Sequentially Adaptive Bayesian Leaning Algorithms for Inference and Optimization[*]

Garland Durham and John Geweke[†]

October, 2015

Abstract

The sequentially adaptive Bayesian leaning algorithm is an extension and combination of sequential particle filters for a static target and simulated annealing. A key distinction between SABL and these approaches is that the introduction of information in SABL is *adaptive* and *controlled*, with control guaranteeing that the algorithm performs reliably and efficiently in a wide variety of settings without any specific further attention. This avoids the need for tedious tuning, tinkering, trial and error. The algorithm is pleasingly parallel and when executed using one or more graphics processing units is much faster than competing algorithms, many of which are not pleasingly parallel and unable to exploit the massively parallel architecture of GPUs. This paper describes the algorithm, provides theoretical foundations more self-contained than those in the existing literature, provides applications to Bayesian inference and optimization problems illustrating many advantages of the algorithm, and briefly describes the nonproprietary SABL software.

[*]Incomplete version prepared for European Seminar on Bayesian Econometrics 2015 Conference, October 29-30, Study Center Gerzensee. Please do not cite without permission.

[†]Durham, California Polytechnic Institute, `gbdurham@calpoly.edu`; Geweke, Univeristy of Technology Sydney and Amazon, `John.Geweke@uts.edu.au`.The Australian Research Council provided financial support through grant DP130103356 and through the ARC Centre of Excellence for Mathematical and Statistical Frontiers of Big Data, Big Models, New Insights, grant CD140100049. The Australian Research Council provided financial support through grant DP130103356 and through the ARC Centre of Excellence for Mathematical and Statistical Frontiers of Big Data, Big Models, New Insights, grant CD140100049. The team resonsible for the SABL software consists of John Geweke, Distinguished Professor; Huaxin Xu, Senior Scientific Programmer; Bin Peng, Postdoctoral Research Fellow; and Simon Yin, Scientific Programmer, all at the UTS School of Business. We acknowledge useful discussions with William C. McCausland and Bart Frischkecht in the course of developing SABL.

Contents

1 Introduction

Sequentially adaptive Bayesian learning is an algorithm that constructs a finite sequence of distributions converging to a stated posterior distribution. At each step the approximation is constructed based on what has already been learned in the previous step in such a way that the next iteration will be able to do the same: thus the approximations adapt to the specifics of what has already been learned. The procedure is Bayesian because at each step in the sequence it conditions on what is known at the end of the last one to assure that the current step will be effective. SABL addresses optimization by recasting it as a sequence of Bayesian inference problems.

This global strategy has been used in some approaches to Bayesian inference and to optimization over the past three decades. For Bayesian inference it has much in common with sequential particle filters for a static target (e.g. Chopin 2002). For optimization it has much in common with simulated annealing (Kirkpatrick et al., 1983; Goffe et al., 1994). A key distinction between SABL and these approaches is that in SABL the introduction of information is *adaptive* and *controlled*, guaranteeing that the algorithm performs reliably and efficiently in a wide variety of settings without any specific further attention. This avoids the need for tedious tuning, tinkering, trial and error familiar to anyone who has tried to use sequential particle filters or simulated annealing in a real application (as opposed to a textbook problem). SABL fully and automatically addresses issues like the particle depletion problem in sequential particle filters and the temperature reduction problem in simulated annealing. This paper states the procedures involved, derives their properties, and demonstrates their effectiveness.

A further innovation in SABL is that it fully exploits the fact that most of the algorithm is embarrassingly parallel. The observation that sequential particle filters have this property is commonplace, but exploitation of the property is quite another matter, requiring that a host of important technical issues be addressed, and so far as we are aware there is no other implementation of the sequential particle filter that does so as effectively as SABL. Importance sampling is also embarrassingly parallel and this is trivial to exploit, but there is no adaptation. Similarly some MCMC procedures, like the Metropolis random walk, are embarrassingly parallel and this has been exploited in some recent work but again there is no adaptation; other MCMC procedures, like Gibbs sampling, are not parallel. Not coincidentally importance sampling and the Metropolis random walk are the computationally intensive part of SABL and account for its embarrassingly parallel properties. The remaining portions account for a trivial proportion of computing time.

