambda\) will be discussed in Section 7.2.3.
In the fully Bayesian framework, we need to assign appropriate priors for model parameters. We choose proper priors on parameters $\theta$, $\tau$, $c_i$, $\sigma_i^2$, $i = 1, \ldots, I$.

\[
\theta \sim MVN(0_D, \sigma_\theta^2 I_D),
\]

\[
\tau \sim U(t_1, t_f),
\]

\[
c_i \sim MVN(0_L, \sigma_c^2 I_L), \quad i = 1, \ldots, I,
\]

\[
\sigma_i^2 \sim IG(g_0, h_0), \quad i = 1, \ldots, I,
\]

where $\sigma_\theta^2$, $\sigma_c^2$, $g_0$ and $h_0$ are the hyper-parameters in prior distributions, $D$ is the dimension of $\theta$.

We introduce a new notation $\beta = (\tau, \theta, c, \sigma)'$ to denote all the parameters of interest. We use $\pi$ to denote the normalized posterior distribution for $\beta$. Let $p(y|\beta)$ denote the penalized likelihood function as follows

\[
p(y|\beta) \propto \prod_{i=1}^I \prod_{j=1}^J \sigma_i^2 \exp \left\{ -\sum_{i=1}^I \left( \sum_{j=1}^J \frac{(y_{ij} - \Phi(t_{ij})c_i)^2}{2\sigma_i^2} + \lambda \int_{t_i}^{t_{i+1}} \left[ \frac{d\Phi(s)'}{ds} c_i - g_i(\Phi(s)'c, \Phi(s-t)'c|\theta) \right]^2 ds \right) \right\}.
\]

The integral

\[
R_{ij} = \int_{t_i}^{t_{i+1}} \left[ \frac{d\Phi(s)'}{ds} c_i - g_i(\Phi(s)'c, \Phi(s-t)'c|\theta) \right]^2 ds.
\]

usually does not have a closed form expression. However it can be evaluated by numerical quadrature approximation. We approximate the integral by using the composite Simpson’s rule (Burden et al., 2001)

\[
R_{ij} = \sum_{m=1}^M v_{jm} \cdot \left( \left[ \frac{d\Phi(s)'}{ds} c_i - g_i(\Phi(s)'c, \Phi(s-t)'c|\theta) \right]^2 \right)_{s=t_{jm}},
\]

where $M$ is the number of quadrature points, $t_{jm}$ is the $m$-th quadrature point in $[t_i, t_{i+1}]$, and $v_{jm}$ is the corresponding quadrature weight.

Let $\pi_0(\beta)$ denote the prior distribution. We are interested in the normalized posterior for $\beta$

\[
\pi(\beta) = \frac{\gamma(\beta)}{Z} = \frac{\pi_0(\beta)p(y|\beta)}{Z},
\]

where $\gamma(\beta)$ is the unnormalized posterior distribution of $\beta$, $Z$ is the normalizing constant. The unnormalized posterior distribution of $\beta$ can be written as

\[
\gamma(\beta) = \prod_{i=1}^I \prod_{j=1}^J (\sigma_i^2)^{-1/2} \exp \left\{ -\sum_{i=1}^I \left( \sum_{j=1}^J \frac{(y_{ij} - \Phi(t_{ij})c_i)^2}{2\sigma_i^2} + \lambda \int_{t_i}^{t_{i+1}} \left[ \frac{d\Phi(s)'}{ds} c_i - g_i(\Phi(s)'c, \Phi(s-t)'c|\theta) \right]^2 ds \right) \right\} \prod_{i=1}^I (\sigma_i^2)^{-g_0 - 1} \exp \left\{ -\sum_{i=1}^I \frac{h_0}{\sigma_i^2} \cdot \exp \left\{ -\sum_{i=1}^I \frac{c_i^2}{\sigma_i^2} \right\} \exp \left\{ -\frac{\theta\theta'}{\sigma_\theta^2} \right\} \right\}.
\]

The normalizing constant

\[
Z = \int \gamma(\beta) d\beta
\]

is intractable.
7.2.3 The choice of $\lambda$

The tuning parameter $\lambda$ is important in balancing the data information and the DE fitting. A small value of $\lambda$ does not impose much information about the DE fitting. If $\lambda \to 0$, we end up fitting least squares for spline coefficients with the data. If we choose a large value of $\lambda$, then the prior information of DE system is too strong and not much information about the data is taken into consideration. Hence, it is crucial to choose a proper value of $\lambda$ to balance the DE fitting and data information.

One approach to choose $\lambda$ is through cross-validation (Wang and Cao, 2012; Reiss and Todd Ogden, 2009) from a range of reasonable choices of $\lambda$. However, this approach is not feasible in Bayesian frameworks as it increases computational cost. We propose another approach for selection of $\lambda$. We treat $\lambda$ as an unknown parameter, adding a prior distribution on $\lambda$ and working with a fully Bayesian version of DEs. This idea is adapted from Berry et al. (2002), in which they automatically select a smoothing parameter for splines. The prior distribution for the smoothing parameter is $\text{Gamma}(a, b)$.

7.3 Methodology

One classical methodology for Bayesian inference of nonlinear DE parameters is Markov chain Monte Carlo (MCMC). In MCMC, we construct an ergodic Markov chain which admits the normalized posterior as its stationary distribution. If we run the chain long enough, convergence to the posterior is guaranteed. We show the details of this method in the Appendix.

However, Markov chain Monte Carlo (more specifically, the Metropolis Hastings (MH) algorithm) is inefficient for estimating parameters of non-linear DEs for several reasons. First, the posterior surface is extremely sensitive to DE parameters $\theta$. There may exist isolated modes in the posterior distribution. The posterior may change quite a bit even with a tiny change in parameter value. Second, the computation of likelihood function involves numerically solving nonlinear DEs, which is computationally expensive. Third, the convergence of MCMC is generally hard to assess.

In Section 7.3.1, we propose an SMC method for nonlinear DE inference based on the Bayesian hierarchical structure proposed in Section 7.2. In Section 7.3.2, we introduce an advanced scheme to adaptively determine the intermediate target distributions in SMC.

7.3.1 An annealed sequential Monte Carlo for Bayesian DE inference.

To better cope with the inadequates of MCMC, we propose a sequential Monte Carlo (SMC) algorithm in the general SMC (Del Moral et al., 2006) framework for Bayesian DEs. The general SMC is a generic method to approximate a sequence of intermediate probability distributions $\{\pi_r(\beta)\}_{0 \leq r \leq R}$ defined on a common measurable space $(E, \mathcal{E})$. This method is different from the standard SMC algorithm (Doucet et al., 2000, 2001), as the sequence of intermediate probability distributions $\{\pi_r(\beta)\}_{0 \leq r \leq R}$ in standard SMC are generally defined on measurable spaces with incremental dimen-
With this backward kernel, the importance weight becomes

\[
W_{k,r+1} \propto W_{k,r} \cdot \frac{\pi_{r+1}(\beta_{k,r+1}) L_r(\beta_{k,r+1}, \beta_{k,r})}{\pi_r(\beta_{k,r}) T_{r+1}(\beta_{k,r}, \beta_{k,r+1})},
\]

where \(L_r(\beta_{k,r+1}, \beta_{k,r})\) is the artificial backward kernel (Del Moral et al., 2006, 2012), denoting the probability of moving from \(\beta_{k,r+1}\) to \(\beta_{k,r}\). The selection of this backward kernel is important as it will impact the variance of \(\{W_{k,r+1}\}_{k=1, \ldots, K}\). We refer readers to Del Moral et al. (2006) for more detailed discussion of this artificial backward kernel. One typical approach in general SMC framework is to select \(T_{r+1}(\beta_{k,r}, \beta_{k,r+1})\) to be a \(\pi_{r+1}\)-invariant MCMC kernel. A convenient backward Markov kernel that allows an easy evaluation of the importance weight is

\[
L_r(\beta_{k,r+1}, \beta_{k,r}) = \frac{\pi_{r+1}(\beta_{k,r}) T_{r+1}(\beta_{k,r}, \beta_{k,r+1})}{\pi_{r+1}(\beta_{k,r+1})}.
\]

With this backward kernel, the importance weight becomes

\[
W_{k,r+1} \propto W_{k,r} \cdot \frac{\pi_{r+1}(\beta_{k,r+1}) L_r(\beta_{k,r+1}, \beta_{k,r})}{\pi_r(\beta_{k,r}) T_{r+1}(\beta_{k,r}, \beta_{k,r+1})}.
\]

\[
= W_{k,r} \cdot \frac{\pi_{r+1}(\beta_{k,r+1})}{\pi_r(\beta_{k,r})} \cdot \frac{\pi_{r+1}(\beta_{k,r}) T_{r+1}(\beta_{k,r}, \beta_{k,r+1})}{\pi_{r+1}(\beta_{k,r+1})} \cdot \frac{1}{T_{r+1}(\beta_{k,r}, \beta_{k,r+1})}
\]

\[
\propto W_{k,r} \cdot \frac{\gamma_{r+1}(\beta_{k,r+1})}{\gamma_r(\beta_{k,r})}.
\]

Thus, we do not require pointwise evaluation of the forward kernel \(T_{r+1}(\beta_{k,r}, \beta_{k,r+1})\) and the backward kernel \(L_r(\beta_{k,r+1}, \beta_{k,r})\) to compute the weight function.

In this article, we propose a sequence of annealing intermediate target distributions (Neal, 2001; Wang et al., 2019) \(\{\pi_r(\beta)\}_{0 \leq r \leq R}\) to facilitate the exploration of posterior space, such that

\[
\pi_r(\beta) \propto \gamma_r(\beta) = p(y | \beta)^{\alpha_r} \pi_0(\beta),
\]

where \(0 = \alpha_0 < \alpha_1 < \cdots < \alpha_{R-1} < \alpha_R = 1\) is the sequence of annealing parameters. If there are isolated modes in \(\pi(\beta)\), MCMC may get stuck in one of the modes which is close to the initial value. Introducing a series of powered posterior distribution is to avoid this. With small annealing parameter \(\alpha_r\), the intermediate posterior surface is flat, which makes samples easier to move across modes. The intermediate posterior target with a higher value of annealing parameter is closer to the true posterior. The samples move closer to the target posterior distribution if we increase \(\alpha_r\). One
simple choice of annealing parameters is to equally put parameters across $[0, 1]$, such that $\alpha_0 = 0$, $\alpha_1 = 1/R$, $\alpha_2 = 2/R$, ... , $\alpha_{R-1} = (R-1)/R, \alpha_R = 1$.

We now introduce an SMC algorithm with a defined sequence of intermediate targets. First, we initialize particles $\{\beta_{k,0}\}_{k=1,2,...,K}$. At each step $r - 1$, we keep a list of $K$ particles $\{\beta_{k,r-1}\}_{k=1,2,...,K}$ in memory. We let $\{\tilde{\beta}_{k,r-1}\}_{k=1,2,...,K}$ denote particles after resampling step (see Step 3). We iterate between the following three steps to obtain the approximated intermediate target posterior

$$\hat{\pi}_r(\beta) = \sum_{k=1}^{K} W_{k,r} \cdot \delta_{\beta_{k,r}}(\beta), (r = 1, \ldots, R).$$

**Step 1.** We compute the weights function for particles at iteration $r$ with

$$W_{k,r} \propto W_{k,r-1} \cdot \frac{\gamma_r(\tilde{\beta}_{k,r-1})}{\gamma_{r-1}(\tilde{\beta}_{k,r-1})} = W_{k,r-1} \cdot p(y|\tilde{\beta}_{k,r-1})^{\alpha_r - \alpha_{r-1}}. \tag{7}$$

Note that the weight update function for particles at the $r$-th iteration only depends on particles at the $(r-1)$-th iteration, which is different from the standard SMC algorithm (Doucet et al., 2000, 2001).

**Step 2.** We propagate new samples $\{\beta_{k,r}\}_{k=1,...,K}$ via $\pi_r$-invariant MCMC moves, $\{\beta_{k,r} \sim T_r(\tilde{\beta}_{k,r-1}, \cdot)\}_{k=1,...,K}$. The conditional posterior distributions, $\pi_r(\sigma_i^2|c_i)$, $\pi_r(\tau|c, \theta)$, $\pi_r(\theta|c, \tau)$ and $\pi_r(c_i|\tau, \theta, \sigma, c_{-i})$ admit the following forms

- The full conditional distribution for $\sigma_i^2$ admits $\pi_r$-invariant is

$$\sigma_i^2 | c_i \sim IG\left( g_0 + \frac{J}{2}, h_0 + \frac{\alpha_r}{2} \sum_{j=1}^{J} (y_{ij} - \Phi(t_{ij})'c_i)^2 \right). \tag{8}$$

- The conditional distribution $\pi_r(\tau|c, \theta)$ does not admit closed form. We conduct random walk MH with Gaussian kernel with

$$\gamma_r(\tau|c, \theta) \propto \exp \left\{ -\alpha_r \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{m=1}^{M} v_{jm} \cdot \left( \frac{d\Phi(s)'c_i - g_i(\Phi(s)'c_i\theta) - \Phi(s - \tau'c_i\theta)}{d\tau} \right)^2 \right\}. \tag{9}$$

We use an example to illustrate the procedure of a random walk MH algorithm with a Gaussian kernel to propose $\tau$,

1. $\tau^* \sim N(\tau^{(n)}, \sigma_\tau^2)$,
2. compute the acceptance probability

$$p_{MH} = \min \left\{ 1, \frac{\gamma_r(\tau^*|c, \theta, \sigma^2)}{\gamma_r(\tau^{(n)}|c, \theta, \sigma^2)} \right\},$$

3. sample $u \sim U(0, 1)$, we accept $\tau^{(n+1)} = \tau^*$ if $u < p_{MH}$, otherwise we set $\tau^{(n+1)} = \tau^{(n)}$. 121
• The existence of closed form conditional posterior distributions \( \pi_r(c_i|\theta, \sigma, c_{-i}) \) depends on \( g_i \) \((i = 1, 2, \ldots, I)\). If all \( g_i \) are linear functions of \( \theta \), there exists closed form conditional posterior distribution \( \pi_r(c_i|\theta, \sigma, c_{-i}) \), which is Gaussian distributed. Otherwise, we conduct random walk MH algorithm with a Gaussian kernel.

• If the conditional distribution \( \pi_r(c_i|\theta, \sigma, c_{-i}) \) does not admit closed form. We conduct random walk MH with a Gaussian kernel with

\[
\gamma_r(c_i|\theta, \sigma, c_{-i}) \propto \exp\left( -\alpha_r \sum_{i=1}^I \sum_{j=1}^J \left( \frac{(y_{ij} - \Phi(i,j)c_i)^2}{2\sigma_i^2} \right) + \frac{1}{2} \sum_{m=1}^M \sum_{j=1}^J \sum_{m=1}^M \sum_{m=1}^M v_{jm} \cdot \left( \left[ \frac{d\Phi(s)^'}{ds}c_i - g_i(\Phi(s)'c, \Phi(s - \tau')c|\theta) \right]_{s=t_{jm}}^2 \right) \right).
\]

• If the conditional distribution \( \pi_r(\theta|c, \tau) \) does not admit closed form. We conduct random walk MH with a Gaussian kernel with

\[
\gamma_r(\theta|c, \tau) \propto \exp\left( -\alpha_r \sum_{i=1}^I \sum_{j=1}^J \left( \frac{1}{2} \sum_{m=1}^M \sum_{j=1}^J \sum_{m=1}^M \sum_{m=1}^M v_{jm} \cdot \left( \left[ \frac{d\Phi(s)^'}{ds}c_i - g_i(\Phi(s)'c, \Phi(s - \tau')c|\theta) \right]_{s=t_{jm}}^2 \right) \right) \right).
\]

• The conditional posterior distribution of \( \lambda \) is \( \text{Gamma}(a_\lambda + \alpha_r \sum_{i=1}^I (L_i - 2)/2, b_\lambda') \), where

\[
\frac{1}{b_\lambda'} = \frac{1}{b_\lambda} + \frac{\alpha_r}{2} \sum_{i=1}^I \sum_{j=1}^J \sum_{m=1}^M v_{jm} \cdot \left( \left[ \frac{d\Phi(s)^'}{ds}c_i - g_i(\Phi(s)'c, \Phi(s - \tau')c|\theta) \right]_{s=t_{jm}}^2 \right).
\]

Step 3. We conduct a resampling step to prune particles with small weights. The particles after resampling step are denoted by \( \{\tilde{\beta}_{k,r}\}_{k=1,\ldots,K} \). The resampling scheme is typically triggered when the ESS falls below a given thresholds \( \epsilon \). The effective sample size (ESS) at iteration \( r \) can be computed by

\[
\text{ESS}_{K,r} = \frac{1}{\sum_{k=1}^K (W_{k,r})^2}.
\]

ESS\(_{K,r}\) denotes the number of ‘perfect’ samples used to approximate the intermediate distribution \( \pi_r \). Effective sample size takes value between 1 and \( K \). It takes value \( K \) if all particles are equally weighted, and it takes value 1 if one of particles has much larger weights compared with rest. We recommend do not conduct resampling at every iteration as resampling will creates additional variation to the estimator (Chopin, 2004). We do not recommend to resample at last SMC iteration. Advanced resampling schemes such as stratified resampling (Hol et al., 2006), residual resampling (Douc and Cappé, 2005) are more preferable than multinomial resampling, since multinomial resampling will create more variance for the SMC estimator compared with advanced resampling algorithms. In our numerical experiments, we use systematic resampling. After resampling step, all particles are equally weighted. If we never conduct resampling, the annealed SMC algorithm degenerates to the annealed importance sampling (Neal, 2001).
After conducting the annealed SMC algorithm, we obtain a list of weighted samples to empirically represent the posterior distribution $\pi(\beta)$,

$$\hat{\pi}(\beta) = \sum_{k=1}^{K} W_{k,R} \cdot \delta_{\beta_{k,R}}(\beta).$$

### 7.3.2 Adaptive annealing parameter scheme in SMC

In the annealed SMC algorithm, one challenge is to properly select the sequence annealing parameters. If we choose $\alpha_0 = 0$ and $\alpha_0 = 1$, the annealed SMC degenerates to importance sampling. A large number of annealing parameters improves the performance of algorithm, but it will be computationally more intensive. If we select an insufficient number of annealing parameters or an improper annealing scheme, the algorithm may collapse. We propose an adaptive annealing parameter scheme based on the seminar work of Del Moral et al. (2012); Zhou et al. (2016); Wang et al. (2019).

Note that the weight function (Equation 7) for iteration $r$ only depends on particles of $(r-1)$-th iteration, and the difference between two annealing parameters $\alpha_r - \alpha_{r-1}$. This indicates that we can ‘manipulate’ $w_{r,k}$ by changing the annealing parameter $\alpha_r$. If $\alpha_r$ is close to $\alpha_{r-1}$, the incremental weight function $w_{k,r} = p(y|\tilde{\beta}_{k,r-1})^{\alpha_r-\alpha_{r-1}}$ is close to 1, and the variance of $w_{k,r}$ is smaller than it would be if are chosen a larger value of $\alpha_r$. This provides the intuition that we are able to control the discrepancy between two intermediate targets by manipulating $\alpha_r$.