The pleasingly parallel nature of the SABL algorithm means that it is well positioned for computing environments with multiple processors – especially the inexpensive massively parallel platforms provided by GPU installations available through most universities and clouds like Amazon EC2. Since these environments have supplanted the single processor as the venue for increasing computational efficiency and will continue to do so for at least another decade, pleasingly parallel algorithms like SABL will become increasingly important components of the technical infrastructure in statistics, econometrics and economics. Scientific computing with GPUs became possible with the CUDA and OpenCL extensions of C, but the learning curve for most economists and statisticians is still very steep, witnessed in a number of published papers and failed attempts in large institutions like central banks. All of the important technical difficulties are solved in `SABL`[1], with the consequence that experienced researchers and programmers – for example, anyone who has contributed an R module – can overcome what barriers remain in about a day. `SABL` also provides substantial leverage in CPU environments with dozens of cores, which are becoming more commonplace in linux clusters as Moore's law has ceased to apply to single processors.

Section 2 provides an outline of the SABL algorithm sufficient for the rest of the paper and refers to more complete and detailed discussions. It also develops the controlled introduction of information that is unique to SABL and key to its robustness for a wide variety of models and in many different applications Section 3 develops the foundations of SABL in probability theory. Section 4 provides several examples that illustrate different aspects of Bayesian inference with SABL. Section 5 develops the features and properties of SABL specific to optimization, providing solutions of impediments that have long compromised the application of simulated annealing, a related approach, and demonstrating an intimate relationship to maximum likelihood theory. Section 6 demonstrates the relevance of this theory and the attendant advantages in several illustrations. Section 7 provides a quick introduction to the `SABL` Matlab toolbox, which incorporates numerous advantages for research and application beyond those developed elsewhere in the paper. Proofs of propositions appear in Section 8.

2 The SABL algorithm for Bayesian inference

Fundamental to SABL is the vector $\theta \in \Theta \subseteq \mathbb{R}^k$, which is a parameter vector in Bayesian inference and the maximand in optimization. All functions in this paper should be presumed Riemann integrable on Θ. While for many purposes this could be weakened to Lebesgue integrability, we use that condition here because (1) it is rarely violated in applications and practical problems and (2) it is required in work that relies on digital representation of functions. In what follows all functions should be presumed Riemann integrable. A function $f(\theta) : \Theta \to \mathbb{R}$ is a *kernel* if $f(\theta) \geq 0 \ \forall \ \theta \in \Theta$ and $\int_\Theta f(\theta)\, d\theta < \infty$. The *initial kernel* is $p^{(0)}(\theta)$, $k(\theta)$ is the *information kernel*, $k^*(\theta) = p^{(0)}(\theta)\, k(\theta)$ is the *target kernel*, and $g(\theta)$:$\Theta \to \mathbb{R}$ is the *function of interest*. It is implicit that $\int_\Theta p^{(0)}(\theta)\, k(\theta)\, d\theta < \infty$ and we further require that $\int_\Theta p^{(0)}(\theta)\, k(\theta)\, |g(\theta)|\, d\theta < \infty$. A leading problem addressed by SABL is to approximate

$$\overline{g} = \int_\Theta g(\theta)\, p^{(0)}(\theta)\, k(\theta)\, d\theta \Big/ \int_\Theta p^{(0)}(\theta)\, k(\theta)\, d\theta$$

[1]Throughout, `SABL` refers specifically to the software described briefly in Section 7, whereas SABL refers to the algorithm.

and to assess its accuracy by means of an associated numerical standard error and central limit theorem. It goes without saying that this should be done in a computationally efficient manner and with little need for problem-specific tedious tuning, tinkering, trial and error relative to alternative approaches to the problem.

This forms a common environment for addressing

1. Problems of Bayesian inference: $p^{(0)}$ is the kernel of the proper prior density; $k(\theta)$ is (proportional to) the likelihood function $p(y \mid \theta)$ for stated density p and observed outcomes y; $k^*(\theta)$ is the posterior kernel; and the problem is to approximate the posterior moment $E(g(\theta) \mid y) = \overline{g}$. Sections 2 through 4 take up matters specific to Bayesian inference.

2. Problems of optimization: For any objective function $h(\theta) \leq \overline{h}$, let $k(\theta) = \exp[rh(\theta)]$, the Boltzman transformation,[2] where r is large; $p^{(0)}$ is the kernel of the instrumental distribution; and let $g(\theta) = \theta_i$. If h has a unique mode θ^* then $\overline{g} \rightarrow \theta_i^*$ as $r \rightarrow \infty$ given reasonable side conditions. Sections 5 and 6 address matters specific to optimization.

This section as well as Section 3 also deals with matters specific to Bayesian inference in a manner that provides a starting point for the treatment of optimization in Section 5.

2.1 Overview of the SABL algorithm and computing environment

When used for Bayesian inference SABL is a posterior simulator. Like all posterior simulators, including importance sampling and Markov chain Monte Carlo, these algorithms represent the posterior distribution by means of a random distribution of arguments θ_i that approximate the posterior distribution and can represent it arbitrarily well. Mature posterior simulation theory, like that for importance sampling (IS) and Markov chain Monte Carlo (MCMC), characterizes precisely the approximation error and the ways in which it can be made arbitrarily small. The focus is, first, on conditions under which the sequence $\{\theta_i\}$ is ergodic and, a close second, conditions under which $\{g(\theta_i)\}$ has a limiting normal distribution with mean $\overline{g}$.

SABL may be regarded as another member of a family of algorithms discussed in the statistics literature variously known as sequential Monte Carlo (filters, samplers), recursive Monte Carlo filters, or (sequential) particle filters, and we refer to these collectively as sequential Monte Carlo (SMC) algorithms. The literature is quite extensive with dozens of major contributors; Creal (2012) is a useful survey, and for references most closely related to SABL see Durham and Geweke (2015). Most SMC algorithms use ideas and techniques from IS and MCMC but none is a special case. For historical

[2]This term, as well as the term instrumental distribution, are used in the simulated annealing literature.

reasons the random arguments θ_i are known as *particles.* The mode of convergence in SMC algorithms is the number of particles.

For reasons explained shortly the particles in SABL are doubly-indexed,

$$\theta_{jn}\,(j = 1, \ldots, J; n = 1, \ldots, N)$$

and the mode of convergence is in N. As in all SMC algorithms the particles evolve over the course of the algorithm through a sequence of L transformations. Index the transformations by ℓ and denote the particles in cycle ℓ by $\theta_{ij}^{(\ell)}$. As in all SMC algorithms the particles $\theta_{ij}^{(\ell)}$ are drawn independently from the initial distribution. The transformation in cycle ℓ targets a distribution with kernel density $k^{(\ell)}(\theta)$, with $k^{(L)}(\theta) = k^{*}(\theta)$ so that the final particles represent the posterior distribution. Each cycle ℓ corresponds to a change in information because $k^{(\ell)}$ changes, and in all SMC variants (for compelling theoretical and practical reasons) this change can be characterized as the introduction of additional information.

Variants in SMC algorithms may be distinguished primarily in two broad dimensions: (a) the selection of the intermediate kernels $k^{(\ell)}$, and (b) the methods by which particles evolve from a distribution with kernel density $k^{(\ell-1)}$ to one with kernel density $k^{(\ell)}$. For the most part the SMC literature has taken (a) for granted and concentrated on (b).