In this article, we use the relative conditional effective sample size (rCESS) (Zhou et al., 2016) to measure the discrepancy between two intermediate targets. The rCESS is defined as

$$\text{rCESS}_r = \frac{\left( \sum_{k=1}^{K} W_{k,r-1} W_{k,r} \right)^2}{\sum_{k=1}^{K} W_{k,r-1}(w_{k,r})^2},$$

which takes value between $1/K$ and 1. The rCESS is equal to relative ESS if we conduct resampling at every SMC iteration. Using the fact that $w_{k,r} = p(y|\tilde{\beta}_{k,r-1})^{\alpha_r-\alpha_{r-1}}$, rCESS$_r$ is a decreasing function of $\alpha_r$, where $\alpha_r \in [\alpha_{r-1}, 1]$. We control rCESS over iterations by selecting the annealing parameter $\alpha$ such that

$$f(\alpha) = \text{rCESS}(W_{k,r-1} \cdot p(y|\beta_{k,r-1})^{\alpha-\alpha_{r-1}}) = \phi,$$

where $\phi$ is a value between 0 and 1. A small value of $\phi$ will lead to a high value of $\alpha_r$, while a large value of $\phi$ will lead to a low value of $\alpha_r$. It is impossible to obtain a closed-form solution of $\alpha^*$ by solving $f(\alpha) = \phi$, but we are able to use bisection method to solve this one-dimensional search problem. The search interval of $\alpha$ is $(\alpha_{r-1}, 1]$. By using $f(\alpha_{r-1}) - \phi > 0$, $f(1) - \phi < 0$ (in case $f(1) \geq \phi$, we set $\alpha_r = 1$), and the continuous property of $f(\alpha) - \phi$. The solution $\alpha^*$ of $f(\alpha) = \phi$ is guaranteed. Algorithm 22 provides detailed description for the SMC algorithm.
Algorithm 22 A sequential Monte Carlo algorithm for Bayesian differential equations

1: Inputs: (a) Priors \( \pi_0 \) over model parameters \( \beta \), where \( \beta = (\theta, \tau, c, \sigma') \); (b) smoothing parameter \( \lambda \); (c) relative CESS \( \phi_k \), \( k = 1, \ldots, K \); (d) resampling threshold \( \epsilon \).
2: Outputs: Approximation of the posterior distribution, \( \hat{\pi}(\beta) = \sum^K_i W_{i,K} \cdot \pi_0(\beta) \).
3: Initialize SMC iteration index and annealing parameter: \( r \leftarrow 0, \alpha_0 \leftarrow 0 \).
4: for \( k \in \{1, 2, \ldots, K\} \) do
5: Initialize particles with independent samples: \( \beta_{k,0} \leftarrow (\theta_{k,0}, c_{k,0}, \sigma_{k,0})' \).
6: Initialize weights to unity: \( W_{0,k} \leftarrow 1/K \).
7: for \( r \in \{1, 2, \ldots\} \) do
8: Compute annealing parameter \( \alpha_r \), using bisection method with
\[
\begin{align*}
f(\alpha) &= r \text{CESS} (W_{k, r-1} \cdot p(y|\hat{\beta}_{k, r-1})^{\alpha_r}) = \phi_r.
\end{align*}
\]
9: for \( k \in \{1, 1, K\} \) do
10: Compute normalized weights for \( \beta_{k,r} \): \( W_{k,r} \propto W_{k,r-1} \cdot [p(y|\hat{\beta}_{k,r-1})]^{\alpha_r} \).
11: Sample particles \( \beta_{k,r} \) with one MCMC move admitting \( \pi_r \), as stationary (shown in Eq (8-10)).
12: if \( \phi_r = 1 \) then
13: return the current particle population \( (\beta_{k,r}, W_{k,r})_{k=1,\ldots,K} \).
14: else
15: if \( \text{rESS} < \epsilon \) then
16: Resample the particles.
17: for \( k \in \{1, 1, K\} \) do
18: Reset particle weights: \( W_{k,r} = 1/K \).
19: else
20: for \( k \in \{1, 1, K\} \) do
21: \( \hat{\beta}_{k,r} = \beta_{k,r} \).

7.4 Real data analysis

In the dynamic system of the blowfly population, resource limitation acts with a time delay, roughly equal to the time for a larva to grow up to an adult. Figure 7.3 displays the counts of blowflies over time studied in Nicholson (1954). The time unit is one day. The oscillations displayed in blowfly population is caused by the time lag between stimulus and reaction (Berezansky et al., 2010). May (1976) proposed to model the counts of blowflies with the following DDE model

\[
\frac{dx(t)}{dt} = rx(t)[1 - x(t - \tau)/(1000 \cdot P)],
\]

where \( x(t) \) is the blowfly population, \( r \) is the rate of increase of the blowfly population, \( P \) is a resource limitation parameter set by the supply of food, and \( \tau \) is the time delay, roughly equal to the time for a larva to grow up to an adult. Our goal is to estimate the initial value, \( x(0) \), and the three parameters, \( r, P, \) and \( \tau \), from the noisy Nicholson’s blowfly data \( y(t) \). The observed counts of blowflies \( y(t) \) is assumed to be lognormal distributed with mean \( x(t) \) and variance \( \sigma^2 \).

The counts of blowfly \( x(t) \) is a positive function. Instead of modelling the constrained function \( x(t) \) by a linear combination of cubic B-spline basis functions \( x(t) = \Phi(t)'c_t \), we transform \( x(t) = e^{W(t)} \) and use B-spline basis functions to model the unconstrained function \( W(t) = \Phi(t)'c_t \), equivalently we solve delay differential equation

\[
\frac{dW(t)}{dt} = r[1 - e^{W(t-\tau)/(1000 \cdot P)}],
\]

124
with noisy observations \( \log y(t) \sim N(W(t), \sigma^2) \).

We put 34 equally spaced interior knots over the time span. The total number of knots is equal to 36. The total number of cubic B-spline functions is \( L = 38 \). Our prior distributions for parameter of interest \((c, r, P, \tau, \sigma^2)\) are

\[
\begin{align*}
    r &\sim N(0, 5^2) I(r > 0),
    P &\sim N(0, 5^2) I(P > 0),
    \tau &\sim \text{Unif}(0, 50),
    c &\sim \text{MVN}(0_L, 100^2 I_L),
    \sigma^2 &\sim \text{IG}(1, 1),
    \lambda &\sim \text{Gamma}(1, 1).
\end{align*}
\]

In our adaptive SMC, we set \( r_{\text{CESS}} = 0.9 \) and resampling threshold \( \epsilon = 0.5 \). The number of particles is \( K = 500 \). Under this setting, the number of SMC iteration \( R = 227 \). Figure 7.4 displays the estimated DDE trajectory. The left panel of Figure 7.4 shows the estimated \( W(t) \) and the 95\% credible intervals; the right panel of Figure 7.4 shows \( X(t) = e^{W(t)} \), in which the blue points are observed data.

Table 7.3 displays the posterior mean and corresponding 95\% credible interval (CI) for DDE in Equation (13). Figure 7.5 displays the posterior samples of \( r, \tau \) and \( P \). We also calculate the correlation between posterior samples: \( \text{corr}(r, P) = 0.598 \), \( \text{corr}(r, \tau) = 0.139 \), \( \text{corr}(P, \tau) = 0.008 \). Recall that \( P \) is a resource limitation parameter, \( r \) is the rate of increase of the blowfly population, and \( \tau \) is the time delay, roughly equal to the time for a larva to grow up to an adult. The relative large positive correlation between \( r \) and \( P \) can be easily understood: the blowfly population grows faster when there is a larger food supply. The delay parameter \( \tau \) is the main parameter of interest. The posterior mean of the time for a larva to grow up to an adult and its 95\% CI for \( \tau \) is 8.368 (5.656, 9.916). The tiny positive value of the correlation between \( \tau \) and \( P \) implies that the amount of food supply has a small impact on the period of being a larva. The small positive value of the

Figure 7.3: Blowfly population in one experiment published in Nicholson (1954); the time unit is one day.
correlation between $\tau$ and $r$ indicates that the blowfly population will increase if larvae take their
time to well develop.

Table 7.1: Posterior mean and corresponding 95% credible interval (CI) for real data.

<table>
<thead>
<tr>
<th></th>
<th>$r$</th>
<th>$P$</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.176</td>
<td>2.372</td>
<td>8.368</td>
</tr>
<tr>
<td>(2.5%, 97.5%)</td>
<td>(0.074, 0.284)</td>
<td>(1.307, 3.333)</td>
<td>(5.656, 9.916)</td>
</tr>
<tr>
<td>$W(0)$</td>
<td>$\sigma^2$</td>
<td>$\lambda$</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>8.299</td>
<td>0.527</td>
<td>3.348</td>
</tr>
<tr>
<td>(2.5%, 97.5%)</td>
<td>(7.220, 9.355)</td>
<td>(0.460, 0.607)</td>
<td>(1.633, 5.910)</td>
</tr>
</tbody>
</table>

7.5 Simulation study

We use simulation study to demonstrate the effectiveness of our proposed model and method.
The experiments include both ODE and DDE examples. We use the R package deSolve (Soetaert et al., 2010) to simulate differential equations.

7.5.1 A non-linear ordinary differential equation example

In this section, we use a non-linear ODE example to illustrate the numerical behaviour of SMC algorithm. We generate ODE trajectories according to the following ODE system,

\[
\frac{dx_1(t)}{dt} = \frac{72}{36 + x_2(t)} - \theta_1, \\
\frac{dx_2(t)}{dt} = \theta_2 x_1(t) - 1,
\]

(14)
Figure 7.5: Posterior samples of $r$, $\tau$, $P$ for DDE in Equation (13) estimated via SMC. We resample the particles at last SMC iteration such that they are equally weighted.

Figure 7.6: Simulated ODE trajectories and observations. Red lines in Figure refer to simulated ODE trajectories and blue points refer to simulated observations.

where $\theta_1 = 2$ and $\theta_2 = 1$, and initial conditions $x_1(0) = 7$ and $x_2(0) = -10$. The observations $y_i$ are simulated from normal distribution with mean $x_i(t|\theta)$ and variance $\sigma_i^2$, where $\sigma_1 = 1$ and $\sigma_2 = 3$. We generate 121 observations for each ODE function, equally spaced within $[0, 60]$ (see Figure 7.6). Under this setting, the posterior distribution of $\theta_1$ and $\theta_2$ will have multiple local modes (see Figure 7.1).

We use cubic B-spline functions (see Figure 7.2) to represent ODE trajectories. We put equally spaced knots on each of eight observations. The total number of knots is 16, including 14 interior knots. The total number of cubic B-spline functions is $L = 18$. We select weak prior distributions of $\beta$ for SMC algorithm,

$$
\begin{align*}
\theta_1 &\sim N(5, 5^2), \quad \theta_2 \sim N(5, 5^2), \\
c_1 &\sim MVN(0_L, 100^2 I_L), \quad c_2 \sim MVN(0_L, 100^2 I_L), \\
\sigma_1^2 &\sim IG(1, 1), \quad \sigma_2^2 \sim IG(1, 1), \quad \lambda \sim Gamma(1, 1).
\end{align*}
$$
One bimodal example

We first alter Equation (14) to produce a symmetric, bimodal posterior for $\theta_1$, 

$$
\frac{dx_1(t)}{dt} = \frac{72}{36 + x_2(t)} - |\theta_1|, \quad \frac{dx_2(t)}{dt} = \theta_2 x_1(t) - 1.
$$

(15)

In our adaptive SMC, we set $\text{rCESS} = 0.9$ and resampling threshold $\epsilon = 0.5$. The total number of particles we use is $K = 500$. Under this setting, the number of annealing parameters is 752. We show the approximated intermediate posterior distribution for $\theta$ and $\sigma$ when $t = 1, 50, 300, 752$ (see two panels in the first row of Figure 7.7). With the increment of annealing parameters, the particles gradually move to the posterior distribution. We create two main modes for $\theta_1$ in Equation (15), the SMC algorithm is able to find these two global modes of $\theta_1$ while avoiding being stuck in local modes. We reported the estimated ODE trajectories and the 95% credible intervals in the two panels in the second row of Figure 7.7. The estimated mean ODE trajectories are very close to the true ODE trajectories. The 95% credible intervals covers the true ODE trajectories.
We compare SMC with MCMC illustrated in Appendix. With given samples $\theta^{(n)}, x(0)^{(n)}$ at one iteration, we use deSolve (Soetaert et al., 2010) to solve ODEs, to obtain $x(t_j|\theta^{(n)}, x(0)^{(n)})$. We select weak prior distributions for $\theta$ and $x(0)$. We run an MCMC algorithm with 10,000 iterations, which is close to $K \cdot R$ in SMC. The acceptance rate of MH algorithm is 25.2%, which is close to the “gold standard” MH acceptance rate 23.4%. Figure 7.8 displays the MCMC trace plots of $\theta$. It indicates that MCMC is getting stuck in local modes close to the initial value, and cannot explore the two main modes created in Equation (15).

**Comparison of using different rCESS and $K$**

We conduct experiments to investigate the performance of the SMC algorithm with different thresholds of rCESS and $K$ using the ODE system showed in Equation (14). We compare the performance of the SMC algorithm in terms of estimated $\theta$, $\sigma$ and estimated ODE. With estimated basis coefficients $\hat{c}_i$, we are able to compute the estimated $i$–th ODE trajectory $\hat{x}_i(t) = \Phi(t)^\prime \hat{c}_i$. We define the distance between the estimated $i$–th ODE trajectory $\hat{x}_i(t)$ and the true ODE $x_i(t)$ we used to simulate data as

$$
\text{RMSE}(x_i(t)) = \left[ \frac{1}{J} \sum_{j=1}^{J} (\Phi(t_j)^\prime \hat{c}_i - x_i(t_j))^2 \right]^{1/2}.
$$

(16)

We select three different levels of rCESS (rCESS = 0.8, 0.9, 0.99). We put equally space interior knots on each of 12 observations, the total number of basis function is 13. The number of particles we use is $K = 500$. For each level of rCESS, we repeat the SMC algorithm 20 times. Figure 7.9 displays boxplots of SMC with different rCESS threshold, in terms of $\theta$, $\sigma$ and RMSE$(x_i(t))$. It indicates that the parameter estimates get closer to true value, and RMSE of ODE trajectory gets smaller, when we increase rCESS thresholds. A higher value of rCESS threshold is equivalent to more intermediate target distributions.
We select three different levels of $K$ ($K = 10, 100, 2000$). We also put equally space interior knots and the total number of basis function is 13. We set rCESS = 0.9. For each level of $K$, we repeat SMC algorithm 20 times. Figure 7.10 displays boxplots of SMC with different level of $K$, in terms of $\theta$, $\sigma$ and RMSE($X_i(t)$). It indicates that the proposed SMC method performs better when we use a large number of particles. The consistency of the SMC algorithm holds when number of particles goes to infinity (Chopin, 2004; Wang et al., 2019; Del Moral et al., 2006). However, we cannot use an arbitrarily large value of $K$ as the computational cost of the SMC algorithm is a linear function of $K$. We recommend increasing rCESS in the SMC (using a larger number of intermediate distributions $R$), as increasing $R$ does not increase memory burden.

Number of basis functions and selection of $\lambda$

In this section, we first conduct experiments to investigate the performance of the SMC algorithm with different number of basis functions. We also compare SMC in terms of estimates of $\theta$, $\sigma$ and the RMSE of ODEs. The knots for basis function are equally spaced. We choose three different number of basis function, $\text{nbasis} = 7, 11, 16, 31, 61$. We set $K = 500$ and rCESS = 0.9 for the SMC algorithm. For each level of number of basis function, we repeat SMC 20 times. Figure 7.11 displays the ODE parameter estimates with different number of basis functions. The parameter estimates $\theta$ and $\sigma$ get closer to true value, and the RMSE of estimated ODE trajectory decreases if we increase
the number of basis functions from 7 to 16. However, the parameter estimates and RMSE of the estimated ODE trajectory become worse if we use too few basis functions. This experiment indicates a sufficient number of basis functions is important in ODE trajectory estimation. However, we do not recommend using a too large number of basis functions as it will cause over fitting and induce a heavy computational cost.

The second experiment we conduct is a comparison between the performance of the SMC algorithm with different choice of $\lambda$ ($\lambda = 0.1, 1, 10, 100$ and full Bayesian scheme). We put equally spaced knots and the number of basis functions we use is 16. We set $K = 500$ and rCESS = 0.9 for the SMC algorithm. For each choice of $\lambda$, we repeat SMC methods 20 times. Figure 7.11 displays the ODE parameter estimates with different choices of $\lambda$. The fully Bayesian scheme performs satisfactory in terms of parameter estimates and RMSE of the ODE trajectory. Figure 7.13 displays the posterior samples of $\lambda$ for one SMC replicate. The posterior samples lies within 0 and 1.

![Figure 7.11: ODE parameter estimates provided by different number of knots.](image1)

![Figure 7.12: ODE parameter estimates with different choices of $\lambda$.](image2)
7.5.2 Delay differential equation examples

Hutchinson’s equation

Our first DDE example is the Hutchinson’s equation, which is used to model the blowfly data in Section 7.4,

\[
\frac{dx(t)}{dt} = rx(t)[1 - x(t - \tau)/(1000 \cdot P)],
\]

where \( \tau, r \) and \( P \) are parameters of interest in the DDE. We set \( x(0) = 3500, \tau = 3, r = 0.8 \) and \( P = 2 \) to simulate DDE trajectory. The DDE trajectory is observed with measurement error. The error is lognormal distributed with mean 0 and standard deviation \( \sigma = 0.4 \). We simulate 3 data sets, with 101, 201 and 401 observations respectively, equally spaced in \([0, 100]\).

We transform the positive constraint function \( x(t) = e^{W(t)} \) and use B-spline basis functions to model the unconstrained one \( W(t) = \Phi(t)'c_i \). This is equivalent to solving the delay differential equation displayed in Equation (13). We put 51 knots equally spaced in \([0, 100]\), including 49 interior knots. The total number of cubic B-spline functions is \( L = 53 \). The hyper-parameters in DDE parameter prior and sequential Monte Carlo setups are same as Section 7.4. Table 7.2 displays the estimated parameters \((r, P, \tau)\) and RMSE defined in Equation (16) for \( W(t) \). For the same DDE function, a larger number of observations improves the performance of estimation.

A nonlinear delay differential equation example

In this section, we investigate a nonlinear delay differential equation model proposed by Monk (2003) to model the feedback inhibition of gene expression. The nonlinear DDE is described as follows:

\[
\begin{align*}
\frac{dx_1(t)}{dt} &= \frac{1}{1 + (x_2(t - \tau)/p_0)^n} - \mu m x_1(t), \\
\frac{dx_2(t)}{dt} &= x_1(t) - \mu p x_2(t).
\end{align*}
\]

(17)
Table 7.2: Estimated parameters and MSE of $W(t)$ for three simulated data sets.

<table>
<thead>
<tr>
<th>J</th>
<th>$r$</th>
<th>P</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>0.8</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>101</td>
<td>0.639 (0.429, 0.817)</td>
<td>2.072 (1.564, 2.607)</td>
<td>3.054 (2.664, 3.560)</td>
</tr>
<tr>
<td>201</td>
<td>0.729 (0.600, 0.903)</td>
<td>1.975 (1.655, 2.320)</td>
<td>3.020 (2.732, 3.256)</td>
</tr>
<tr>
<td>401</td>
<td>0.750 (0.632, 0.863)</td>
<td>2.086 (1.800, 2.383)</td>
<td>2.997 (2.843, 3.156)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>J</th>
<th>$W(0)$</th>
<th>$\sigma^2$</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>8.161</td>
<td>0.16</td>
<td>–</td>
</tr>
<tr>
<td>101</td>
<td>7.926 (7.343, 8.471)</td>
<td>0.119 (0.083, 0.184)</td>
<td>0.255</td>
</tr>
<tr>
<td>201</td>
<td>7.902 (7.529, 8.328)</td>
<td>0.141 (0.112, 0.179)</td>
<td>0.206</td>
</tr>
<tr>
<td>401</td>
<td>7.977 (7.569, 8.396)</td>
<td>0.166 (0.145, 0.193)</td>
<td>0.136</td>
</tr>
</tbody>
</table>

Figure 7.14: Simulated DDE trajectories and observations. Red lines in Figure refer to simulated DDE trajectory and blue points refer to simulated observations.

In Equation (17), $x_1(t)$ denotes the expression of mRNA at time $t$, $x_2(t)$ denotes the expression of a protein at time $t$. There is a delayed repression of mRNA production by the protein. The DDE system depends on the transcriptional delay $\tau$, and degradation rates $\mu_m$ and $\mu_p$, the expression threshold $p_0$ and the Hill coefficient $n$. As noted in Monk (2003), there is significant nonlinearity in the DDE system when the Hill coefficient $n > 4$.

We simulate a delay differential equation system with $\tau = 25$, $p_0 = 100$, $\mu_m = 0.03$, $\mu_p = 0.03$, $n$ is set to 8. The observations $y_i(t)$ are simulated from a normal distribution with mean $x_i(t|\theta)$ and variance $\sigma_i^2$, where $\sigma_1 = 1$ and $\sigma_2 = 5$. We generate 101 observations for each DDE function, equally spaced in $[0, 500]$. Figure 7.14 represents the simulated DDE system, which exhibits oscillations in mRNA and protein expression.
We allocate equally spaced knots within [0, 500], the total number of cubic B-spline basis functions we use is \( L = 28 \). We select the same weak prior distributions of \( \beta \) for the SMC algorithm,

\[
\theta_1 \sim N(0, 5^2), \quad \theta_2 \sim N(0, 5^2), \quad \tau \sim Unif(0, 50),
\]
\[
e_1 \sim MVN(0_L, 100^2 I_L), \quad e_2 \sim MVN(0_L, 100^2 I_L),
\]
\[
\sigma_1^2 \sim IG(1, 1), \quad \sigma_2^2 \sim IG(1, 1), \quad \lambda \sim Gamma(1, 1).
\]

In our adaptive SMC, we set rCESS = 0.9 and resampling threshold \( \epsilon = 0.5 \). The total number of particles we use is \( K = 300 \). Under this setting, the number of annealing parameters is \( R = 850 \). We show the parameter estimates and the corresponding 95% credible interval (CI) in Table 7.3. The mean of parameters are fairly close to the true value, and the 95% credible interval covers the true value. The estimated posterior mean of \( \lambda \) is 0.225.