With respect to dimension (a), the SMC literature has addressed inference but not optimization, and for a sample $y_1, \ldots, y_T$ has taken $k^{(\ell)}(\theta) \propto p\left(y_1, \ldots, y_{(\ell)} \mid \theta\right)$ with $L = T$. But a common variant is to take $k^{(\ell)}(\theta)\, p\left(y_1, \ldots, y_{(t_\ell)} \mid \theta\right)$, t_ℓ monotone increasing with $t_L = T$, for reasons of computational efficiency. The sequence $\{t_\ell\}$ can be specified at the outset. Alternatively, the SMC algorithm can use the distribution of the particles to determine the value of t_ℓ early in cycle ℓ. This is a simple example of an adaptive algorithm, and it exerts partial control over the introduction of information. But it does not exert full control, because in a time series context some observations introduce a great deal more information than others, and this can lead to substantial inefficiency in SMC algorithms, so much so that the entire enterprise is compromised. (This becomes abundantly clear for crises like market collapses or the great recession of 2008 - 2009.) Applications of SMC outside time series contexts are less common, and we are not aware of any SMC literature dealing with the problem of ordering the observations. (It can make a great deal of difference.) Section 2.1 shows how SABL exerts full control over the introduction of information in each cycle.

With respect to dimension (b), the developed theoretical foundations of SMC have generally ignored adaptation entirely, assuming that there is no feedback from particles to choice of the sequence $\{t_\ell\}$ and to the mutation that is critical to avoid the particle depletion problem. A notable exception is Del Moral et al. (2012), which incorporates the common use of effective sample size as the basis for choosing $\{t_\ell\}$. This is not a simple oversight: everyone who as even tried to implement a SMC algorithm is quite aware that effective adaptation is critical to success, and that if one adheres strictly to the inflexibility in the mathematical foundations of SMC then practical work is precluded. The problem is that the challenges to development of theory are daunting for any one stated method of adaptation: feedback from the particles to the parameters of the algorithm raises complications that are similar to but an order of magnitude more complex than those that arise from the fact that adaptive Markov chain Monte Carlo implies that the algorithm is no longer Markov.

The global outline of SABL is as follows. The innovations in SABL that do not appear elsewhere are in italics.

- Represent initial information by $\theta_{ij}^{(0)} \overset{iid}{\sim} p^{(0)}(\theta)$
- For a sequence of cycles $\ell = 1, 2, \ldots$
  - Correction (C) phase: *Determine* $k^{(\ell)}(\theta)$ and find weights for each of the particles $\theta_{ij}^{(\ell-1)}$ that correctly represent $k^{(\ell)}$.
  - Selection (S) phase: Use multinomial or residual resampling to remove the weights, resulting in unweighted particles $\theta_{ij}^{(\ell,0)}$ that represent $k^{(\ell)}$.
  - Mutation (M) phase: *Use the particles to construct a variance matrix for a Metropolis random walk*, and then execute steps in the random walk *rescaling the variance to maintain a target acceptance ratio in each step. Conclude the steps when the particles are sufficiently independent.*
- If $k^{(\ell)}(\theta) = k(\theta)$ then $L = \ell$ and the algorithm terminates.

The S phase is well-established in the SMC literature. Residual resampling is more efficient than multinomial resampling, and the alternatives of systematic and stratified resampling do not lead to the central limit theorems critical to the mathematical foundations of SMC. Residual resampling is the default choice in SABL. Resampling is inherently nonparallel, and there seems to be an impression in some quarters that this fatally compromises any attempt to exploit the embarrassingly parallel nature of the other phases. That impression is incorrect: for all but very small problems where execution is fast without exploiting the embarrassingly parallel characteristics of the C and M phases anyway, the S phase accounts for a trivial portion of execution time.

The following three sections address the C, S and M phases, respectively, concentrating primarily on the innovations in SABL but also discussing their embarrassingly parallel character. Greater detail can be found in Durham and Geweke (2015) and in the *SABL Handbook*.[3]

2.2 The correction (C) phase

The correction phase determines the kernel $k^{(\ell)}$ and reweights the particles from cycle $\ell - 1$ to reflect the updated kernel using standard importance sampling. The weight

[3]URL

function is $w^{(\ell)}(\theta) = k^{(\ell)}(\theta)/k^{(\ell-1)}(\theta)$, applied to each particle. Thus the primary computation task in the C phase is

$$w^{(\ell)}(\theta_{ij}) = k^{(\ell)}(\theta_{ij})/k^{(\ell-1)}(\theta_{ij}) \quad (j = 1, \dots J; n = 1, \dots, N).$$

For likelihood functions that can be evaluated in essentially closed form this task is embarrassingly parallel in the particles θ_{ij}, of which there are $2^{14} = 16,384$ by default in `SABL` ($j = 2^4, n = 2^{10}$). When `SABL` executes using one or more GPUs each particle is a thread – that is, each particle corresponds to a single core of the GPU. The underlying CUDA code declares the number of threads and coding proceeds for a virtual GPU with as many cores as desired ($J \cdot N$, in our case; things are not quite this simple but the complications are transparent to the `SABL` user). The CUDA compiler handles the assignment to physical cores in a highly efficient manner through a cascade of caches.

If the information introduced in each cycle is the one next observation, that is the end of the C phase. If one or more observations are being introduced, then the widespread practice in SMC is, beginning with $w_0^{(\ell)}(\theta) = 1$ and $j = 1$, is to compute the candidate weight function

$$w_j^{(\ell)}(\theta) = w_{j-1}^{(\ell)}(\theta) \cdot p(y_{t_\ell+j} \mid y_1, \dots, y_{t_\ell+j-1}, \theta)$$

and then evaluate the effective sample size

$$ESS = \left[\sum_{j=1}^{J}\sum_{n=1}^{N} w^{(\ell)}\left(\theta_{ij}^{(\ell-1)}\right)\right]^2 / \sum_{j=1}^{J}\sum_{n=1}^{N} w^{(\ell)}\left(\theta_{ij}^{(\ell-1)}\right)^2$$

developed by Liu and Chen (1998) and other contributors to the SMC literature. (Observe that if all weights were identical then $ESS = JN$ whereas if only one is positive then $ESS = 1$.) A closely related score is relative effective sample size $RESS = ESS/JN$. The addition of observations, $j = 1, 2, \dots$ continues until $RESS$ first drops below a target value $RESS^*$. Del Moral et al (2012) extends the existing probability-theoretic foundations of SMC to include this case. Experience suggests that execution time does not vary much for $RESS^* \in [0.1, 0.9]$: higher (lower) values imply more (fewer) cycles but fewer (more) iterations in the M phase. The computation of ESS is not parallel because it involves all particles and is therefore not particle specific. But these computations are trivial compared with likelihood function evaluation and Matlab harvests the $w(\theta_{ij})$ from the respective threads efficiently.

This introduction of information by moving through the sample has been termed data tempering. The literature also mentions power tempering (e.g. Duan and Fulop, 2015) but it is not widely used. In power tempering information is introduced by means of a monotone increasing sequence r_ℓ with $r_1 > 0$ and $r_L = 1$. Indeed one can simply specify such a sequence like $r_\ell = \ell/10$, all the underlying theory applies, and the result will be disastrous except perhaps for an isolated case or two. (Linear increase is much too fast at the beginning, much too slow at the end, and particle depletion will be nearly complete after one or two cycles.)

The relative effective sample size for power tempering in cycle ℓ is

$$RESS = f\left(r_\ell\right) = \frac{\sum_{j=1}^{J}\sum_{n=1}^{N}\left[k\left(\theta_{ij}^{(\ell-1)}\right)^{r_\ell - r_{\ell-1}}\right]^2}{JN\sum_{j=1}^{J}\sum_{n=1}^{N}k\left(\theta_{ij}^{(\ell-1)}\right)^{2(r_\ell - r_{\ell-1})}}. \tag{1}$$

This expression merely evaluates the quality of the weight function in cycle ℓ implied by the power increment $r_\ell - r_{\ell-1}$. But this criterion can also be used to introduce information – that is, to choose r_ℓ given $r_{\ell-1}$ – in completely controlled fashion. The foundation for this approach is the following result.