Table 7.3: Parameter estimates and corresponding 95% credible interval (CI) for nonlinear DDE models.

<table>
<thead>
<tr>
<th></th>
<th>True</th>
<th>Mean</th>
<th>95% CI</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu_m )</td>
<td>0.03</td>
<td>0.0275</td>
<td>(0.0083, 0.0507)</td>
</tr>
<tr>
<td>( \mu_p )</td>
<td>0.03</td>
<td>0.0304</td>
<td>(0.0294, 0.0317)</td>
</tr>
<tr>
<td>( p_0 )</td>
<td>100</td>
<td>94.966</td>
<td>(68.494, 120.058)</td>
</tr>
<tr>
<td>( \tau )</td>
<td>25</td>
<td>24.739</td>
<td>(12.384, 34.736)</td>
</tr>
<tr>
<td>( \sigma_1 )</td>
<td>1</td>
<td>1.015</td>
<td>(0.872, 1.173)</td>
</tr>
<tr>
<td>( \sigma_2 )</td>
<td>5</td>
<td>4.576</td>
<td>(4.036, 5.247)</td>
</tr>
<tr>
<td>( x_1(0) )</td>
<td>7</td>
<td>6.874</td>
<td>(4.392, 9.137)</td>
</tr>
<tr>
<td>( x_2(0) )</td>
<td>-10</td>
<td>-10.745</td>
<td>(-16.664, -5.061)</td>
</tr>
</tbody>
</table>

We reported the estimated DDE trajectories and the 95% credible intervals in the upper two panels of Figure 7.15. The estimated mean DDE trajectories are generally very close to the true DDE trajectories. The 95% confidence intervals covers the true DDE trajectories.

For the same data set, we fit the DDE trajectories with a larger number of cubic B-spline basis functions \( L = 43 \), with equally spaced knots within [0, 500]. We use same algorithmic setting described above. The lower two panels of Figure 7.15 reported the estimated DDE trajectories and the 95% credible intervals, which exhibit overfit in DDE trajectories.

7.6 Discussion

Parameter estimation in nonlinear differential equations (e.g., ordinary differential equations, delay differential equations) is a challenge task. First, non-linear DEs generally do not have closed form analytic solutions, and require expensive numerical solvers. Second, the solutions to differential equations also depend on some unknown parameters and initial value of DEs systems. The likelihood surface of a DE is multi-modal and very sensitive to model parameters.
Figure 7.15: Estimated DDE trajectories and the 95% confidence interval using $L = 28$ basis functions (upper two panels) and using $L = 43$ basis functions (lower two panels).
In this article, we propose an adaptive semi-parametric Bayesian framework to solve DEs, this avoids expensive numerical approximation. We also avoid expensive tuning for global smoothing parameters. In addition, we develop a sequential Monte Carlo algorithm in an annealing framework to estimate the model parameters. The developed SMC method fully explores the multi-modal posterior surface of DEs parameters, and intermediate target distributions of SMC are adaptively determined by data and particles.

We use different simulation scenarios to explore the numerical behavior of our model and method, and demonstrated it can perform well in both ODE and DDE parameter estimation. We apply our method to a real data example to model the population dynamics of blowflies with a delay differential equation.

In all of our numerical experiments, we put equally spaced knots for smoothing splines and the number of knots are pre-determined before running experiments. In future work, we will explore use a smaller number of knots that are well placed, and let the number of knots and knots locations determined by data. The adaptive control of knots for smoothing spline in DEs will benefit the estimation of DEs with sharp changes. In practice, it is often the case that there are several DE models that can be used to describe the same dynamic system. This requires selection between differential equations models, and another avenue of future work is the exploration of DE model selection via sequential Monte Carlo methods.

**APPENDIX**

In this Appendix, we introduce classical Markov chain Monte Carlo algorithm for inference of parameters in Bayesian differential equations. We follow the notation described in Section 7.2. The joint likelihood function of $\theta$, $\tau$, $x(0)$ and $\sigma^2$ can be written as

$$L(\theta, \tau, x(0), \sigma^2) = \prod_{i=1}^{I} \prod_{j=1}^{J} (\sigma^2)^{-1/2} \exp \left\{ -\frac{(y_{ij} - x_{i}(t_{ij}|\theta, \tau, x(0)))^2}{2\sigma^2_i} \right\}. \quad (18)$$

To construct a Bayesian framework, we assign appropriate priors distributions for model parameters $\theta$, $\tau$, $x(0)$ and $\sigma^2$, denoted by $\pi_0(\theta)$, $\pi_0(\tau)$, $\pi_0(x(0))$ and $\pi_0(\sigma^2)$. The full conditional posterior distribution of $\sigma^2_i$ is Inverse-Gamma distributed. The full conditional posterior distributions of $\theta$, $\tau$, $x(0)$ do not have analytical solutions. We conduct random walk MH algorithm to sample new parameters. We use $\tau$ as an illustrative example. Conditional on samples at $n$–th iteration $\theta^{(n)}$, $x(0)^{(n)}$ and $\sigma^{2(n)}$,

1. $\tau^* \sim N(\tau^{(n)}, \sigma^2)$.

2. Solve DEs numerically and obtain $x(t_{ij}|\theta^{(n)}, \tau^*, x(0)^{(n)})$.

3. Compute the acceptance probability

$$p_{MH} = \min \left\{ 1, \frac{\gamma(\theta^{(n)}, \tau^*, x(0)^{(n)}, \sigma^{2(n)})}{\gamma(\theta^{(n)}, \tau^{(n)}, x(0)^{(n)}, \sigma^{2(n)})} \right\}.$$
4. Sample $u \sim U(0, 1)$, we accept $\tau^{(n+1)} = \tau^*$ if $u < p_{MH}$, otherwise $\tau^{(n+1)} = \tau^{(n)}$. 
Chapter 8

Inference for Misclassified Multinomial Data with Covariates

8.1 Introduction

The analysis of categorical data (Agresti and Kateri, 2011) has a longstanding and extensive literature that is applicable to many disciplines including the social, biomedical and marketing sciences. In some cases, categorical data are misclassified. For example, a subject whose “true” classification is the first category may be incorrectly classified in the second category.

Bross (1954) developed the standard framework for the analysis of binomial data subject to misclassification. In addition, he demonstrated that severely biased estimators can occur when the effect of misclassification is ignored. Since the seminal paper by Bross (1954), there has been a considerable amount of research directed towards misclassification in categorical data. Much of the effort has been from a Bayesian point of view, perhaps due to the fact that misclassification problems yield additional parameters where the number of parameters can sometimes exceed the dimensionality of the data. A literature review of the Bayesian analysis of misclassified multinomial data is given by Pérez et al. (2007).

The work presented here extends two previous papers in significant directions. Swartz et al. (2004) investigated the Bayesian analysis of multinomial data where special attention was given to the inherent problem of nonidentifiability. Here, we extend their model beyond a single multinomial cohort to the case of subject-specific covariates. In addition, the possibility of gold standard data are considered. Gerlach and Stamey (2007) developed Bayesian methods for variable selection in logistic regression models where misclassification is present. Here, we extend their model to the multinomial context where Dirichlet distributions are assigned to the primary parameters.

In Section 2, we present the multinomial model subject to misclassification in the presence of covariates. Whereas our focus is on data that are categorized subject to misclassification, we also provide modelling for gold standard data that are classified by an infallible classifier. A prior distribution is proposed where alternative parameterizations are provided according to default (i.e. reference) distributions or highly subjective priors. In Section 3, we explore pragmatic issues resulting
from the highly parametrized model. In particular, the problem of nonidentifiability is investigated, and a new definition of nonidentifiability is proposed that is particularly relevant to Bayesian settings. Computation is also discussed, and this is paramount in high-dimensional problems. In Section 4, we address the analysis of two datasets. The simulated dataset allows us to assess the reliability of inferences when the true parameters are known. We also are able to investigate the importance of the priors as we alter the amount of gold standard data. We then consider an actual dataset where the results appear to be sensible. Some concluding remarks are provided in Section 5.

8.2 Model development

8.2.1 The Data Model

Consider $n_i$ independent observations belonging to the $i$th covariate pattern, $i = 1, \ldots, r$, and assume that each observation is classified according to one of $q$ categories. Then the $j$th observation adhering to the $i$th covariate pattern takes the form $y_{ij} = (y_{ij1}, \ldots, y_{ijq})'$ where $y_{ijl} = 1$ denotes that the observation is classified according to category $l$ and $y_{ijk} = 0$ for $k \neq l$. Without misclassification, and assuming independence between observations, this is a standard product multinomial model and the likelihood is given by

$$L_0 = \prod_{i=1}^{r} \prod_{j=1}^{n_i} \prod_{k=1}^{q} p_{ik}^{y_{ijk}}$$

where $p_{ik} = \text{Prob}(y_{ijk} = 1)$ for $k = 1, \ldots, q$, $j = 1, \ldots, n_i$, $i = 1, \ldots, r$.

In the presence of misclassification, we denote $p_{il}$ as the probability that a subject belonging to the $i$th covariate pattern has the true (but unobserved) classification $l$. In other words, whereas $y_{ijk}$ is the observed classification, the true classification is a latent variable. Further, we let $\pi_{ilk}$ denote the probability that a subject with the $i$th covariate pattern is classified as $k$ given that its true classification is $l$. By the law of total probability, it follows that the probability of classification in category $k$ for a subject with the $i$th covariate pattern is $\sum_{l=1}^{q} p_{il} \pi_{ilk}$. This leads to the more complex likelihood

$$L_1 = \prod_{i=1}^{r} \prod_{j=1}^{n_i} \left( \sum_{l=1}^{q} p_{il} \pi_{ilk} \right)^{y_{ijk}} \sum_{j=1}^{n_i} \left( \sum_{l=1}^{q} p_{il} \pi_{ilk} \right)^{y_{ijk}}.$$

Therefore the parametrization of the standard product multinomial model has increased considerably with the addition of the $rq(q-1)$ misclassification parameters $\pi_{ilk}$.

We also allow for the case where some observations are doubly classified; first as above where the possibility of misclassification exists. Secondly, these observations are classified according to an infallible classifier where the classification is known to be correct. An infallible classifier is typically more expensive or less readily available than a fallible classifier. With double classification, the $j$th
observation adhering to the $i$th covariate pattern takes the form $(x_{ij}, z_{ij})$ where $x_{ij} = (x_{ij1}, \ldots, x_{ijq})'$ corresponds to the fallible classifier and $z_{ij} = (z_{ij1}, \ldots, z_{ijq})'$ corresponds to the infallible classifier. The pair $(x_{ij}, z_{ij})$ are known as gold standard data where each vector consists of $q - 1$ zeros and a single one. We let $m_i$ denote the number of observations in the $i$th covariate class that are doubly classified. This leads to the gold standard likelihood contribution

$$L_2 = \prod_{i=1}^{r} \prod_{j=1}^{m_i} \text{Prob}(x_{ij}, z_{ij})$$

$$= \prod_{i=1}^{r} m_i \prod_{j=1}^{q} \prod_{k=1}^{q} \left( \frac{z_{ijk}}{\sum_{l=1}^{q} z_{ijl}} \right) \prod_{k=1}^{q} \prod_{l=1}^{q} \frac{z_{ijl} x_{ijk}}{\pi_{lk}}$$

$$= \prod_{i=1}^{r} \prod_{j=1}^{q} \left( \sum_{l=1}^{q} z_{ijl} x_{ijk} \right) \prod_{k=1}^{q} \prod_{l=1}^{q} \pi_{lk}$$

$$\text{(2)}$$

We note that with some covariate patterns, there may be no data that are solely classified with a fallible classifier (i.e. $n_i = 0$), or there may be no doubly classified data (i.e. $m_i = 0$). Putting together the likelihoods associated with the data that are solely classified with a fallible classifier (1) and the gold standard data (2), the overall likelihood for the misclassified multinomial model in the presence of covariates is given by

$$L = L_1 \cdot L_2.$$  \hspace{1cm} (3)

Although a less realistic scenario, it is also possible to have data that are only classified by the infallible classifier. Such data provide information about the true classification probabilities $p_{ik}$. We define the data vector $w_{ij}$ with $s_i$ cases corresponding to the $i$th covariate pattern. Although we do not study this case, for reference, we provide its likelihood contribution

$$L_3 = \prod_{i=1}^{r} \prod_{j=1}^{s_i} \text{Prob}(w_{ij})$$

$$= \prod_{i=1}^{r} s_i \prod_{j=1}^{q} \left( \sum_{k=1}^{q} w_{ijk} \right) \prod_{k=1}^{q} \pi_{ik}$$

$$= \prod_{i=1}^{r} \prod_{j=1}^{q} \left( \sum_{k=1}^{q} w_{ijk} \right) \prod_{k=1}^{q} \pi_{ik}$$

8.2.2 The Prior Distribution

In specifying the prior distribution, we use the generic notation $[A \mid B]$ to denote the conditional probability density function of $A$ given $B$. In addition to the true classification probabilities $p$ and the misclassification probabilities $\pi$ developed above, we introduce additional parameters $a, b, \sigma_a$ and
\[ \begin{align*}
\sigma_p. \text{ Via conditional probability, the prior distribution is given by } \\
[p, \pi, a, b, \sigma_a, \sigma_b] &= [p, \pi \mid a, b, \sigma_a, \sigma_b] \ [a, b \mid \sigma_a, \sigma_b] \ [\sigma_a, \sigma_b] 
\end{align*} \]

where the parameter hierarchy is apparent; \( p \) and \( \pi \) are primary parameters, \( a \) and \( b \) are secondary parameters and \( \sigma_a \) and \( \sigma_b \) are tertiary parameters. The hierarchical labelling is relevant to the discussion in Section 3.1.

In Gerlach and Stamey (2007), it was convenient to model the logits of the true classification probabilities as linear combinations of covariate vectors \( u \) and impose normal priors. With constraints \( \sum_{k=1}^{q} p_{ik} = 1 \) in the multinomial setting, we instead model

\[ (p_{i1}, \ldots, p_{iq})' \sim \text{Dirichlet}(\exp(a_1'u_i), \ldots, \exp(a_q'u_i)) \quad (5) \]

where the true classifications are covariate dependent. Note that the distribution in (5) corresponds to the \( i \)th covariate pattern and independence is assumed for \( i = 1, \ldots, r \). Typically, it would be sensible to include a constant (intercept) term in the covariate vector \( u_i \). As in Gerlach and Stamey (2007), it is also reasonable to consider misclassification rates that are covariate dependent via

\[ (\pi_{i1}, \ldots, \pi_{iq})' \sim \text{Dirichlet}(\exp(b_1'u_i), \ldots, \exp(b_q'u_i)) \quad (6) \]

where \( i = 1, \ldots, r, l = 1, \ldots, q \) and (6) is conditionally independent of (5).

With covariate vectors of dimension \( v \), hyperpriors may be assigned as follows:

\[ (a_{k1}, \ldots, a_{kv})' \sim \text{Normal}(a_{k1}^{(0)}, \ldots, a_{kv}^{(0)}'), D_{\sigma_a}) \]

\[ (b_{lk1}, \ldots, b_{lkv})' \sim \text{Normal}(b_{lk1}^{(0)}, \ldots, b_{lkv}^{(0)}'), D_{\sigma_b}) \]

where \( l, k = 1, \ldots, q \), and \( D_{\sigma_a} \) and \( D_{\sigma_b} \) are diagonal matrices with diagonal vectors \( \sigma_a \) and \( \sigma_b \), respectively. The components of \( \sigma_a \) and \( \sigma_b \) have independent Inverse Gamma(\( r_a^{(0)}, s_a^{(0)} \)) and Inverse Gamma(\( r_b^{(0)}, s_b^{(0)} \)) distributions, respectively. Parameters with a superscript (0) are user-specified.

Hence the posterior distribution is proportional to the product of (3) and (4) with possible parameter constraints used to ameliorate nonidentifiability issues (Swartz et al., 2004).

### 8.3 Pragmatic issues

#### 8.3.1 Nonidentifiability

As modern statistical practice entertains increasingly complex models, the problem of nonidentifiability has become an increasingly important topic. Not only is it sometimes difficult to handle nonidentifiability issues, it is sometimes difficult to even detect nonidentifiability. Strangely, the topic of nonidentifiability does not appear to be receiving its due coverage in current statistics curricula. In a non-scientific survey of the third author’s bookshelf, he discovered 12 texts on mathematical
A statistician’s first encounter with nonidentifiability often occurs in a regression context. Consider the simple one-way ANOVA

\[ y_{ij} = \mu + \tau_i + \epsilon_{ij} \]

where \( y_{ij} \) is the \( j \)th response under treatment \( i \), \( \mu \) is the overall mean, \( \tau_i \) is the \( i \)th treatment effect, \( i = 1, 2 \), and the \( \epsilon_{ij} \) are random error terms. In matrix notation, \( y = (y_{11}, \ldots, y_{m1}, y_{21}, \ldots, y_{2n})' \) and the design matrix is given by \( X = (1, x_1, x_2) \) where 1 is a vector of ones, \( x_1 \) is a vector of \( n_1 \) ones followed by \( n_2 \) zeros, and \( x_2 \) is a vector of \( n_1 \) zeros followed by \( n_2 \) ones. The least squares estimator of \( (\mu, \tau_1, \tau_2)' \) given by \( (X'X)^{-1}X'y \) cannot be calculated since \( X'X \) is not invertible (i.e. \( X \) is not full rank). Clearly, this is a problem, but a simple solution is to introduce the constraint \( \tau_1 + \tau_2 = 0 \).

More formally, Basu and Ghosh (1983) defined nonidentifiability as follows:

**Definition.** Let \( U \) be an observable random variable with distribution function \( F_\theta \) and let \( F_\theta \) belong to a family \( \mathcal{F} = \{ F_\theta : \theta \in \Omega \} \) of distribution functions indexed by a parameter \( \theta \). Here \( \theta \) could be scalar or vector valued. We say that \( \theta \) is nonidentifiable by \( U \) if there is at least one pair \((\theta, \theta')\), \( \theta \neq \theta' \), where \( \theta \) and \( \theta' \) both belong to \( \Omega \) such that \( F_\theta(u) = F_{\theta'}(u) \) for all \( u \). In the contrary case we shall say \( \theta \) is identifiable.

At this point, we suggest that the Basu and Ghosh (1983) definition is overly rigid, and may not be appropriate for Bayesian statistical practice. For example, consider the following posterior density expressed in a common hierarchical structure

\[
[\theta_1, \theta_2 \mid y] \propto [y \mid \theta_1] \cdot [\theta_1 \mid \theta_2] \cdot [\theta_2].
\]

Although there is nothing wrong with (7), according to Basu and Ghosh (1983) there exists a nonidentifiability as both \( (\theta_1, \theta_2) \) and \( (\theta_1', \theta_2') \) yield the same sampling distribution given by \( [y \mid \theta_1] \). In fact, all hierarchical models, which are the mainstream of Bayesian modelling are nonidentifiable.

On the opposite side of the spectrum, there is an extreme point of view with respect to nonidentifiability and Bayesian statistics. Lindley (1972) stated, “In passing, it might be noted that unidentifiability causes no real difficulty in the Bayesian approach.” Reading into Lindley’s statement, he may have intended that after writing down the posterior which is proportional to the product of the likelihood and prior, there is no obstacle to integrating.

Despite Lindley’s claim, nonidentifiability is a problem for Bayesian statistical practice, and this has been explored by many authors including Gustafson et al. (2005); Gustafson (2010) and San Martín and González (2010). A common problem concerning nonidentifiability is that non-negligible posterior probability is dispersed over regions which are not apriori plausible. Essentially, problematic issues involving nonidentifiability in Bayesian statistics can be viewed as an artefact of non-sensical model building.
In Swartz et al. (2004), various constraints were introduced on the parameter space to avoid non-sensical model building in the misclassified multinomial model. A simple constraint which was suggested there, and will be utilized in this paper is

$$\pi_{ilk} < \pi_{ill}$$ (8)

for \(i = 1, \ldots, r\) and all \(l \neq k\). The constraint (8) essentially states that it is more probable for a subject to be classified correctly than incorrectly. Although (8) does not entirely take care of nonidentifiability in our model, it goes a long way in improving inferences as it eliminates unacceptable regions of the posterior space.