Proposition 1 *Solving for power increment. The function $f\left(r_\ell\right)$ in (1) is monotone decreasing for $r_\ell \in [r_{\ell-1}, \infty)$. Let m be the number of (i,j) for which $k\left(\theta_{ij}^{(\ell-1)}\right) = \max_{i,j} k\left(\theta_{ij}^{(\ell-1)}\right)$. If $RESS^* \in (m/JN, 1)$ then $f\left(r_\ell\right) = RESS^*$ has exactly one solution $r_\ell \in (r_{\ell-1}, \infty)$; otherwise it has none.*

Proof. Section 8 ■

Proposition 1 implies that $f\left(r_\ell\right) = RESS^*$ can be solved to machine accuracy using successive bifurcation in 52 or fewer iterations (52 being the number of mantissa bits in standard 64-bit floating point arithmetic). The computation time is negligible. The result is that the target $RESS^*$ is met exactly in each cycle – whereas data tempering can (and often does) end with a final observation t_ℓ for which $RESS$ falls substantially below target; and, especially in larger models and/or more diffuse priors, $RESS$ can be impractically small for the early observations. (These problems with data tempering become evident in a very unbalanced weight function, very few unique particles – perhaps only one – following the S phase, and a great many iterations of the Metropolis random walk algorithm in the M phase to recover.)

Power tempering is the default variant of the C phase in `SABL`, and the default value of $RESS^*$ is 0.5.

2.3 The selection (S) phase

The S phase is well-established in the SMC literature. Residual resampling is more efficient than multinomial resampling, and the alternatives of systematic and stratified resampling do not lead to the central limit theorems critical to the mathematical foundations of SMC. `SABL` uses residual resampling by default. Resampling is inherently nonparallel, and there seems to be an impression in some quarters that this fatally compromises any attempt to exploit the embarrassingly parallel nature of the other phases. The second observation is incorrect: for all but very small problems where execution is fast without exploiting these characteristics of the C and M phases anyway, the S phase accounts for a trivial portion of execution time even after fully exploiting the embarrassingly parallel structure of the C and M phases.

In SABL the particles are organized into J groups of N particles each. The selection phase proceeds independently in each group: that is, particles are resampled from within and never across groups. In a non-adaptive SMC context this renders the particle groups independent. We then have the independent approximations

$$\overline{g_j} = N^{-1} \sum_{n=1}^{N} g\left(\theta_{jn}^{(\ell)}\right) \quad (j = 1, \ldots, J)$$

and these can be used in the obvious way to assess the accuracy of the grand approximation $\overline{g} = J^{-1} \sum_{j=1}^{J} g_j$; details are in Durham and Geweke (2015) and the *SABL Handbook*. The result is a numerical variance and standard error for the SABL approximation of any posterior moment. Posterior variance is simply the sample variance of $g\left(\theta_{ij}\right)$ across all groups and particles, and relative numerical efficiency (Geweke, 1989) is then the ratio of posterior variance to numerical variance. Numerical accuracy is also implicit in the central limit theorems of the SMC literature, but we are unaware of any practical utilization of these abstract expressions.

Of course, the adaptive nature of SABL creates dependence across groups, and so the foregoing reasoning does not apply. This is but one consequence of a much deeper problem with the existing probability theoretic foundations of SMC for the adaptive variants of SMC that are essential to practical application. Section 3.1 returns to the way SABL removes this problem.

2.4 The mutation (M) phase

At the conclusion of the S phase the particles are dependent due to the fact that some are repeated while others have been eliminated. If the algorithm had no mutation phase then the number of distinct particles would be weakly monotonically decreasing. In the application of SMC to Bayesian inference it is easy to verify that for any given number of particles and under mild regularity conditions for the likelihood function the number of distinct particles would diminish to one as sample size increases. It is widely recognized that the M phase is critical to any practical application of SMC algorithms and, furthermore, that the M phase must be adaptive – in particular it must use the distribution of particles to derive an effective variance matrix for the Metropolis random walk. (A similar situation can arise in MCMC with Metropolis-Hastings algorithms.) A flexible adaptive scheme is essential to effective application of SMC that obviates the need for tedious tuning, tinkering, trial and error. Moreover, computational intensity tends to concentrate more and more in the M phase as the scale and complexity of the problem increases. An efficient M phase is essential to practical SMC algorithms.