How do we then reconcile the overly rigid Basu and Ghosh (1983) definition with legitimate concerns for nonidentifiability in Bayesian practice? We suggest an alternative definition which takes the Bayesian hierarchical structure into account. For ease of notation, let \(y = \theta_0\), and express the posterior density as

$$[\theta_1, \ldots, \theta_k | \theta_0] \propto [\theta_0 | \theta_1] [\theta_1 | \theta_2] \cdots [\theta_{k-1} | \theta_k] = \prod_{i=1}^{k} [\theta_i | \theta_j]$$

where \(\theta_1, \ldots, \theta_k\) are potentially vector-valued and there is uniqueness in the hierarchical structure. We then say that a Bayesian model contains a hierarchical nonidentifiability if for some \(i = 1, \ldots, k\), there exists \(\theta_i \neq \theta'_i\) for which \([\theta_i | \theta_j] = [\theta_i | \theta'_j]\) for all values \(\theta_{i-1}\). The idea is that nonidentifiability occurs when at least one of the links in the hierarchical model do not identify between parameters. This is a weaker definition than the Basu and Ghosh (1983) definition, and the proposed definition prevents the immediate disqualification of hierarchical models as being nonidentifiable.

In the model considered in this paper, we refer to the expressions in Section 2.2, and obtain the posterior density

$$[p, \pi, a, b, \sigma_a, \sigma_b | y, x, z] \propto [y, x, z | p, \pi] [p, \pi | a, b, \sigma_a, \sigma_b] [a, b | \sigma_a, \sigma_b] [\sigma_a, \sigma_b]$$ (9)

where \([y, x, z | p, \pi] = L_1 L_2\) from (1) and (2). Viewing the terms in (9), the only possibility of hierarchical nonidentifiability occurs in the \([y, x, z | p, \pi]\) specification. In fact, if complete gold standard data exists (i.e. \(m_i > 1, i = 1, \ldots, r\)), then the model is hierarchically identifiable. In the absence of gold standard data (i.e. no \(L_2\) term), there are nonidentifiability issues as discussed in Swartz et al. (2004). The impact of gold standard data, the prior specification and constraints with respect to nonidentifiability and inference are explored in the simulated data example of Section 4.1.
8.3.2 Computation

Misclassified multinomial data in the presence of covariates leads to a complex model where the number of observations $\sum_{i=1}^{r} (n_i + m_i)$ is sometimes exceeded by the number of parameters. The model parameters are summarized in Table 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Number of Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{ik}$</td>
<td>primary parameters of interest which describe true classification probabilities</td>
<td>$r(q - 1)$</td>
</tr>
<tr>
<td>$\pi_{ilk}$</td>
<td>primary parameters of interest which describe misclassification probabilities</td>
<td>$rq(q - 1)$</td>
</tr>
<tr>
<td>$a_{kt}$</td>
<td>secondary parameters of interest relating covariates to true classification probabilities</td>
<td>$qv$</td>
</tr>
<tr>
<td>$b_{jkt}$</td>
<td>secondary parameters of interest relating covariates to misclassification probabilities</td>
<td>$q^2v$</td>
</tr>
<tr>
<td>$\sigma_a, \sigma_b$</td>
<td>tertiary parameters of interest</td>
<td>$2v$</td>
</tr>
</tbody>
</table>

Table 8.1: Parameter summary for the misclassified multinomial model with covariates.

With so many parameters and constraints, inferential procedures relying on exact integration are clearly infeasible. Approximation techniques such as Laplace’s method, importance sampling and quadrature (Evans and Swartz, 2000) are also not well-suited to this application. It seems that a Markov chain procedure may be the only avenue for obtaining inferential summaries. In a Markov chain approach, a Markov chain is constructed whose equilibrium distribution corresponds to the posterior. By averaging over samples, posterior estimates can be approximated.

A first approach in a Markov chain implementation is usually the consideration of a Gibbs sampling algorithm. This may be facilitated via data augmentation where the latent variable $T_{ij}$ is introduced in the context of data that are only classified by a fallible classifier. For $i = 1, \ldots, r$, $j = 1, \ldots, n_i$, we define

$$T_{ij} \equiv k \text{ if the } j\text{th observation sharing the } i\text{th covariate pattern has true classification } k$$

Accordingly, the posterior density simplifies and takes the form

$${[p, \pi, a, b, \sigma_a, \sigma_b, T \mid y, x, z]} \propto \left( \prod_{i=1}^{r} \prod_{j=1}^{n_i} p_{i,T_{ij}} \pi_{i,T_{ij}} \sum_{k=1}^{q} k y_{ijk} \right) \cdot L_2 \cdot [p, \pi, a, b, \sigma_a, \sigma_b]$$

where the likelihood $L_2$ and the prior $[p, \pi, a, b, \sigma_a, \sigma_b]$ are given in Section 2.2.

Although it is possible to derive the required full conditional distributions from the above expression, it is clearly a laborious process that is prone to coding errors. It is especially tedious when small changes to the model or changes in parameter constraints invoke considerable changes to the full conditional distributions.

For this problem, we implement a hybrid of Gibbs sampling and slice sampling (Neal, 2003) to generate variates from the joint posterior distribution $[p, \pi, a, b, \sigma_a, \sigma_b, T \mid y, x, z]$. We generically let
[A|·] denote the full conditional density of the variable A given the data and all other parameters. We sample [p, π, T|·] occurring in the L1 case according to following distributions

\[
[T_{ij} = m|·] = p_{lm} \cdot \pi_{im} \sum_{k = 1}^{\hat{q}} k y_{ijk} / \sum_{l = 1}^{q} p_{li} \cdot \pi_{il} \sum_{k = 1}^{\hat{q}} k y_{ijk}
\]

\[
[p_{i1}, \ldots, p_{iq}|·] \sim \text{Dirichlet} \left( \sum_{j=1}^{n_i} I(T_{ij} = 1) + \exp(a'_1 u_i), \ldots, \sum_{j=1}^{n_i} I(T_{ij} = q) + \exp(a'_q u_i) \right)
\]

\[
[\pi_{il1}, \ldots, \pi_{ilq}|·] \sim \text{truncated-Dirichlet}(c_{il1}, \ldots, c_{ilq}),
\]

\[
c_{ilm} = \exp(b'_{i1} u_i) + \sum_{j=1}^{n_i} I(T_{ij} = l) I(\sum_{k=1}^{q} k y_{ijk} = m)
\]

where \(i = 1, \ldots, r, l = 1, \ldots, q, m = 1, \ldots, q\). The truncated-Dirichlet distributions are truncated according to constraint (8). We use the rejection sampling algorithm and only accept samples that satisfy the constraint.

The full conditional densities \([p|·]\) and \([\pi|·]\) occurring in the L2 case are readily available and are given by

\[
[p_{i1}, \ldots, p_{iq}|·] \sim \text{Dirichlet} \left( \sum_{j=1}^{m_i} z_{ij1} + \exp(a'_1 u_i), \ldots, \sum_{j=1}^{m_i} z_{ijq} + \exp(a'_q u_i) \right)
\]

\[
[\pi_{il1}, \ldots, \pi_{ilq}|·] \sim \text{Dirichlet} \left( \sum_{j=1}^{m_i} z_{ij1} x_{ij1} + \exp(b'_{i1} u_i), \ldots, \sum_{j=1}^{m_i} z_{ijq} x_{ijq} + \exp(b'_{iq} u_i) \right)
\]

where \(i = 1, \ldots, r, l = 1, \ldots, q\).

The full conditional densities \([\sigma_a|·]\) and \([\sigma_b|·]\) are given by

\[
[\sigma_a|·] \sim \text{Inverse Gamma}(r_a^{(0)} + q, s_a^{(0)} + \sum_{q=1}^{v} a_k^2 / 2)
\]

\[
[\sigma_b|·] \sim \text{Inverse Gamma}(r_b^{(0)} + q^2, s_b^{(0)} + \sum_{q=1}^{v} \sum_{t=1}^{q} b_{kt}^2 / 2).
\]

The full conditional densities \([a_k|·]\) and \([b_{lk}|·]\) do not have familiar forms, and hence we implement slice sampling on

\[
[a_k|·] \propto \exp \left\{ - \frac{(a_k - a_k^{(0)})'(a_k - a_k^{(0)})}{2 \sigma_a^2} \right\} \prod_{j=1}^{r} \left[ \frac{\Gamma(\sum_{q=1}^{q} \exp(a'_q u_i))}{\Gamma(\exp(a'_q u_i))} \right] p_{ik}
\]

\[
[b_{lk}|·] \propto \exp \left\{ - \frac{(b_{lk} - b_{lk}^{(0)})'(b_{lk} - b_{lk}^{(0)})}{2 \sigma_b^2} \right\} \prod_{j=1}^{r} \left[ \frac{\Gamma(\sum_{q=1}^{q} \exp(b'_q u_i))}{\Gamma(\exp(b'_q u_i))} \right] \pi_{ilk}
\]

where \(l = 1, \ldots, q\) and \(k = 1, \ldots, q\).
8.4 Examples

8.4.1 Simulated Data

We considered \( r = 50 \) covariate patterns and \( q = 5 \) categorical responses which is typical of surveys carried out on a Likert scale. In the baseline dataset, we began with 20% gold standard data where there are 1000 observations. This was accomplished by setting \( n_i = 0 \) for covariate patterns \( i = 5, 10, 15, \ldots, 50 \) and \( n_i = 20 \) otherwise using the fallible classifier according to (1). For the gold standard data, we set \( m_i = 20 \) for covariate patterns \( i = 5, 10, 15, \ldots, 50 \) and \( m_i = 0 \) otherwise according to (2).

The covariates \( u_i \) in (5) and (6) have dimension \( v = 2 \) where \( u_i = (1, \bar{i})' \) contains a constant term and a term which increases linearly according to the covariate pattern \( i \). The parameter coefficients corresponding to the covariate in (5) were set according to \( a_1 = (0.1, 0.05)' \), \( a_2 = (0.2, 0.05)' \), \( a_3 = (0.3, 0.05)' \), \( a_4 = (0.4, 0.05)' \) and \( a_5 = (0.5, 0.05)' \). This implies that the probabilities of classification \( p_{ik} \) have prior probability distributions that are increasing across the five cells but the effect diminishes as \( i \) increases.

For the misclassification probabilities \( \pi_{ilk} \) in (6), we set the parameter coefficients in (6) according to

\[
\begin{align*}
b_{11} &= (2.0, 0.0)' \quad b_{12} = (1.0, 0.0)' \quad b_{13} = (0.0, 0.0)' \quad b_{14} = (0.0, 0.0)' \quad b_{15} = (0.0, 0.0)' \\
b_{21} &= (1.0, 0.0)' \quad b_{22} = (2.0, 0.0)' \quad b_{23} = (1.0, 0.0)' \quad b_{24} = (0.0, 0.0)' \quad b_{25} = (0.0, 0.0)' \\
b_{31} &= (0.0, 0.0)' \quad b_{32} = (1.0, 0.0)' \quad b_{33} = (2.0, 0.0)' \quad b_{34} = (1.0, 0.0)' \quad b_{35} = (0.0, 0.0)' \\
b_{41} &= (0.0, 0.0)' \quad b_{42} = (0.0, 0.0)' \quad b_{43} = (1.0, 0.0)' \quad b_{44} = (2.0, 0.0)' \quad b_{45} = (1.0, 0.0)' \\
b_{51} &= (0.0, 0.0)' \quad b_{52} = (0.0, 0.0)' \quad b_{53} = (0.0, 0.0)' \quad b_{54} = (1.0, 0.0)' \quad b_{55} = (2.0, 0.0)'.
\end{align*}
\]

This implies that misclassification occurs in neighbouring categories with the greatest prior probability and does not depend on the covariate pattern \( i \).

Using the above settings, we generated the underlying classification probabilities \( p_{ik} \) and the underlying misclassification probabilities \( \pi_{ilk} \) according to the Dirichlet distributions in (5) and (6). In turn, we used multinomial distributions to generate the data \( y_{ij} \) according to the fallible classifier and to generate the doubly classified gold standard data \( (x_{ij}, z_{ij}) \).

The model is completely specified except for the hyperparameters. For the hyperparameters, we begin with settings \( a_{kj}(0) = b_{kj}(0) = 0 \) for all \( k = 1, \ldots, q \), \( l = 1, \ldots, q \) and \( j = 1, \ldots, v \). We also introduce diffuse hyperpriors \( \sigma_a \sim \text{Inverse Gamma}(1, 0.01) \) and \( \sigma_b \sim \text{Inverse Gamma}(1, 0.01) \).

There is no shortage of ways that one might investigate the simulation results. We choose to concentrate on the true classification probabilities \( p_{ik} \) since these are the parameters of primary interest. Letting \( \hat{p}_{ik} \) denote the underlying values that were generated and letting \( \tilde{p}_{ik} \) denote the estimated posterior means from the Markov chain, we define the statistic

\[
D = \frac{1}{rq} \sum_{i=1}^{r} \sum_{k=1}^{q} (p_{ik} - \tilde{p}_{ik})^2 \tag{10}
\]
as a measure of the ability of the model to estimate the classification probabilities. We interpret $D$ in (10) as the average distance between an estimated $p_{ik}$ and its underlying value. Smaller values of $D$ indicate successful estimation. Standard diagnostics were assessed to confirm convergence of the Markov chains.

In the baseline model, although there was evidence of autocorrelation in some of the parameters $b_{lk2}$, we observed stability in the parameter estimates. Here, we observed $D = 0.0052$ based on 500,000 iterations of the Markov chain. We then increased the size of the dataset from 1000 observations to 5000 observations and retained the 20% gold standard data ratio. In this case, we observed $D = 0.0043$. This suggests (as expected) that more data improves estimation.

To investigate the effect of the constraint (8) which is intended to reduce nonidentifiability issues, we repeated the simulation procedure with the original 1000 observations under the baseline model without the constraint. In this case, we observed a larger $D = 0.0062$ which suggests that the constraint is helpful.

It is also the case that a higher proportion of gold standard data improves estimation. This was observed by setting $n_i = 0$ for covariate patterns $i = 2, 4, 6, \ldots, 50$ and $n_i = 20$ for covariate patterns $i = 1, 3, 5, \ldots, 49$ using the fallible classifier according to (1). For the gold standard data, we set $m_i = 20$ for covariate patterns $i = 2, 4, 6, \ldots, 50$ and $m_i = 0$ otherwise according to (2). In this case, we had 50% gold standard data and 1000 observations as before. In this case, there was a reduction from $D = 0.0052$ in the baseline model to $D = 0.0047$.

Finally, we fiddled with the prior specification. We set the hyperparameters according to $a_k^{(0)} = (0.3, 0.05)'$, $b_{kk}^{(0)} = (2.0, 0.0)'$ and $b_{lk}^{(0)} = (0.0, 0.0)' (l \neq k)$ for all $k = 1, \ldots, q$ and $l = 1, \ldots, q$. Previously, the hyperpriors on $\sigma_a$ and $\sigma_b$ were diffuse. With a more informative prior Inverse Gamma(10000, 1.0), less probability is assigned to implausible regions in the contours resulting from nonidentifiability. This resulted in quicker mixing and improved estimation $D = 0.0048$ when compared to the baseline results.

### 8.4.2 Actual Data

The following example is relatively simple and does not make full use of the extensive modelling and methods developed in this paper. However, the application area is novel, and the results are both readily interpretable and interesting.

Here we consider data taken from Replay Challenges during the last five regular seasons (2014-2018) of the National Football League (NFL). During this period, there have been no rule changes with respect to Replay Challenges. In an attempt to improve decision making by game officials, each of the two coaches in an NFL game is allowed to appeal the decisions of up to two plays using video replay technology. In the case where a coach has made two successful challenges, he is allowed to make a third challenge.

The data were obtained using the R package nflscrapr (Horowitz et al., 2018), and we have restricted the dataset to the 432 coaches’ challenges that involved pass completion rulings. In this dataset, we assume the result of the video replay is correct and this provides us with gold standard
data. Therefore, following (2), \((x_{ij1}, x_{ij2})\) and \((z_{ij1}, z_{ij2})\) correspond to the \(j\)th challenge under the \(i\)th covariate pattern where \((x_{ij1}, x_{ij2}) = (1, 0)\) if the play was ruled a completion on the field, \((x_{ij1}, x_{ij2}) = (0, 1)\) if the play was ruled an incompletion on the field, \((z_{ij1}, z_{ij2}) = (1, 0)\) if the play was ruled a completion by video replay and \((z_{ij1}, z_{ij2}) = (0, 1)\) if the play was ruled an incompletion by video replay. The primary parameters of interest are the probabilities \(p_{i1}\) that plays on the field were determined as completions. However, we are mostly interested in the misclassification probabilities \(\pi_{i12}\) that the video replay confirmed a completion and the misclassification probabilities \(\pi_{i21}\) that the video replay reversed an incompletion to a completion.

For simplicity, we consider \(r = 2\) covariate patterns:

- \(i = 1 \equiv \) challenge by the home team
- \(i = 2 \equiv \) challenge by the road team

We also set default covariates \(u_1 = u_2 = 1.0\), default hyperparameters \(a^{(0)}_{11} = a^{(0)}_{21} = 0.0, b^{(0)}_{111} = b^{(0)}_{121} = a^{(0)}_{211} = b^{(0)}_{221} = 0.0, d^{(0)} = d^{(0)} = 1.0, s^{(0)} = s^{(0)} = 0.01\). For this application, we did not impose constraints as in (8) since the gold standard data alleviated nonidentifiability.

For this simple problem, the Markov chain converged rapidly. We obtained the following posterior estimates of the true classification and the misclassification probabilities: \(p_{11} = 0.53, p_{21} = 0.48, \pi_{112} = 0.37, \pi_{121} = 0.55, \pi_{212} = 0.36\) and \(\pi_{221} = 0.51\). Our first observation is that home team challenges appear to have slightly higher probabilities of reversal than road team challenges (i.e. \(\pi_{112} > \pi_{212}\) and \(\pi_{121} > \pi_{221}\)); this may be evidence of the phenomena known as the “home team advantage”.

Furthermore, we note that the probability that a completion is upheld exceeds the probability that it is reversed and the probability that an incompletion is upheld exceeds the probability that it is reversed. This may be a surprise to some football fans who doubt that officials make good decisions. However, what makes this result even more surprising is that coaches typically invoke challenges only when they believe officials have made mistakes.

### 8.5 Concluding remarks

This paper has expanded earlier models on multinomial misclassification by considering the inclusion of subject specific cohorts in the presence of both fallible and infallible classifiers. In addition, an alternative definition of nonidentifiability has been proposed which appears well-suited to hierarchical models.

From our investigations on sample datasets, we have observed that (1) nonidentifiability resulting from misclassification complicates inference in this class of problems, (2) that sufficient gold standard data helps alleviate the nonidentifiability issue and that (3) a sufficiently informative prior helps alleviate the nonidentifiability issue.

In this project, MCMC algorithms have been implemented in R to sample from the corresponding distributions. The code is available from the authors upon request.
Chapter 9

Adaptive Importance Sampling from Restricted Skew-Student Distributions

9.1 Introduction

The evaluation of integrals is a fundamental problem that presents itself in many diverse fields such as mathematical finance, economics and physics. For statisticians, integrals are commonplace in the Bayesian framework and arise as posterior expectations. In many applications, particularly in high dimensions, the integrals in question are intractable. Therefore, one must resort to methods of integral approximation. Evans and Swartz (2000) describe the major approaches used in the approximation of integrals with a particular emphasis on integrals arising in statistics.

One of the long-standing approaches to integral approximation is importance sampling which dates back to at least Metropolis and Ulam (1949). Importance sampling proceeds by rewriting the integral of interest

\[ I(f) = \int f(y) \, dy \]  

as

\[ I(f) = \int \left( \frac{f(y)}{w(y)} \right) w(y) \, dy \]

where the density function \( w(y) \) is introduced and is referred to as an importance sampler. In importance sampling, independent variates \( y^{(1)}, \ldots, y^{(N)} \) are generated from the distribution corresponding to \( w(y) \), and we obtain the importance sampling estimator

\[ \hat{I}(f) = \frac{1}{N} \sum_{i=1}^{N} \frac{f(y^{(i)})}{w(y^{(i)})}. \]
We note that the estimator \( \hat{I}(f) \) is unbiased and is a consistent estimator where

\[
Var(\hat{I}(f)) = \frac{1}{N} \left[ \int \frac{f^2(y)}{w(y)} \, dy - \hat{I}^2(f) \right]
\]

is finite if the integral in (3) is finite. From (2), we see that the variance of \( \hat{I}(f) \) is small when \( f(y) \approx kw(y) \) for some constant \( k \). Therefore our goal is to choose an importance sampler \( w(y) \) which permits convenient variate generation and whose shape mimics the shape of \( f(y) \).

As a general purpose integration technique, it appears that importance sampling has fallen out of favour compared to some of the popular Markov chain methods such as the Metropolis-Hastings algorithm (see Gilks et al. (1995)). However, in principle there is nothing wrong with importance sampling. In fact, importance sampling has several advantages over Markov chain methods. For example, error assessment of averages in a Markov chain is not straightforward due to the dependence structure in the chain. Also, Markov chain methods require the determination of “convergence to stationarity” of the chain. Furthermore, even if a practitioner wishes to use Markov chain methods, it is comforting to have an alternative technique which provides corroborating evidence of the accuracy of the approximations. We believe that the “problem” with importance sampling is that there is a limited choice of practical importance samplers for integrals defined on \( \mathbb{R}^n \). In most instances, practitioners use the multivariate normal or the longer tailed multivariate Student families (Evans and Swartz, 2000). A drawback with these families is that they are symmetric and may not be effective when the integrand \( f(y) \) in (1) is skewed.

In this paper, we propose the use of restricted skew-Student distributions for importance sampling on \( \mathbb{R}^n \). A comprehensive treatment of the properties and applications of skew-elliptical distributions (with particular emphasis on skew-normal distributions) appears in the volume edited by Genton (2004). The skew-Student family extends the range of integrals for which importance sampling is successful. In Section 2, we describe the restricted skew-normal family of distributions and provide the relevant details for the implementation of adaptive importance sampling. For example, we describe variate generation and propose adaptive methods for fitting a member of the restricted skew-normal family to a particular integral. Adaptive importance sampling is then extended to the family of restricted skew-Student distributions in Section 3. In Section 4, we demonstrate the utility of the approach with some examples. We conclude with a short discussion in Section 5.

### 9.2 Restricted skew-normal distributions

There are a number of variations in the definition of skew-elliptical distributions in the literature (Genton, 2004). We consider the standard multivariate skew-normal distribution as defined by Azzalini and Valle (1996).
We observe that this stretching (skewing) provides different shapes than the elliptical contours of multivariate normal distributions. Whereas the multivariate normal family is characterized by $n(n + 3)/2$ parameters, the $RSN_n$ family consists of $3n$ parameters (i.e. $\epsilon$, $s$ and $\alpha$) and is more parsimonious when $n \geq 4$.

By the change of variables formula, the probability density function for $y \sim RSN(\epsilon, s, \alpha)$ is given by

$$
w(y) = \frac{2}{(2\pi)^{n/2} |S|} \exp\left\{ -\frac{1}{2} (y - \epsilon)' S^{-2} (y - \epsilon) \right\} \Phi\left( \alpha' S^{-1} (y - \epsilon) \right)
= \frac{2}{(2\pi)^{n/2} \prod_{i=1}^{n} s_i} \exp\left\{ -\frac{1}{2} \sum_{i=1}^{n} \left( \frac{y_i - \epsilon_i}{s_i} \right)^2 \right\} \Phi\left( \sum_{i=1}^{n} \frac{\alpha_i (y_i - \epsilon_i)}{s_i} \right).
$$

(4)
An important consideration for the implementation of importance sampling is that \( w(y) \) in (4) is easily evaluated at any point \( y \). This is not the case for all of the various skew-elliptical distributions which have been proposed in the literature.

The next relevant issue for importance sampling is the generation of random variates \( y \) having the pdf (4) given the parameters \( \epsilon, s \) and \( \alpha \). To generate, we use the characterization in Proposition 1.6.2 of Genton (2004) and follow the steps:

1. set \( \delta = (1 + \alpha' \alpha)^{-1/2} \alpha \)

2. obtain the Cholesky factor \( A \) of \( \Sigma = \begin{pmatrix} 1 & \delta' \\ \delta & I_n \end{pmatrix} \) (i.e. obtain the unique lower triangular matrix \( A \) with positive diagonal entries such that \( AA' = \Sigma \))

3. generate a sample \( t_1, \ldots, t_{n+1} \) of Normal(0, 1) variates

4. set \( \begin{pmatrix} u \\ v \end{pmatrix} = A \begin{pmatrix} t_1 \\ \vdots \\ t_{n+1} \end{pmatrix} \) where \( u \) is a scalar

5. set \( y = \begin{cases} \epsilon + Sv & \text{if } u > 0 \\ \epsilon - Sv & \text{if } u < 0 \end{cases} \)

The final and most challenging issue relevant to the implementation of restricted skew-normal importance sampling is the determination of a member of the restricted skew-normal family corresponding to a particular integral. Again, we seek a restricted skew-normal density \( w(y) \) which mimics the integrand \( f(y) \) in (1). We consider an adaptive procedure where we fit the parameters \( \epsilon, s \) and \( \alpha \) in stages. We begin by assuming that \( f(y) \) in (1) is non-negative as in Bayesian applications where \( f(y) \) is proportional to the posterior density. We set \( \alpha^{(0)} = 0 \) which reduces the restricted skew-normal importance sampler to a multivariate normal importance sampler with mean \( \epsilon \) and covariance matrix
A standard approach (Evans and Swartz, 2000) is to set $\epsilon = \epsilon^{(0)}$ and $S = S^{(0)}$ using a numerical approach that is based on the normal approximation of $f(y)$. We let

- $\epsilon^{(0)}$ be the solution of $\frac{\partial \log f(y)}{\partial y} = 0$ (i.e. $\epsilon^{(0)}$ maximizes $f(y)$)

- $S^{(0)}$ be the diagonal matrix whose diagonal entries are the same as the diagonal entries of the Cholesky factor of the matrix $\left( -\frac{\partial^2 \log f(y)}{\partial y_i \partial y_j} \right)^{-1}$

At this point, we have defined a skew-normal importance sampler $w^{(0)}(y)$ based on $\epsilon^{(0)}$, $S^{(0)}$ and $\alpha^{(0)}$. However, the shape of $w^{(0)}(y)$ may not mimic the shape of the integrand $f(y)$ very well. To improve the fit, we consider an adaptive sampling approach where we first sample $y^{(1)}, \ldots, y^{(N)}$ from $w^{(0)}(y)$ and calculate the ratio

$$\hat{R}(m) = \frac{\hat{I}(m f)}{\hat{I}(f)} = \frac{\frac{1}{N} \sum_{i=1}^{N} m(y^{(i)}) f(y^{(i)}) / w^{(0)}(y^{(i)})}{\frac{1}{N} \sum_{i=1}^{N} f(y^{(i)}) / w^{(0)}(y^{(i)})} \quad (5)$$

for various functions $m(y)$. The $3n$ functions $m(y)$ are chosen so that $\hat{R}(m)$ corresponds to estimates of the mean, the second central moments and the third central moments of the density proportional to $f(y)$. In other words, we obtain the $n$-dimensional vector $\hat{\mu} = (\hat{\mu}_1, \ldots, \hat{\mu}_n)'$ where $\hat{\mu}_j$ is calculated according to (5) by setting $m(y) = y_j$. We also obtain the $n$-dimensional vector $\hat{\gamma} = (\hat{\gamma}_1, \ldots, \hat{\gamma}_n)'$ where $\hat{\gamma}_j$ is calculated by setting $m(y) = (y_j - \hat{\mu}_j)^2$. We also obtain the $n$-dimensional vector $\hat{\tau} = (\hat{\tau}_1, \ldots, \hat{\tau}_n)'$ where $\hat{\tau}_j$ is calculated by setting $m(y) = (y_j - \hat{\mu}_j)^3$.

In the first stage of adaptation, we update $\epsilon$, $s$ and $\alpha$ by referring to the expected values for skew-normal distributions as derived in Azzalini and Valle (1996) and in Genton et al. (2001). We match sample moments with theoretical moments

$$\hat{\mu}_i = \epsilon_i + s_i \left( \frac{2}{\pi} \right)^{1/2} \delta_i \quad (6)$$

$$\hat{\gamma}_i = s_i^2 \left( 1 - \frac{2}{\pi} \delta_i^2 \right) \quad (7)$$

and

$$\hat{\tau}_i = \frac{\sqrt{2}(4 - \pi)}{\pi^{3/2}} s_i^3 \delta_i^3 \quad (8)$$

where $\delta_i = (1 + \alpha' \alpha)^{-1/2} \alpha_i$ and $i = 1, \ldots, n$. Using (6), we substitute $\delta_i = (\pi/2)^{1/2}(\hat{\mu}_i - \epsilon_i)/s_i$ into (8) which yields

$$\epsilon_i^{(1)} = \hat{\mu}_i - \left( \frac{2\hat{\tau}_i}{4 - \pi} \right)^{1/3} \quad (9)$$
for \( i = 1, \ldots, n \). Therefore, (9) is used to provide the updated parameter \( \epsilon^{(1)} \) for adaptive importance sampling.

Similarly, we use (6) and substitute \( \delta_i = (\pi / 2)^{1/2}(\hat{\mu}_i - \epsilon_i^{(1)}/s_i \) into (7) where the updated parameter \( \epsilon_i^{(1)} \) is used. This yields

\[
\gamma^{(1)}_i = \left( \hat{\gamma}_i + (\hat{\mu}_i - \epsilon_i^{(1)})/s_i \right)^{1/2} \tag{10}
\]

for \( i = 1, \ldots, n \) which provides the updated parameter \( s^{(1)} \) for adaptive importance sampling.

Lastly, we update the skew parameter \( \alpha \) using the estimated moments \( \hat{\mu}, \hat{\gamma} \) and \( \hat{\tau} \), and the previously updated parameters \( \epsilon^{(1)} \) and \( s^{(1)} \). First, using \( \delta_i = (1 + \alpha' \alpha)^{-1/2} \delta_i \), it is easy to establish that \( \alpha_i = (1 - \delta' \delta)^{-1/2} \delta_i \). Then, we again use (6) and obtain \( \delta_i^{(1)} = (\pi / 2)^{1/2}(\hat{\mu}_i - \epsilon_i^{(1)})/s_i^{(1)} \) which is substituted into the expression for \( \alpha_i \) which yields

\[
\alpha_i^{(1)} = (1 - \delta^{(1)' \delta^{(1)})}^{-1/2} \delta_i^{(1)} \tag{11}
\]

for \( i = 1, \ldots, n \). We note that the method of moments estimation does not respect constraints such as \( \delta^{(1)' \delta^{(1)}} < 1 \). Later, in the examples, we do not update the importance sampling parameters (and continue sampling) if the constraint \( \delta^{(1)' \delta^{(1)}} < 1 \) is not satisfied.

Therefore, (9), (10) and (11) provide the steps for obtaining updated parameters \( \epsilon^{(1)}, s^{(1)} \) and \( \alpha^{(1)} \). These parameters define an updated importance sampler \( w^{(1)}(y) \) which hopefully better mimics the integrand \( f(y) \).

In the second stage of adaptation, we sample \( y^{(1)}, \ldots, y^{(N)} \) according to the importance sampler \( w^{(1)}(y) \) and repeat the above fitting process. This leads to a new importance sampler \( w^{(2)}(y) \).

Obviously, adaptation can continue by repeated sampling and fitting. However, in the applications which we have considered, only a few (e.g. less than six) rounds of adaptation are required since subsequent iterations typically result in marginal changes to the current importance sampler. Also, there may be various strategies in combining the moment estimates from each round of sampling. For example, one might take weighted averages where the weights correspond to the inverse of the standard errors of the estimates. Alternatively, one might simply ignore all of the estimates obtained from the early rounds of adaptation, and instead, approximate integrals based on only the results from a long run using the final importance sampler.

As a by-product of estimating integrals, we note that the proposed algorithm attempts to find a good importance sampler, i.e. an importance sampler that mimics the posterior density in Bayesian calculations. In the Metropolis-Hastings algorithm, a popular strategy is to seek a candidate generating density that mimics the posterior. The resultant MCMC (Markov chain Monte Carlo) algorithm is often referred to as independence sampling. Therefore, our algorithm may also be seen as a pre-conditioner to MCMC. By determining an importance sampler, we obtain a candidate generating density for Metropolis-Hastings independence sampling.
9.3 Restricted skew-student distributions

Although the multivariate normal distribution has been used extensively in importance sampling applications, the multivariate Student can be implemented with no real additional difficulties (see Evans and Swartz (2000)). An advantage of the Student over the normal is longer “tails”. In some applications, the shorter tails of the normal may lead to importance sampling estimators with infinite variance.

We therefore consider an extension of the proposed adaptive importance sampling algorithm where restricted skew-normal distributions are replaced by restricted skew-Student distributions. With some change of notation, we follow the development of skew-Student distributions as given by Azzalini and Capitanio (2003).

Letting \( z \sim SSN_n(\Omega, \alpha) \), we define

\[
y = \epsilon + S z \sqrt{v/W}
\]

(12)

where \( W \sim \chi^2_v \) is distributed independently of \( z \), and \( v > 3 \) so that the third moment \( E(y^3) \) is finite. This is the analogue of the traditional scale mixing with respect to the chi-squared when transforming variables from multivariate normal to multivariate Student. The random variable \( y \) in (12) has a skew-Student \( n(v, \epsilon, S, \Omega, \alpha) \) where \( v \) is referred to as the degrees of freedom. We note that as \( v \to \infty \), the distribution of \( y \) converges to the \( SN_n(\epsilon, S, \Omega, \alpha) \) distribution, and therefore the skew-Student family can be seen as a generalization of the skew-normal family. Although the notation differs, the skew-Student \( n(v, \epsilon, S, \Omega, \alpha) \) distributions coincide with the distributions in Branco and Dey (2001).

In the restricted skew-Student setting, we require \( \Omega = I \). For adaptive restricted skew-Student importance sampling, restricted skew-Student variates \( y \) are generated following the construction in (12). Specifically, we generate \( (\epsilon + S z) \sim RSN_n(\epsilon, s, \alpha) \) as previously discussed in Section 2 and then generate \( W \sim \chi^2_v \). Then using the second expression in (12) with appropriate algebra yields the restricted skew-Student variate \( y \).

Restricted skew-Student importance sampling also requires the evaluation of the density of the importance sampler and this is given by

\[
w(y) = \frac{2\Gamma\left(\frac{n+v}{2}\right)(v/2)^{v/2}}{(2\pi)^{v/2}\Gamma\left(\frac{v}{2}\right)\prod_{i=1}^{n} s_i} \left(\frac{q(y)}{2}\right)^{-(n+v)/2} \text{Prob} \left(T \leq \sqrt{(n+v)/q(y)} \sum_{i=1}^{n} \alpha_i (y_i - \epsilon_i) / s_i \right)
\]

where \( q(y) = v + \sum_{i=1}^{n} (y_i - \epsilon_i)^2 / s_i^2 \) and \( T \sim \text{Student}_{n+v} \). Therefore we also require the evaluation of the distribution function of the univariate Student distribution and this is available in many statistical software packages such as the R programming language.

The final step relevant to the implementation of adaptive restricted skew-Student importance sampling is the determination of a member of the restricted skew-Student family in each round of adaptation. Consider then the first round of adaptation where the same steps are taken in subsequent
rounds. We follow the approach previously developed for adaptive restricted skew-normal importance sampling by matching moments. As in expressions (6), (7) and (8), we use coordinate-wise formulae for the first central moment

$$\hat{\mu}_i = \epsilon_i + s_i \delta_i \left( \frac{v}{\pi} \right)^{1/2} \frac{\Gamma(\frac{1}{2} v - \frac{1}{2})}{\Gamma(\frac{1}{2} v)}$$

(13)

the second central moment

$$\hat{\gamma}_i = s_i^2 \left[ \frac{v}{v - 2} - \left( \delta_i \left( \frac{v}{\pi} \right)^{1/2} \frac{\Gamma(\frac{1}{2} v - \frac{1}{2})}{\Gamma(\frac{1}{2} v)} \right)^2 \right]$$

(14)

and the third central moment

$$\hat{\tau}_i = \frac{3v s_i^2}{(v - 3)(v - 2)} (\hat{\mu}_i - \epsilon_i) - \frac{\pi \Gamma^2(\frac{v}{2})}{(v - 3) \Gamma^2(\frac{1}{2} v - \frac{1}{2})} (\hat{\mu}_i - \epsilon_i)^3 + 2(\hat{\mu}_i - \epsilon_i)^3.$$ (15)

We note that a difference between (6)-(8) and (13)-(15) is that the latter equations also depend on $v$. As before, we solve these equations to give updated skew-Student importance sampling parameters. Parameter $\epsilon_i$ does not have a closed form solution. Therefore, we numerically solve

$$\hat{\tau}_i = \frac{3}{v - 3} [\hat{\gamma}_i + (\hat{\mu}_i - \hat{\epsilon}_i)^2] (\hat{\mu}_i - \epsilon_i) - \frac{\pi \Gamma^2(\frac{v}{2})}{(v - 3) \Gamma^2(\frac{1}{2} v - \frac{1}{2})} (\hat{\mu}_i - \epsilon_i)^3 + 2(\hat{\mu}_i - \epsilon_i)^3.$$ (16)

to obtain the method of moments estimates. The method of moments estimates $s_i$ and $\delta_i$ are given by

$$s_i = \left[ \frac{v - 2}{v} (\hat{\gamma}_i + (\hat{\mu}_i - \hat{\epsilon}_i)^2) \right]^{1/2}$$

(17)

and

$$\delta_i = \frac{\hat{\mu}_i - \epsilon_i}{s_i (\hat{\gamma}_i)^{1/2} \frac{\Gamma(\frac{1}{2} v - \frac{1}{2})}{\Gamma(\frac{1}{2} v)}}.$$ (18)

Therefore equations (16), (17) and (18) provide the parameter updates for successive iterations of adaptive skew-Student importance sampling. Note that in adaptive skew-Student importance sampling, we do not currently have a good method for fitting the degrees of freedom parameter $v > 3$. Instead, we take the approach of setting $v$ in advance where smaller degrees of freedom lead to longer tailed distributions. Our view is that longer tails (e.g. $v = 5$) are better in terms of insuring estimators with finite variances. Alternatively, one can run the algorithm several times with different choices of $v$, and select the value which gives an estimator with the smallest variance.
9.4 Examples

9.4.1 Example 1

We first consider a 12-dimensional integral where the integrand \( f(x) \) is the density of the \( RS_{12}(\epsilon, s, \alpha) \) distribution where

\[
\epsilon = (1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12)' \\
s = (1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \ 3 \ 3 \ 3 \ 3)' \\
\alpha = (1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2\ 2)' 
\]

This is a good test case as the dimensionality is sufficiently high to be problematic for most asymptotic approximations and quadrature methods (Evans and Swartz 2000), and ideally, we would like to see the restricted skew-normal importance sampler approach the distribution given by the integrand.

In this problem, since \( f(x) \) is a density, the inverse norming constant \( I(f) = 1.0 \) and we use the estimates \( \hat{I}(f) \) to assess the performance of the adaptive importance sampling algorithm. We chose \( N = 30,000,000 \) iterations in each adaptive step. Each round of sampling and adaptation required approximately five minutes of computation on a 1.3 GHz Intel Core i5 processor.

In Table 1, we give some summary results of the algorithm based on four stages of adaptation. In particular we provide \( \hat{I}(f) \) and its standard error. We also provide the parameters \( s^{(1)}, \ldots, s^{(4)} \) updated after each round of sampling. We observe that common approach of multivariate normal importance sampling (i.e. stage (1)) is unreliable as the estimate \( \hat{I}(f) \) is nearly eight standard errors from its true value. This is a typical problem with importance sampling (Evans and Swartz 2000) when the importance sampler does not adequately mimic the integrand. However, we do see that the adaptive algorithm quickly delivers a good fitting importance sampler, where after four iterations, the \( s \) vector in the importance sampler matches the \( s \) vector of the integrand. We note that after four rounds of adaptation, the standard error of \( \hat{I}(f) \) reduced to 0.000002. Therefore, ignoring the fact that the standard error in the first round underestimates the true standard error, the adaptive algorithm provides an increase in efficiency by a factor of \( (0.008619/0.000002)^2 \approx 1.9 \times 10^7 \). In other words, at least \( 1.9 \times 10^7 \) times as much multivariate normal importance sampling is required to obtain the equivalent precision as restricted skew-normal importance sampling based on \( (\epsilon^{(4)}, s^{(4)}, \alpha^{(4)}) \).

Finally, we remark that in a similar fashion to \( s \), the importance sampling parameters \( \epsilon \) also converged quickly to the parameter values corresponding to the integrand. However, the \( \alpha \) vector does not converge quickly. After six rounds of adaptation,

\[
\alpha^{(6)} = (0.97 \ 0.90 \ 0.96 \ 0.80 \ 0.86 \ 0.92 \ 1.93 \ 1.89 \ 1.90 \ 1.90 \ 1.88 \ 1.91)' .
\]

The lack of speedy convergence of the skew parameter \( \alpha \) is consistent with the literature (Azzalini and Capitanio, 1999) where maximum likelihood estimation is often problematic due to flat likelihood surfaces in regions surrounding the mle of \( \alpha \). Fortunately, for adaptive importance sampling, this is not such an important issue. Recall we are trying to find an importance sampler that mimics the
indefinitely, and any skew-normal that fits reasonably well accomplishes our goal whether or not it is optimal.

<table>
<thead>
<tr>
<th>Stage of Adaptation</th>
<th>Restricted Skew-normal</th>
<th>Restricted Skew-Student_{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
</tr>
<tr>
<td>( \hat{I}(f) )</td>
<td>0.93021 0.99954 1.00000 1.00002</td>
<td>1.00182 1.00000 0.99999 0.99993</td>
</tr>
<tr>
<td>SE(( \hat{I}(f) ))</td>
<td>0.008619 0.000032 0.000003 0.000002</td>
<td>0.003404 0.000159 0.000159 0.000159</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>0.96 1.03 1.00 1.00</td>
<td>1.04 0.88 0.89 0.89</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>0.99 1.04 1.00 1.00</td>
<td>0.46 0.89 0.89 0.89</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>1.30 1.04 1.00 1.00</td>
<td>1.05 0.89 0.89 0.76</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>0.95 1.03 1.00 1.00</td>
<td>0.66 0.88 0.89 0.89</td>
</tr>
<tr>
<td>( s_5 )</td>
<td>0.71 1.93 2.00 2.00</td>
<td>2.00 1.77 1.77 1.77</td>
</tr>
<tr>
<td>( s_6 )</td>
<td>2.30 2.03 2.00 1.99</td>
<td>2.71 1.77 1.77 1.77</td>
</tr>
<tr>
<td>( s_7 )</td>
<td>1.55 2.07 2.00 2.00</td>
<td>1.70 1.71 1.71 1.71</td>
</tr>
<tr>
<td>( s_8 )</td>
<td>2.47 2.13 2.00 2.00</td>
<td>1.94 1.71 1.71 1.71</td>
</tr>
<tr>
<td>( s_9 )</td>
<td>3.68 3.14 3.00 3.00</td>
<td>3.28 2.57 2.57 2.57</td>
</tr>
<tr>
<td>( s_{10} )</td>
<td>2.01 3.10 3.00 3.01</td>
<td>3.18 2.57 2.57 2.57</td>
</tr>
<tr>
<td>( s_{11} )</td>
<td>3.15 3.05 2.99 3.01</td>
<td>2.96 2.57 2.57 2.57</td>
</tr>
<tr>
<td>( s_{12} )</td>
<td>2.97 2.95 3.01 3.00</td>
<td>2.24 2.57 2.57 2.57</td>
</tr>
</tbody>
</table>

Table 9.1: Some summary results corresponding to Example 1.

In Table 1, we also provide some summary results corresponding to four rounds of adaptive restricted skew-Student_{10} importance sampling. Since the integrand is restricted skew-normal, we do not expect the algorithm to perform quite as well as adaptive restricted skew-normal importance sampling, and this is the case. However, the standard error of \( \hat{I}(f) \) in restricted skew-Student_{10} importance sampling quickly (i.e. two rounds) reduced to 0.000159 and this represents an increase in efficiency by a factor of \((0.003404/0.000159)^2 \approx 458\) over standard multivariate normal importance sampling. Additional simulations (not reported) indicate that the algorithm performs less well as the degrees of freedom of the importance sampler decrease and we move away from normality.

### 9.4.2 Example 2

Our second example is taken from the second test case in Evans et al. (1995). This is a 9-dimensional integral based on the Bayesian analysis of a contingency table with parameter vector \( \theta = (\theta_1, \ldots, \theta_9)' \). The data involves the cross-classification of 132 long-term schizophrenic patients into three row categories describing the frequency of hospital visits and three column categories describing the length of stay. Evans et al. (1995) applied various integration techniques to the calculation of posterior means in this test case. In the example, they concluded that subregion adaptive integration (Genz, 1991) proved excessively time consuming, and a MCMC approach based on a Metropolis independence chain suffered from high correlations.

We applied the proposed adaptive importance sampling approach based on the restricted skew-Student distribution to this non-trivial problem. We chose \( N = 10,000,000 \) iterations in each adaptive
step for direct comparison with Student$_5$ importance sampling reported in Evans et al. (1995). We consider the estimation of the posterior mean of $\theta_1$ as this is the most problematic integral considered in Evans et al. (1995). In Table 2, we provide the exact value (based on 41 hours of Student importance sampling) and the adaptive skew-Student estimates after the fourth round of adaptation. We also provide the Normal (non-adaptive) importance sampling results. We observe that adaptive restricted skew-normal importance sampling provides an improvement over Normal importance sampling as the standard errors are reduced. We suspect that the improvements are not as great as in Example 1 as the posterior is not as heavily skewed. We observe a slight benefit in adaptive restricted skew-Student importance sampling over adaptive restricted skew-normal importance sampling. This supports the assertion that little is lost by choosing the longer tailed skew-Student over the skew-normal.

<table>
<thead>
<tr>
<th></th>
<th>Exact</th>
<th>Normal</th>
<th>Skew-normal</th>
<th>Skew-Student$_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimates</td>
<td>0.4215</td>
<td>0.4323</td>
<td>0.4298</td>
<td>0.4269</td>
</tr>
<tr>
<td>Std Error</td>
<td>(0.054)</td>
<td>(0.0486)</td>
<td>(0.04301)</td>
<td></td>
</tr>
</tbody>
</table>

Table 9.2: Estimates of the posterior mean of $\theta_1$ and standard errors corresponding to Example 2.

### 9.5 Discussion

This paper introduces new importance sampling algorithms for integrals defined on $\mathbb{R}^n$. The approach is adaptive and relies on the restricted skew-normal and restricted skew-Student families of distributions. Although there is no “universal” approach to multivariate integration, the proposed algorithms generalize the widely used approach based on multivariate normal importance sampling.

Like all integration techniques, our approach suffers from the curse of dimensionality where sampling and fitting become more complex as the dimension of the integral rises. Nevertheless, we have demonstrated the utility of the approach in several challenging examples.

No doubt, there are many variations of our algorithm that may be considered in future research. One potentially fruitful direction is the extension of our approach to more general skew-elliptical distributions.
Chapter 10

An online sequential Monte Carlo algorithm for dynamic models with the zero-inflated negative binomial distribution

10.1 Introduction

Recommender systems are becoming progressively important in real-world electronic marketing, such as Facebook, YouTube, and Amazon. They can provide users with personalized recommendations to help them choose the items they may like from thousands of products. Generally speaking, recommender systems can be divided into two categories. The first is content filtering approach, which creates a profile for each user or product to characterize its nature; an alternative approach is collaborative filtering, which only requires users’ past behaviour (Pazzani, 1999; Adomavicius et al., 2005; Koren et al., 2009).

Most of the literature for collaborative filtering focuses on recommendations for users with static preference (Bennett et al., 2007; Koren, 2009). However, users’ preference could change over time due to, for example, the change of users’ work and social environment. Hence, recommender systems based on static preference may no longer be valid if a user’s preference is dynamical.

We consider a collaborative filtering method based on the hidden Markov model (HMM) to account for users’ time-varying preferences. Hidden Markov models (HMMs) describe the relationship between the observed states and underlying hidden states, which has been widely used in sequence analysis, e.g. speech recognition (Gales and Young, 2008; Sha and Saul, 2007), human action recognition (Lee and Cho, 2011). In our HMM-based recommender system, the hidden states are users’ preferences for items (products), and the observations include counts of products rated by users and binary rating data. Since the counts of products in recommender system are very sparse and highly over-dispersed, we use zero-inflated negative binomial (ZINB) distribution to model them.

The common approach to estimate an HMM is to use the batch EM algorithm (Dempster et al., 1977). In the E-step, we model the distribution over hidden variables using the parameter estimates
we have obtained, and in the M-step we optimize the expected log-likelihood. The two steps in the EM algorithm increase the likelihood monotonically. However, the batch EM algorithm becomes impractical for large scale stream data, because we have to compute the expected log-likelihood function and optimize it using the whole data set when new data come in. For this reason, online algorithms, which use each data point only once and operate in a recursive manner, become more appealing since the computational complexity is constant in recursive steps. The online EM algorithm for HMM proposed by Cappé (2011) originates from batch EM. The key ingredient of the online EM algorithm is the recursive computation of smoothing functions required by the EM algorithm. However, the requirement that the emission distribution in the HMM belongs to an exponential family is quite strong in some real statistical models. For example, the zero-inflated negative binomial distribution in our proposed HMM-based recommender system does not belong to an exponential family when the dispersion parameter is unknown.

One classical approach of online inference for the hidden variables in HMMs is the sequential Monte Carlo (SMC) method (Doucet et al., 2001). Methods have been proposed for estimating the static parameter $\Theta$ of an HMM within the SMC algorithm (Fearnhead, 2002; Storvik, 2002; Andrieu et al., 1999). Another popular method is to construct an artificial kernel estimator for the posterior density of $\Theta$ given the observed data, which is called the “jittering” step. The key for this method is to control the variance of jittered particles by shrinking the density estimator. However, the dimension of the extended hidden states equals to the dimension of static parameter $\Theta$ plus the dimension of hidden states. If the dimension of parameters is quite large, this may lead to high requirement of computing resources and SMC will suffer from the degeneracy issue (Fearnhead, 2002; Liu and West, 2001).

Our contributions are summarized as follows. We first propose an HMM-based recommender system that better accounts for the dynamic preference of users and the sparseness of ratings. We develop an efficient online SMC-EM algorithm for HMMs, in which the emission distribution may not belong to an exponential family (for example, the ZINB distribution when the dispersion parameter is unknown). Our proposed method is related to the work in Olsson and Westerborn (2015) and Cappé (2009). But their HMMs are relatively simple with distributions from an exponential family and the parameter space is small. In contrast, our proposed HMM for recommender system has two main challenges: the emission distribution does not belong to an exponential family and the parameter space is large when there are hundreds and thousands of items to choose from. We specify the details of the implementation of our proposed algorithm for an HMM-based recommender system. We evaluate the performance of the proposed online SMC-EM algorithm by comparing with the batch EM algorithm using simulation studies and real data analysis. The proposed online SMC-EM algorithm can perform as well as the batch EM algorithm in terms of making predictions as a recommender system but is computationally more efficient.

The remainder of the article is organized as follows. We introduce the HMM-based dynamic model in Section 2 and the model estimation in Section 3. Section 4 focuses on model selection.
Section 5 is about the prediction of the recommender system. The simulation study and the real data analysis are provided in the last two sections.

### 10.2 HMM model for recommender system

We use $Z \in \{1, 2, \cdots, K\}$ to denote the users’ preference, $u \in \{1, 2, \cdots, U\}$ to denote the label of users, $t \in \{1, 2, \cdots, T\}$ to denote the time stamp. We assume that the total number of items is $J$, $I_{u,t} = \{I_{u,t,j}\}_{j=1:J}$ where $I_{u,t,j}$ is an indicator function denoting if the $j$-th item is rated by the $u$-th user at time $t$, that is

$$I_{u,t,j} = \begin{cases} 1 & \text{if item } j \text{ is rated by user } u \text{ at time } t \\ 0 & \text{otherwise.} \end{cases}$$

The total number of items rated by the $u$-th customer at time $t$ is $N_{u,t} = \sum_{j=1}^{J} I_{u,t,j}$. The task of a recommender system is to predict the product that a user will prefer in a time stamp $(T + 1)$ given all the products that the user has rated until time $T$.

Figure 10.1 shows the graphical representation of the HMM model. In rating data, users change the amount and type of items they rate over time. This indicates a distinct change in the user’s rating behavior. Let us define the preference of a user $Z_{u,t}$ to be a hidden state, which is a Markov chain. The observations, $I_{u,t}$ and $N_{u,t}$, in a given time period only depend on the hidden state $Z_{u,t}$ in that time period. A zero-inflated distribution is commonly used to model a distribution with frequent
zero-observed values. We consider using the zero-inflated negative binomial (ZINB) distribution to model the observed counts of rating.

Conditional on $Z_{u,t} = k$, we assume that $N_{u,t}$ is zero inflated negative binomial distributed with unknown parameters $\sigma_k, \mu_k$ and $a_k$, where

$$
P(N_{u,t}|a_k, \mu_k, \sigma_k) = \left[ \sigma_k + (1 - \sigma_k) \left( \frac{a_k}{\mu_k + a_k} \right)^{a_k} \right] 1(N_{u,t} = 0) + \left[ (1 - \sigma_k) \frac{\Gamma(N_{u,t} + a_k)}{\Gamma(N_{u,t} + 1)\Gamma(a_k)} \left( \frac{\mu_k}{\mu_k + a_k} \right)^{N_{u,t}} \left( \frac{a_k}{\mu_k + a_k} \right)^{a_k} \right] 1(N_{u,t} > 0),
$$

where $\sigma_k$ is the structure zero parameter that takes value between $[0, 1]$, $\mu_k$ is the mean parameter, $a_k$ is the dispersion parameter, and $1(\cdot)$ is an indicator function. There are two zero generating processes in the ZINB distribution. The first process generates structure zeros by a binary distribution with probability $\sigma_k$, and the second process generates counts of NB distribution.

Conditional on $Z_{u,t} = k$ and $N_{u,t}$, $I_{u,t}$ is multinomial distributed with parameters $\tau_k = (\tau_{1k}, \tau_{2k}, \ldots, \tau_{jk})'$. Hence, the emission model in this HMM is a product of the ZINB distribution and the multinomial distribution as follows

$$f(N_{u,t}, I_{u,t}|Z_{u,t} = k) = P(N_{u,t}|Z_{u,t} = k)P(I_{u,t}|N_{u,t}, Z_{u,t} = k).$$

We use $\pi$ to denote the starting probability distribution of the user preference, which is a $K \times 1$ vector. The $k$-th element is the probability that the user has preference $k$ at time $t = 1$. The transition probability matrix is represented by $P$, which is a $K \times K$ matrix. In this matrix, the element in the $i$-th row and the $j$-th column denotes the probability that the user changes its state from $i$ to $j$. We use a concise notation $Y_{u,t}$ to denote the observations, $(N_{u,t}, I_{u,t})$.

Let $\tau = (\tau_1', \ldots, \tau_K')'$, $\mu = (\mu_1, \ldots, \mu_K)'$, $a = (a_1, \ldots, a_K)'$, and $\sigma = (\sigma_1, \ldots, \sigma_K)'$. Let $\Theta$ be a vector of all elements of $P$, $\tau$, $\mu$, $a$ and $\sigma$. The joint likelihood function of $Y_{u,1:T}$ and hidden states $Z_{u,1:T}$ for this HMM is

$$P(Z_{u,1:T}, Y_{u,1:T} | \Theta) = \pi(Z_{u,1}) \prod_{t=2}^{T} P(Z_{u,t}|Z_{u,t-1}) \prod_{t=1}^{T} \left[ P(N_{u,t}|Z_{u,t} = k)P(I_{u,t}|N_{u,t}, Z_{u,t} = k) \right].$$

The marginal likelihood can be computed by taking the advantage of factorization (Sahoo et al., 2010).

10.3 Estimation

10.3.1 Batch EM algorithm

One common way to estimate this HMM is the EM algorithm (Dempster et al., 1977). In the E-step, we compute the expectation of log likelihood function over the hidden variables using the
values of parameters obtained so far and observations. In the **M-step**, we maximize the expected log likelihood over the parameters.

The expected log likelihood can be written in the form:

\[
Q(\Theta|\Theta^{old}) = \sum_{u,k} P(Z_{u,1} = k|Y_{u,1:T}, \Theta^{old}) \cdot \log \pi(Z_{u,1} = k) + \sum_{u,t,k} P(Z_{u,t-1} = j, Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) \cdot \log P(Z_{u,t} = k|Z_{u,t-1} = j) + \sum_{u,t,k} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) \cdot \log f(Y_{u,t}|\tau, a_k, \mu_k),
\]

where

\[
P(Z_{u,t}|Y_{u,1:T}, \Theta^{old}) \propto \alpha(Z_{u,t}) \cdot \beta(Z_{u,t}),
\]

\[
P(Z_{u,t-1}, Z_{u,t}|Y_{u,1:T}, \Theta^{old}) \propto \alpha(Z_{u,t-1}) \cdot P(Z_{u,t}|Z_{u,t-1})f(Y_{u,t}|Z_{u,t})\beta(Z_{u,t}),
\]

\[
\alpha(Z_{u,t}) \propto \sum_{Z_{u,t-1}} \alpha(Z_{u,t-1})P(Z_{u,t-1}|Z_{u,t-1})f(Y_{u,t}|Z_{u,t}),
\]

\[
\beta(Z_{u,t}) \propto \sum_{Z_{u,t+1}} \beta(Z_{u,t+1})P(Z_{u,t+1}|Z_{u,t}) \cdot f(Y_{u,t+1}|Z_{u,t+1}).
\]

The \(\alpha(Z_{u,t})\) is computed during a forward pass over a data sequence, when \(t = 1\), \(\alpha(Z_{u,1}) \propto \pi(Z_{u,1}) \cdot P(Y_{u,1}|Z_{u,1})\). The \(\beta(Z_{u,T})\) is computed during a backward pass over a data sequence, when \(t = T\), \(\beta(Z_{u,T}) = \alpha(Z_{u,T})\).

In the **M-step**, we maximize \(Q(\Theta|\Theta^{old})\). The MLE of \(\pi(Z_{u,1})\), \(P(Z_{u,t}|Z_{u,t-1})\), \(\tau\) and \(\sigma\) has closed analytical form. The MLE of \(\mu\) and \(\alpha\) in the ZINB distribution can be obtained through numerical optimization. The formulas are derived in Appendix 10.9.

In the following sections, we omit the subscript \(u\) for simplicity of the notation when there is no confusion.

### 10.3.2 Online SMC-EM for HMM recommender system

Using online (recursive) algorithms for estimating hidden Markov models is an interesting topic in time series modeling. In the batch algorithm, the computation can be prohibitive for large scale data. Instead of using all data in estimation, we run the data only once sequentially. For example, if \(\Theta_u\) is our static parameter estimate based on \(n\) observations, after observing the new data point \(y_{n+1}\), an online algorithm could update the estimate to \(\Theta_{n+1}\) by using \(y_{n+1}\) and some low dimensional summary statistics of \(y_{1:n}\), which means each observation \(y_i\) is only used once.

**Online EM algorithm**

Cappé (2011) proposed an online EM algorithm for hidden Markov models by combining the usage of complete sufficient statistics and the recursive form of smoothing of hidden Markov model.
The key assumption of the online EM algorithm is that the model belongs to an exponential family (\textit{i.e.} Assumption 1 in Cappé (2011)), such that we can recursively update the E-step and the M-step.

We denote the model complete sufficient statistics by \(s(Z_{t-1}, Z_t, Y_t)\), where \(Z_t\) refers to hidden states and \(Y_t\) refers to the observations. In the E-step, we compute

\[
Q(\Theta|\Theta^{old}) = \int \phi_t(z) \rho_t(z) dz = \frac{1}{t} E_Z \left[ \sum_{n=1}^t s(Z_{n-1}, Z_n, Y_n)|Y_{1:t} \right],
\]

where the filtering density \(\phi_t(z)\) and the auxiliary function \(\rho_t(z)\) can be constructed recursively. In the M-step, we update the parameter \(\Theta\) by maximizing \(Q(\Theta|\Theta^{old})\). The recursive construction of \(\phi_t(z)\) and \(\rho_t(z)\) is detailed in Section 2.2 of Cappé (2011).

The requirement of complete sufficient statistics \(s(Z_{t-1}, Z_t, Y_t)\) is not a realistic assumption in many statistical models. For example, the dispersion parameter of the ZINB distribution does not have complete sufficient statistics. Hence, this algorithm cannot be applied to many hidden Markov models, in which complete sufficient statistics cannot be obtained.

**SMC algorithm**

Sequential Monte Carlo (Doucet et al., 2001; Liu and Chen, 1998), also known as particle filtering, provides a scheme to do online inference. We describe an SMC scheme that approximates recursively the sequence of filtering density \(p_\Theta(z_T|y_{1:T})\) for a fixed static parameter value \(\Theta\) in Algorithm 1. To be concise, particle and time indices are always respectively superscripts and subscripts, the letter \(f\) refers to probability densities defined by the model, and \(M\) is the number of particles.

**Algorithm 23** Sequential Monte Carlo bootstrap filter

1: **Input:** data \(y_{1:T}\), parameter \(\Theta\)
2: **Output:** \(\{z_i^t, W_i^t\}_{1 \leq i \leq M}\)
3: Draw \(z_i^1 \sim \pi_{1,\Theta}(\cdot)\);
4: Compute and normalize the weights:
   \[
   w_{1,\Theta}^1 = f_\Theta(y_1|z_i^1), \quad W_{1,\Theta}^1 = w_{1,\Theta}^1 / \sum_{i=1}^M w_{1,\Theta}^i;
   \]
5: **for** \(t = 2\) to \(T\) **do**
   6: Sample the ancestor of particle \(i\) at time \(t\), \(A_i^t \sim\) Discrete(\(\{W_{i-1,\Theta}^j\}_{1 \leq j \leq M}\));
   7: Sample \(z_i^t \sim P_\Theta(\cdot|z_{i-1}^{A_i^t})\);
   8: Compute and normalize the weights:
   \[
   w_{t,\Theta}^i = f_\Theta(y_t|z_i^t), \quad W_{t,\Theta}^i = w_{t,\Theta}^i / \sum_{i=1}^M w_{t,\Theta}^i;
   \]

At iteration \(t\), the following quantity

\[
\hat{p}(y_t|y_{1:t-1}, \Theta) = \frac{1}{M} \sum_{i=1}^M w_{t,\Theta}^i(z_i^t)
\]
can be used to construct an unbiased estimator of $p(y_{1:t}|\Theta)$ (Chopin et al., 2013; Andrieu et al., 2010),
\[
\hat{p}(y_{1:t}|\Theta) = \prod_{n=1}^{t} \left( \frac{1}{M} \sum_{i=1}^{M} w_{n,i}(z_{n}^{i}) \right).
\] (3)

When our inference relies on static parameter $\Theta$ and hidden states $Z_{1:T}$, we augment the hidden states by incorporating the static parameter to obtain a combined sample $\{z_{i}^{j}, \Theta_{i}^{j} : i = 1, \ldots, M\}$ and associated weights $\{W_{i}^{j} : i = 1, \cdots, M\}$. We propose to use a kernel smoothing method (Liu and West, 2001; Carvalho and Lopes, 2007) to construct an artificial density. The smooth kernel density is given by
\[
p(\Theta_{t+1}^{j}|\Theta_{t}^{1:M}) \sim N(b\Theta_{t}^{j} + (1 - b)\bar{\Theta}_{t}, h^{2}V_{t}),
\] (4)

where $b = (3\delta - 1)/2\delta$, $h^{2} = 1 - ((3\delta - 1)/2\delta)^{2}$, $\delta$ is a tuning parameter which typically takes value around 0.95 – 0.99, and $\bar{\Theta}_{t}$ and $V_{t}$ are respectively the Monte Carlo mean and variance of $\Theta_{t}^{1:M}$. This method can perform satisfactorily in practice (Kantas et al., 2015; Flury and Shephard, 2011). By the usage of SMC, we are able to obtain the approximate posterior distribution
\[
\hat{p}(z_{1:T}, \Theta|y_{1:t}) = \sum_{i=1}^{M} W_{i}^{j} \delta_{z_{1:T}, \Theta}(z_{1:T}, \Theta),
\]
where $\delta_{x_{0}}(x)$ denotes the Dirac delta mass located at $x_{0}$. We could obtain the estimates for the marginal posterior distribution of $p(z_{1:T}|y_{1:t})$ and $p(\Theta|y_{1:t})$ by simply dropping the particles $\Theta_{1:M}$ and $\{z_{1:T}^{j}\}_{1 \leq i \leq M}$, respectively, as follows:
\[
\hat{p}(z_{1:T}|y_{1:t}) = \sum_{i=1}^{M} W_{i}^{j} \delta_{z_{1:T}}(z_{1:T}),
\]
and
\[
\hat{p}(\Theta|y_{1:t}) = \sum_{i=1}^{M} W_{i}^{j} \delta_{\Theta}(\Theta).
\]

The degeneracy issue becomes more severe when introducing static parameters $\Theta$ into the state space, especially for high dimensional $\Theta$. In the HMM for recommender system, the emission distribution is a product of ZINB and multinomial, and the dimension of multinomial parameter $\tau$ for rating data is generally very high.

**Online SMC-EM algorithm**

To obtain a practical method, we propose an online SMC-EM algorithm. Our new algorithm originates from the Monte Carlo EM (MCEM) algorithm (Wei and Tanner, 1990; Booth and Hobert, 1999), sequential Monte Carlo algorithms and the former SMC-EM work (Olsson and Westerborn, 2015; Cappé, 2009). The ZINB distribution belongs to the exponential family conditional on the dispersion parameter $\alpha$, hence we are able to obtain the complete sufficient statistics for all the other
parameters in the HMM conditional on $a$. We use $\Theta$ to denote a vector of all elements of $P$, $\tau$, $\mu$ and $\sigma$. We omitted initial probability $\pi$ in the online estimation as the influence of this term vanishes with $t$. In the following proposition, we show the complete sufficient statistics of the HMM described in (Figure 10.1) with ZINB dispersion parameter $a$ assumed to be known.

**Proposition 1:** Assuming dispersion parameter $a$ of ZINB is known, the sufficient statistics $s(Z_{n-1}, Z_n, Y_n)$ for the parameters $\Theta$ in the HMM model (Figure 10.1) given $(N_n, I_n)$ for $n = 1, \ldots, t$ can be written as

\[
s(Z_{n-1}, Z_n, Y_n) = \left\{ 1(Z_{n-1} = i, Z_n = j), 1(Z_n = i)I_n, 1(Z_n = i)1(N_n = 0), 1(Z_n = i)1(N_n > 0) \right\}_{i,j=1}^K.
\]

To construct online EM algorithm for the HMM, we consider extending the latent state by incorporating the dispersion parameter $a$, such that $X_{1:t} = (Z_{1:t}, a)$. By integrating the model static parameter $a$ out of the joint likelihood makes the emission distribution exponential family distributed. Then in the E-step, we compute recursively the normalized $Q(\Theta | \Theta^{old}) = \frac{1}{t} E_{\Theta^{old}}[\sum_{n=1}^{t} s(Z_{n-1}, Z_n, Y_n)|Y_{1:t}]$ by the filtering density $\phi_t(x) = p(x_t | y_{1:t})$ and the auxiliary function $\rho_t(x) = \frac{1}{t} E[\sum_{n=1}^{t} s(Z_{n-1}, Z_n, Y_n)|Y_{1:t}, X_t = x]$, where $X_t = (Z_t, a)$, such that

\[
\int \phi_t(x) \rho_t(x) dx = \frac{1}{t} E \left[ \sum_{n=1}^{t} s(Z_{n-1}, Z_n, Y_n)|Y_{1:t} \right].
\]

The $m$-th particle for $X_t$ is denoted by $x_t^m = (z_t^m, a^m)$. The filtering density is approximated by

\[
\hat{\phi}_t(x) = \hat{p}(a, z_t|y_{1:t}) = \sum_{i=1}^{M} W_i^t \delta_{(a^i, z_t^i)}(a, z_t)
\]

using the particles, $\{x_t^m\}_{1 \leq m \leq M}$, obtained from the SMC algorithm. The left hand side of Eq. (5) can be approximated by particles:

\[
\int \phi_t(x) \rho_t(x) dx \approx \sum_{m=1}^{M} W_t^m \rho_t^m,
\]

where $\rho_t^m$ denotes the $m$-th particle for $\rho_t$. 

167
The five components of $\mathbf{\rho}_t$ are of the following form:

$$g^m_{1,t}(i, j) = \frac{1}{t} E \left[ \sum_{n=1}^{t} 1(Z_{n-1} = i, Z_n = j | Y_{1:t}, X_t = x^m_t) \right].$$

$$g^m_{2,t}(i) = \frac{1}{t} E \left[ \sum_{n=1}^{t} 1(Z_n = i) I_a | Y_{1:t}, X_t = x^m_t \right].$$

$$g^m_{3,t}(i) = \frac{1}{t} E \left[ \sum_{n=1}^{t} 1(Z_n = i) N_1 | Y_{1:t}, X_t = x^m_t \right].$$

$$g^m_{4,t}(i) = \frac{1}{t} E \left[ \sum_{n=1}^{t} 1(Z_n = i) 1(N_n = 0) | Y_{1:t}, X_t = x^m_t \right].$$

$$g^m_{5,t}(i) = \frac{1}{t} E \left[ \sum_{n=1}^{t} 1(Z_n = i) 1(N_n > 0) | Y_{1:t}, X_t = x^m_t \right].$$

Note that $\mathbf{\rho}_t$ is a vector $\mathbf{\rho}_t = (\text{Vec}(g^m_{1,t}), \text{Vec}(g^m_{2,t}), \text{Vec}(g^m_{3,t}), \text{Vec}(g^m_{4,t}), \text{Vec}(g^m_{5,t}))$, where $g^m_{1,t}$ is a matrix of $K \times K$, $g^m_{2,t}$ is a matrix of length $J \cdot K$, $g^m_{3,t}$, $g^m_{4,t}$ and $g^m_{5,t}$ are all vectors of length $K$, and Vec vectorizes a matrix. The expectations are taken with respect to latent states $Z_{1:t}$.

In the E-step, we derive particle approximation $\hat{g}^m_{1,t}(i, j)$, $\hat{g}^m_{2,t}(i)$, $\hat{g}^m_{3,t}(i)$, $\hat{g}^m_{4,t}(i)$ and $\hat{g}^m_{5,t}(i)$ through the following recursion form using the sampled particles:

$$\hat{g}^m_{1,t}(i, j) = \gamma_t 1(Z^m_{t-1} = i, Z^m_t = j) + (1 - \gamma_t) \hat{g}^m_{1,t-1}(i, j), \quad (6)$$

$$\hat{g}^m_{2,t}(i) = \gamma_t 1(Z^m_t = i) I_a + (1 - \gamma_t) \hat{g}^m_{2,t-1}(i), \quad (7)$$

$$\hat{g}^m_{3,t}(i) = \gamma_t 1(Z^m_t = i) N_1 + (1 - \gamma_t) \hat{g}^m_{3,t-1}(i), \quad (8)$$

$$\hat{g}^m_{4,t}(i) = \gamma_t 1(Z^m_t = i) 1(N_t = 0) + (1 - \gamma_t) \hat{g}^m_{4,t-1}(i), \quad (9)$$

$$\hat{g}^m_{5,t}(i) = \gamma_t 1(Z^m_t = i) 1(N_t > 0) + (1 - \gamma_t) \hat{g}^m_{5,t-1}(i), \quad (10)$$

where $(\gamma_t)_{t \geq 1}$ is a decreasing sequence of step sizes, with $\sum_{t \geq 1} \gamma_t = \infty$ and $\sum_{t \geq 1} \gamma_t^2 < \infty$. In the HMM case, Cappé (2011) recommended the step sizes $\gamma_t$ with form $\gamma_t = t^{-\alpha}$, where $\alpha$ is within interval $(0.5, 0.8)$. The choice of $\gamma_t = t^{-1}$ should be avoided, as this special choice of step sizes is not able to provide robust estimates of parameters.

**Proposition 2:** Our particle approximations $\hat{\mathbf{\rho}}^m_t$ (Eq. 6 - Eq. 10) are properly adjusted, such that

$$\sum_{m=1}^{M} W^m_t \hat{\mathbf{\rho}}^m_t \to \int \phi_t(x) \mathbf{\rho}_t(x) dx, \quad \text{as } m \to \infty,$$

where $m$ denotes the particle index for $\hat{\mathbf{\rho}}_t$. The justification of this proposition is shown in Appendix 10.10.
In the **M-step**, we update the parameters as follows

$$
\tilde{q}_{t,i}(i, j) = \sum_{m=1}^{M} \tilde{q}_{t,i}^{m}(i, j) W_{t}^{m}, l = 1, \cdots, 5. 
$$  \hfill (11)

$$
\tilde{P}(Z_t = j | Z_{t-1} = i) = \frac{\tilde{q}_{t,i}(i, j)}{\sum_{j'}^{K} \tilde{q}_{t,i}(i', j')}, \hfill (12)
$$

$$
\hat{\sigma}_{i} = \frac{\tilde{q}_{4,i}(i) - [\tilde{q}_{4,i}(i) + \tilde{q}_{5,i}(i)](\frac{\hat{a}_{i}}{\hat{\mu}_{i} + \hat{a}_{i}})^{\hat{a}_{i}}}{[\tilde{q}_{4,i}(i) + \tilde{q}_{5,i}(i)](1 - (\frac{\hat{a}_{i}}{\hat{\mu}_{i} + \hat{a}_{i}})^{\hat{a}_{i}})} \hfill (13)
$$

$$
\hat{\tau}_{i} = \frac{\tilde{q}_{2,i}(i)}{\tilde{q}_{3,i}(i)}. \hfill (14)
$$

The MLE of $\mu_{i}$ does not have a closed form. Instead, we optimize the following equation to obtain the MLE of $\mu_{i}$:

$$
Q_{i} = \tilde{q}_{4,i}(i) \log \left( \hat{\sigma}_{i} + (1 - \hat{\sigma}_{i}) (\frac{\hat{a}_{i}}{\hat{\mu}_{i} + \hat{a}_{i}})^{\hat{a}_{i}} \right) + \tilde{q}_{3,i}(i) \log \frac{\mu_{i}}{\mu_{i} + \hat{a}_{i}} - \tilde{q}_{5,i}(i) \hat{a}_{i} \log (\mu_{i} + \hat{a}_{i}). \hfill (15)
$$

The estimates of $a$ can be obtained by simply taking the weighted average of sampled particles $\{a^{m}, W_{t}^{m}\}_{1 \leq m \leq M}$.

We describe our online SMC-EM algorithm in Algorithm 2, in which we omit the subscript $u$ for simplicity. Let $\tilde{\alpha} = \log(\alpha)$. In our numerical experiment, we let $\delta = 0.99$. We choose a number $n_{\text{min}}$ that is not too small to make sure that the M-step is numerically well-behaved since MLE may be degenerated for a very small sample size. For instance, we choose $n_{\text{min}} = 20$ in our simulation study.

### 10.4 Model selection

In the HMM model of Section 10.2, the total number of hidden states $K$ corresponds to the total number of user preferences to products. We want to find the $K$ that can “best” explain the trajectory of user preference, which is a model selection problem. A small value of $K$ could give us a simple model, but may not be able to capture the complexity of data, while a large value of $K$ may overfit the data. Akaike Information Criterion (AIC, Akaike, 1974) and Bayesian Information Criterion (BIC, Schwarz et al., 1978) are two commonly used model selection methods to determine the number of hidden states $K$ in HMM. In our case, the AIC and BIC values of a $K$-state HMM are, respectively, $AIC_{K} = -2 \log L(\hat{\Theta}|Y_{u,1:T}|_{1 \leq u \leq U}) + 2 \cdot (K^{2} + (J + 3)K - 1)$, and $BIC_{K} = -2 \log L(\hat{\Theta}|Y_{u,1:T}|_{1 \leq u \leq U}) + \log N \cdot (K^{2} + (J + 3)K - 1)$, where $\hat{\Theta}$ are the MLE of parameters, $K^{2} + (J + 3)K - 1$ is the total number of parameters, $K^{2} - K$ for the transition matrix, $K - 1$ for the initial distribution of the first hidden state, $3K$ for the zero inflated negative binomial parameters and $J \times K$ for the multinomial parameters. The optimum $K$ is the one that minimizes $AIC_{K}$ or $BIC_{K}$. Sahoo et al. (2010) found that the BIC criterion penalizes models too aggressively. Hence, we mainly focus on the AIC criterion.
Algorithm 24 Online SMC-EM algorithm

1: **Input:** data \(y_{1:T}\), initial value \(\Theta_0\), step sizes \(\gamma_t\), \(t \geq 1\).
2: **Output:** parameter estimates: \(\hat{\Theta}\).
3: **Initialization:** set \(\hat{\rho}_i^0 = 0\), for \(i = 1, 2, \cdots, M\).
4: Draw \(z_{1}^i \sim \pi_{1,\Theta}(\cdot)\) and \(\hat{a}_{1}^i \sim N(\hat{\alpha}_0, V_0)\).
5: Compute and normalize the weights
   \[
   w_{1,\Theta}^i = f_{\Theta}(y_1|z_{1}, a_{1}^i); \quad W_{1,\Theta} = \frac{w_{1,\Theta}^i}{\sum_{i=1}^{M} w_{1,\Theta}^i}.
   \]
6: Adjust auxiliary variable \(\hat{\rho}_{1}^i\) using Eq (6-10).
7: **for** \(t = 2\) to \(T\) **do**
8: Sample the ancestor of particle \(i\) at time \(t\), \(A_{t}^i \sim \text{Discrete}([W_{t-1,\Theta}^m]_{1 \leq m \leq M})\);
9: Sample \(z_{t}^i \sim P_{\Theta}(\cdot|z_{t-1}^{A_{t-1}^i}), \hat{a}_{t}^i \sim p(\cdot|\hat{a}_{t-1}^{m})_{1 \leq m \leq M}\), where \(p(\cdot|\hat{a}_{t-1}^{m})_{1 \leq m \leq M}\) is the artificial kernel constructed by Eq. (4);
10: Compute and normalize the weights:
    \[
    w_{t,\Theta}^i = f_{\Theta}(y_{t}|z_{t}^i, a_{t}^i); \quad W_{t,\Theta} = \frac{w_{t,\Theta}^i}{\sum_{i=1}^{M} w_{t,\Theta}^i}.
    \]
11: Adjust auxiliary variable \(\hat{\rho}_{t}^i\) using Eq (6-10).
12: **if** \(t \geq n_{\text{min}}\) **then**
13: Update the parameters using Eq (12-15).

It is well observed in the literature that the EM algorithm converges to local maximum (Dempster et al., 1977). Thus, the choice of the initial value of EM algorithm is critical. For batch EM and online SMC-EM, we choose a large number of initial values and run for each initial value until convergence is achieved. We choose the \(\hat{\Theta}\) that maximize \(\log L(\Theta|\{Y_{u,1:T}\}_{1 \leq u \leq U})\) as our MLE.

### 10.5 Prediction

The task of a recommender system is to predict the product that a user will prefer at a time stamp \((T + 1)\) given all the products the user has rated until \(T\). Algorithm 25 describes the prediction of the recommender system. The performance of a recommender system can be measured by Precision (P), Recall (R) and the harmonic mean of Precision and Recall (F) (Sahoo et al., 2010), which are defined as follows:

- **Precision (P):** fraction of the recommended items that is correct.
- **Recall (R):** fraction of the correct items that is recommended.
- The harmonic mean of P and R, denoted F, is computed by
  \[
  \frac{1}{F} = \frac{1}{2} \left( \frac{1}{P} + \frac{1}{R} \right).
  \]

Larger values of P, R and F indicate better performance of the algorithm.
Algorithm 25 Online recommender system

1: **Input:** data \( y_{1:T} \);
2: **Output:** \( N \) items recommended for users;
3: **for** HMM with \( K \) hidden states, \( K = 2, 3, \ldots \) **do**
4: **for** each initial value \( \Theta_0 \) **do**
5: **E-step:** Recursively compute \( \{ \varrho^m_T \}_{1 \leq m \leq M} \) and \( \{ \phi^m_T \}_{1 \leq m \leq M} \) as described in Algorithm 2;
6: **M-step:** Update the parameters using Eq. (12-15);
7: Select the HMM model with lowest AIC or BIC, choose \( \hat{\Theta} \) that maximizes the marginal likelihood function.
8: For each user \( u \), we compute \( P(u, j) \) the probability that item \( j \) will be rated at time \( T + 1 \) (See Appendix 10.11).
9: Recommend top \( N \) items with highest \( P(u, j) \).

### 10.6 Simulation

#### 10.6.1 Large sample behavior of online SMC-EM

In this section, we use a simple example to illustrate the behavior of parameter estimates for online SMC-EM when \( T \) gets large. We consider our algorithm in a three-state HMM-based recommender system. Figure 10.2 illustrates the behavior for the parameter estimates obtained by using the online SMC-EM algorithm as a function of the time stamp \( t \). We choose \( \gamma_n = n^{-0.8} \) as the step size. To guarantee the good behavior of MLE, we do not update the parameters when \( t < 20 \). We observe that parameter estimates converge towards the true values with increments of data. It also indicates that the online SMC-EM requires a long data sequence to achieve convergence.

![Figure 10.2: Trace plots of parameter estimates over time.](image-url)
10.6.2 Online recommender system

In this simulation study, we assume there are $U = 50$ users, the total length of time stamps is $T = 3000$, there are $K = 8$ preference states. The total number of items $J = 80$. The total number of items a user prefers at time $t$ is $N_u,t$. For each user, we simulate their preference changing through their state transition probability matrix $P$. To better model the preference change, we set low self transition probability for most of the hidden states. The total number of items a user rated at the $k$-th preference state is simulated from the ZINB distribution with mean parameter $\mu_k$, dispersion parameter $a_k$ and structure zero parameter $\sigma_k$. The items a user prefers at a specific state $k$ is generated using a multinomial distribution with the total number of items $N_u,t$ and the parameter $\tau_k$. We simulate 100 data sets in total. For each data set, we use the simulated data for $t = 1, 2, \cdots, T-1$ as a training set, and use the data generated at $T$ as the testing set. In the online SMC-EM algorithm, it is critical to choose a proper number of particles $M$. A small value of $M$ can lower the memory cost but the algorithm may suffer from particle degeneracy issues. A large value of $M$ can provide more accurate estimates, but imposes heavier computational burden. An alternative strategy is to start with a small value of $M$, and then increase $M$ as the EM algorithm progresses. Generally, a larger number of particles $M$ is required for a model with a larger $K$. For our simulation setting, the number of particles $M = 1000$ suffices. We choose to use $n^{-0.8}$ as the step size $\gamma_n$ for the online EM algorithm.

We compare the parameter estimates and the performance in prediction using our proposed algorithm and the batch EM algorithm. Figure 10.3 shows estimated transitional probability of both algorithms are very close to the true transition probability matrix, which indicates that both batch EM and online EM can well uncover the patterns of change in user preferences. Figure 10.4 shows the boxplots of the estimates of $\mu, a, \tau, \sigma$ and $P$ provided by both algorithms are very close, while the estimation variance of online SMC-EM is larger. Figure 10.5 displays the comparison of prediction for two algorithms. The online SMC-EM achieves similar performance with the batch EM algorithm.
on Precision, Recall and their harmonic mean. We further use a paired t-test to demonstrate there is no significant difference between two algorithms in terms of P, R and F.

We also compare the computing time for the online SMC-EM and the batch EM algorithm (Intel Core i5, 1.3 GHz processor). The numerical analysis is conducted in R software. Each iteration of batch EM costs 501.162 seconds, while it only costs 6.476 seconds for one step of our online SMC-EM algorithm. This indicates that if we observe one more observation at next time stamp, our new algorithm can be hundreds times faster than batch EM algorithm. More details of this simulation study is described in supplementary material.

Figure 10.4: Boxplots for parameter estimates.

Figure 10.5: Precision (P), Recall (R) and their harmonic mean (F) for the online SMC-EM and the batch EM of the simulation study.
10.7 Application

10.7.1 Movie recommendation

The first real data set we consider is the Amazon movie rating data provided by He and McAuley (2016). This data set consists of the movie rating of users as well as the time stamps corresponding to the movies. We consider the data from June 1999 to July 2014, which spans over 15 years. We use the data from movies that have been rated over 30 times and users that have rated for more than 70 movies. This results in a data set with \( U = 270 \) users and \( J = 469 \) items. The length of time \( T = 91 \). Each time stamp refers to a period of two months. The observed matrix of \( N \) is sparse, 70% of the elements equals to 0. The variance of \( N \) is 7.378, which is much larger than the mean value of \( N \), 0.702, which indicates the assumption of ZINB distribution in the emission model is quite reasonable.

The convergence of the online SMC-EM algorithm cannot be achieved if the data sequence is not long enough. Hence, directly applying the online algorithm to short sequences may be questionable. Instead of using the batch EM for such short sequences, we can also apply the SMC-EM to the concatenated sequence \( \{Y_{1:T}, Y_{1:T}, \ldots\} \) (Yildirim et al., 2013). We fit the data with the SMC-EM and the batch EM. We leave \( y_{91} \) as the testing set and \( y_{1:90} \) as the training set. We apply the SMC-EM to the concatenated sequences \( \{Y_{1:90}, Y_{1:90}, \ldots Y_{1:90}\} \), where \( Y_{1:90} \) is replicated for 10 times. The number of particles we use is \( M = 500 \). The model selection criterion AIC is used to choose the number of user preference \( K \). The optimum number of user preference is \( K = 5 \). Both algorithms indicate that the self-transition probability of HMM is very high, which means that the user’s preference in movies does not change frequently (see Figure 10.6). The top 20 prediction made by the two algorithms are similar in terms of \( P \), \( R \), and \( F \). The \( P \), \( R \), and \( F \) for top 20 movies predicted by the SMC-EM are 0.0015, 0.0042, 0.0022 respectively. The \( P \), \( R \), and \( F \) for top 20 movies predicted by the batch EM are 0.0019, 0.0043 and 0.0026 respectively.

10.7.2 Crime data

The second real data set we consider in this paper is the data of crimes in Vancouver (Canada) from 2003 to 2017 (https://www.kaggle.com/damianpanek/vancouver-crime/data). Our interest is to learn the behavior different type of crimes in different areas, and predict what type of crime might happen in future for every area. The total number of regions in Vancouver area is \( U = 24 \), and the number of different type of crimes \( J \) is 11. The number of time stamps is \( T = 5303 \). Every time stamp is one day. The crime data is reported every day, hence it would be of particular interest to develop an online algorithm to analyze this type of sequential data. The percent of \( N \) that equals to zero is 21.6%, and \( N \) is highly over-dispersed, \( E(N) = 3.72 \) and \( Var(N) = 27.25 \).

We use the HMM proposed in this paper to model the crime data and use the batch EM algorithm and online SMC-EM to estimate the model. We use \( M = 500 \) particles and set the step size \( \gamma_n = n^{-0.8} \). We also leave the last observation as the testing set. The model selection criterion AIC indicates that the optimum number of latent classes \( K = 4 \). We first report the computing time for both algorithms...
(Intel Core i5, 1.3 GHz processor), this real data analysis is conducted using R software. One batch iteration costs 597.780 seconds, while an online EM iteration only takes 0.728 seconds. The P, R, and F for top 5 crime predicted by SMC-EM are 0.208, 0.627 and 0.313 respectively. The P, R, and F for top 5 crime predicted by the batch EM algorithm are 0.200, 0.606 and 0.301. Our algorithm outperforms the batch EM in terms of the prediction in this application.

10.8 Discussion

We have proposed an efficient online SMC-EM algorithm to infer the hidden states and estimate the static parameters of an HMM in which the emission distribution does not belong to an exponential family and the parameter space is large. The computational complexity of the batch EM algorithm increases linearly with the length of the sequence. In contrast, our proposed algorithm only requires storing the auxiliary variable that is computed in a recursive manner. Consequently, the online SMC-EM can greatly save computational resources for long sequence data.

The proposed SMC-EM algorithm has been applied to an HMM based dynamic recommender system where the zero-inflated negative Binomial distribution is involved. We demonstrate in the simulation studies and the real data analysis that our method can perform as good as the batch algorithm in terms of parameter estimation and making recommendations. Note that the applications of our proposed method are beyond recommender systems. It is applicable for more general models where we want to use particle approximation for the hidden variables and some of the parameters, and use the EM for updating the other parameters.

With the development of our efficient algorithm, better model based recommender systems can be explored as future work. The current version of the recommender system using the HMM dynamic
model can be improved in the following directions: first, the recommender system should have a mechanism to recommend items that have not been rated; second, it can incorporate more features of items into the recommendation, e.g. the rating score of the item. We leave these developments as our future work.

10.9 APPENDIX 1: Batch EM for HMM

In the M-step, we maximize $Q(\Theta|\Theta^{old})$. The MLE of $\pi(Z_{u,t})$, $P(Z_{u,t}|Z_{u,t-1})$, $\tau$ and $\sigma$ has the following closed analytical form:

\begin{equation}
\hat{\pi}(Z_{u,1} = k) = \frac{\sum_u P(Z_{u,1} = k|Y_{u,1:T}, \Theta^{old})}{\sum_{u,k} P(Z_{u,1} = k|Y_{u,1:T}, \Theta^{old})},
\end{equation}

\begin{equation}
\hat{P}(Z_{u,t} = k|Z_{u,t-1} = j) = \frac{\sum_{u,t} P(Z_{u,t-1} = j, Z_{u,t} = k|Y_{u,1:T}, \Theta^{old})}{\sum_{u,t,j} P(Z_{u,t-1} = j, Z_{u,t} = k|Y_{u,1:T}, \Theta^{old})},
\end{equation}

\begin{equation}
\hat{\tau}_k = \frac{\sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) \overline{I}_{u,t}}{\sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) N_{u,t}},
\end{equation}

\begin{equation}
\hat{\sigma}_k = \left\{ \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) I(N_{u,t} = 0) - \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) \left(\frac{\hat{a}_k}{\mu_k + \hat{a}_k}\right) \right\}
\end{equation}

\begin{equation}
\sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) \left[1 - (\frac{\hat{a}_k}{\mu_k + \hat{a}_k})^\hat{a}_k\right].
\end{equation}

The MLEs of $\mu_k$ and $a_k$ do not have closed forms. We could optimize the following formulas to obtain the MLEs of $\mu_k$ and $a_k$:

\begin{equation}
Q(\mu_k) = \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) I(N_{u,t} = 0) \cdot \log \left(\hat{\sigma}_k + (1 - \hat{\sigma}_k)\left(\frac{\hat{a}_k}{\mu_k + \hat{a}_k}\right)\hat{a}_k\right)
\end{equation}

\begin{equation}
+ \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) N_{u,t} \log \frac{\mu_k}{\mu_k + \hat{a}_k}
\end{equation}

\begin{equation}
- \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) I(N_{u,t} > 0) \hat{a}_k \cdot \log(\mu_k + \hat{a}_k),
\end{equation}

\begin{equation}
Q(a_k) = \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) I(N_{u,t} = 0) \cdot \log \left(\hat{\sigma}_k + (1 - \hat{\sigma}_k)\left(\frac{\hat{a}_k}{\mu_k + \hat{a}_k}\right)\hat{a}_k\right)
\end{equation}

\begin{equation}
+ \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) I(N_{u,t} > 0) \cdot \log \Gamma(N_{u,t} + a_k) - \log \Gamma(\hat{a}_k)
\end{equation}

\begin{equation}
+ \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) N_{u,t} \log \frac{\hat{\mu}_k}{\mu_k + a_k}
\end{equation}

\begin{equation}
+ \sum_{u,t} P(Z_{u,t} = k|Y_{u,1:T}, \Theta^{old}) I(N_{u,t} > 0)a_k \cdot \log(\frac{\hat{a}_k}{\mu_k + \hat{a}_k}),
\end{equation}

176
10.10  APPENDIX 2: Proof of Proposition 2

We use same notation as Section 10.3. We aim to prove as \( m \to \infty \),

\[
\sum_{m=1}^{M} W_i^m \hat{\rho}_i^m \to \int \phi_i(x) \rho_i(x) dx. \tag{22}
\]

The normalized weights \( W_i^m = \frac{w_i^m}{\sum_{m=1}^{M} w_i^m} \), where \( w_i^m = f(y_i|z_i, x_i) \). The adjusted auxiliary function \( \hat{\rho}_i^m = \gamma_i s(z_i, \hat{\gamma}_i, y_i) + (1 - \gamma_i) \hat{\rho}_{i-1}^m \). The left side of Eq. (22) can be written as \( \left( \sum_{m=1}^{M} W_i^m \hat{\rho}_i^m \right) / \sum_{m=1}^{M} W_i^m \). We compute the expectation of the numerator \( \sum_{m=1}^{M} W_i^m \hat{\rho}_i^m \) w.r.t random variables \( x_i \) and \( A_i^m \),

\[
E\left[ \frac{1}{M} \sum_{m=1}^{M} W_i^m \hat{\rho}_i^m \right] = \int f(y_i|z_i, A_i)(\gamma_i s(z_{i-1}, \hat{\gamma}_i, y_i) + (1 - \gamma_i) \hat{\rho}_{i-1}^m) P(z_i|z_{i-1}) p(A_i) dx_i dA_i
\]

\[
= \sum_{m=1}^{M} W_{i-1}^m \int f(y_i|x_i)(\gamma_i s(z_{i-1}, \hat{\gamma}_i, y_i) + (1 - \gamma_i) \hat{\rho}_{i-1}^m) P(x_i|x_{i-1}) dx_i
\]

Using the fact that

\[
\phi_{i-1}(x) = \sum_{i=1}^{M} W_{i-1}^j \delta(x_{i-1})
\]

\[
E\left[ \frac{1}{M} \sum_{m=1}^{M} W_i^m \hat{\rho}_i^m \right] \to \int \int (\gamma_i s(z_{i-1}, z_i, y_i) + (1 - \gamma_i) \hat{\rho}_{i-1}) f(y_i|x_i) P(x_i|x_{i-1}) \phi_{i-1}(x) dx_i dx_{i-1}
\]

Similarly, we can derive

\[
E\left[ \frac{1}{M} \sum_{m=1}^{M} W_i^m \hat{\rho}_i^m \right] \to \int \int f(y_i|x_i) P(x_i|x_{i-1}) \phi_{i-1}(x) dx_i dx_{i-1}
\]

Hence, by using Slutsky’s theorem

\[
\sum_{m=1}^{M} W_i^m \hat{\rho}_i^m \to \int \phi_i(x) \rho_i(x) dx, \quad \text{as} \quad m \to \infty. \tag{23}
\]

10.11  APPENDIX 3: The derivation of \( P(u, j) \)

For each user \( u \), we compute \( P(u, j) \) the probability that item \( j \) will be rated at time \( T + 1 \)

\[
P(u, j) = \sum_{k=1}^{K} P(Z_{u,T+1} = k|Y_{u,1:T}) \sum_{N_{u,T+1}=0}^{\infty} P(N_{u,T+1}|\mu_k, \alpha_k) P(I_{u,T+1,j} > 0|N_{u,T+1}, \tau_k).
\]
We first consider the term

\[ P(I_{u,T+1,j} > 0 | N_{u,T+1}, \tau_k) = 1 - P(I_{u,T+1,j} = 0 | N_{u,T+1}, \tau_k) = 1 - (1 - \tau_{k,j})^{N_{u,T+1}}. \]

Then

\[
\sum_{N_{u,T+1}=0}^{\infty} P(N_{u,T+1}|\mu_k, a_k) P(I_{u,T+1,j} > 0 | N_{u,T+1}, \tau_k)
= 1 - \sum_{N_{u,T+1}=0}^{\infty} P(N_{u,T+1}|\mu_k, a_k)(1 - \tau_{k,j})^{N_{u,T+1}}.
\]

By using the moment generating function for ZINB distribution, the second term in the above equation can be computed as

\[
\sum_{N_{u,T+1}=0}^{\infty} P(N_{u,T+1}|\mu_k, a_k)(1 - \tau_{k,j})^{N_{u,T+1}}
= \sum_{N_{u,T+1}=0}^{\infty} P(N_{u,T+1}|\mu_k, a_k) \exp[N_{u,T+1} \log(1 - \tau_{k,j})]
= \sigma_k + (1 - \sigma_k)(1 + \mu_k \tau_{k,j}/a_k)^{-a_k}.
\]

The probability that user \( u \) at time \( T + 1 \) will rate item \( i \) is

\[
P(u, j) = 1 - \sum_k P(Z_{u,T} = k) \sigma_k - \sum_k P(Z_{u,T} = k)(1 - \sigma_k)(1 + \mu_k \tau_{k,j}/a_k)^{-a_k}.
\]

### 10.12 Appendix 4: Simulation Study in Section 10.6.2

In the simulation study in Section 6.2, we let \( \sigma = (0.1, 0.15, 0.1, 0.05, 0.1, 0.1, 0.1, 0.1, 0.1) \), \( \mu = c(5, 7, 10, 4, 9, 5, 6, 8) \), and \( a = c(3, 5, 4, 5, 6, 5, 8, 5) \). The items that a user prefers at a specific state \( k \) are generated using a multinomial distribution with the total number of items \( N_{u,j} \) and the parameter \( \tau_k \). Let \( J_k = 10 \) be the number of items that a user prefers in a specific preference state \( k \), which leads to a total number of items \( J = J_k \cdot K \). We assume that a user’s preference to the \( J_k \) items is over the rest \( J_k(K - 1) \) ones when the user prefers the \( k \)-th state. We simulate the multinomial parameter \( \tau_k \) such that the \((k - 1)J_k + 1\)-th element to \((kJ_k)\)-th element has a relatively higher value than the rest, which means that these \( J_k \) items are preferable by users at this time. There are in total 640 parameters \( \tau \) in the multinomial distribution. We simulate 100 data sets in total.

We compare the parameter estimates and the performance in prediction using our proposed algorithm and the batch EM algorithm. The parameter estimates are listed in Figure 10.7 - 10.13. For parameter \( a \), the bias and variance of the estimates provided by SMC-EM are larger than those provided by the batch EM. For parameter \( \sigma, \mu \) and \( P \), the bias of the estimates provided by the two
algorithms are similar and small, the variance of the estimates of online SMC-EM are larger than those from the Batch EM.
Figure 10.7: Comparison of the NB dispersion parameter $a$ for online SMC-EM and Batch EM.
Figure 10.8: Comparison of the NB mean parameter $\sigma$ for online SMC-EM and Batch EM.
Figure 10.9: Comparison of the NB mean parameter $\mu$ for online SMC-EM and Batch EM.
Figure 10.10: Comparison of the transition probability $P$ for online SMC-EM and Batch EM.
Figure 10.11: Comparison of the transition probability $P$ for online SMC-EM and Batch EM.

Figure 10.12: Comparison of the transition probability $P$ for online SMC-EM and Batch EM.
Figure 10.13: Comparison of the transition probability $P$ for online SMC-EM and Batch EM.
Chapter 11

Discussion

In this thesis, we have presented novel Monte Carlo methods in various applications. Our first application area is phylogenetic reconstruction. We proposed two classes of sequential Monte Carlo algorithms for Bayesian phylogenetics. The first proposed method is an annealed SMC algorithm with adaptive determination of annealing parameters and dynamic resampling. The annealed SMC algorithm provides a general framework for phylogenetic tree inference, which considers the same state space for all the intermediate distributions. Unlike previous SMC method in phylogenetics, many conventional Metropolis-Hastings tree moves in existing literature can be utilized as the SMC proposal distributions. In addition, annealed SMC can provide an unbiased estimate of the marginal likelihood as a by-product of algorithm, which is a challenging task in Bayesian phylogenetics.

The second SMC approach we proposed is a combinatorial sequential Monte Carlo method with a revert-merge proposal. Unlike existing combinatorial SMC, we first use a revert step to find the parent of the current step, then in each merge step we randomly choose a pair of trees to combine. This flexible proposal can benefit the exploration of tree posterior distribution. We explored to combine revert-merge CSMC with Gibbs sampler to estimate the evolutionary parameter. Several particle Gibbs variates are also explored. Our numerical experiments demonstrated our novel CSMC can outperform existing methods.

Further, we extend our phylogenetic reconstruction technique to genome-wide association studies. We proposed a tree-based linear mixed model for genome-wide association studies, in which the genetic similarity matrix is induced by a phylogeny in a hierarchical nonparametric Bayesian framework. We jointly infer the LMM parameters and phylogeny via SMC methods. Our numerical experiments demonstrate the consistency of our methods, and improved false positive rates over traditional univariate regression and linear mixed models. Our application involves a genome-wide association study on multidrug-resistant TB.

Our next application involves parameter estimation in nonlinear differential equations. We propose an adaptive semi-parametric Bayesian framework to solve DEs. Our method avoids expensive numerical approximation and expensive tuning for global smoothing parameters. We also developed a sequential Monte Carlo algorithm in annealing framework to explore the multi-modal posterior.
surface of DEs parameters. Our numerical examples involve both nonlinear ordinary differential equations and nonlinear delay differential equations.

We also investigated two projects related to classical Bayesian computing problems. In the first project, we investigated misclassification of categorical data by including subject specific cohorts with both fallible and infallible classifiers. We also propose an alternative definition of nonidentifiability for hierarchical models. In the second project, we introduce new importance sampling algorithms that are adaptive and rely on the restricted skew-normal and restricted skew-Student families of distributions. We demonstrated the effectiveness of proposed approach in several challenging examples.

In our last project, we have proposed an efficient online SMC-EM algorithm to infer the hidden states and estimate the static parameters of an HMM in which the emission distribution does not belong to an exponential family and the parameter space is large. The computational complexity of the batch EM algorithm increases linearly with the length of the sequence. In contrast, our proposed algorithm only requires storing the auxiliary variable that is computed in a recursive manner. Consequently, the online SMC-EM can save computational resources for long sequence data.
Bibliography


Drummond, A. and M. Suchard (2010). Bayesian random local clocks, or one rate to rule them all. *BMC Biology* 8(1), 114.


192


Lindley, D. V. (1972). Bayesian statistics, a review, Volume 2. SIAM.


Olsson, J. and J. Westerborn (2015). An efficient particle-based online EM algorithm for general state-space models** this work is supported by the swedish research council, grant 2011-5577. *IFAC-PapersOnLine* 48(28), 963–968.