The M phase is a Metropolis random walk. In each step s of the random walk the proposal is $N\left(\theta_{ij}, \Sigma^{(\ell,s)}\right)$, where $\Sigma^{(\ell,s)}$ is proportional to the sample variance of $\theta_{jn}^{(\ell,0)}$ computed using all the particles. The factor of proportionality increases when the rate of candidate acceptance in the previous step exceeds a specified threshold and decreases otherwise. This draws on established practice in MCMC and works well in this context.

SABL again incorporates variants of the basic Metropolis Gaussian random walk, as well, drawing on experience in the MCMC literature. In `SABL` the default sets acceptance goal at 0.25, proportionality factors are incremented or decremented by 0.1 with a lower bound of 0.1 and an upper bound of 2.0. The initial value of the proportionality factor is 0.5 at the start of cycle $\ell = 1$ and subsequently factors carry through from one cycle to the next. Any or all of these values can changed. They can be made model-specific, in which case they become the default when the model is applied. They can also be changed for a specific application. `SABL` also incorporates a variant of this process in which θ is partitioned and the Metropolis random walk is applied to each partition separately, and there are a number of variants of fixed and random partitions.

The M phase should terminate when the dependence among particles following the S phase has been sufficiently broken – a condition often referred to as "sufficient mixing". This is important to efficient and effective implementation of SMC algorithms but to our knowledge has not been discussed in the literature. SABL accomplishes this by tracking the relative numerical efficiency of some simple, model-specific functions of the particles at the end of each step s of the Metropolis random walk. Typically relative numerical efficiency increases, with some noise, as the steps proceed. The M phase and the cycle terminates when the average RNE across the tracking functions reaches a specified target. `SABL` uses one target for all cycles except the last (default value 0.4), which has its own target (default value, 0.9). The higher target in the final cycle results in more iterations of the M phase in that cycle, but a value close to 1 results in particles that are weakly correlated at the end of the algorithm. Default values can again be made model or application specific. The M phase also terminates if the number of cycles becomes excessive without attaining the target relative numerical efficiency; details are in the *SABL Handbook*.

3 Foundations of the SABL algorithm

Material and subsections to be inserted.

3.1 The two-pass algorithm

As described in the previous Section the SABL algorithm builds the kernels $k^{(\ell)}$ in the C phase of successive cycles using information in the particles $\theta_{jn}^{(\ell-1)}$ and the Metropolis random walk in the M phase using information in the succession of particles $\theta_{jn}^{(\ell,s-1)}$ $(s = 1, 2, \ldots)$. These critical adaptations, especially in the M phase, are precluded by the theorems for sequential Monte Carlo, a key underpinning of SABL. Indeed, all practical implementations of sequential Monte Carlo are subject to this limitation.

SABL resolves this problem and achieves analytical integrity by utilizing two passes. The first pass is the adaptive variant in which the algorithm is self-modifying. The second pass fixes the design emerging from the first pass and then executes the fixed Bayesian learning algorithm to which the theorems in the sequential Monte Carlo literature directly apply. Experience strongly suggests that the second pass leads to qualitatively the same results as the first pass – in particular, to similar simulated posterior moments and associated numerical standard errors. Differences in posterior moment approximations, between the two passes, are typically consistent with numerical standard errors. Yet this is simply an observation, and it is prudent to check this condition occasionally in the course of a research project.

4 Bayesian inference with SABL

This section takes up some selected applications of Bayesian inference with the objective of illustrating different aspects of the SABL algorithm and its execution using the `SABL` toolbox (Section 7).

4.1 Example: The EGARCH model of asset returns

This example uses a model from Durham and Geweke (2015) with updated data to illustrate the facility of SABL in model comparison and document execution time on several GPU and CPU platforms. The data are the log daily returns y_t for the S&P 500 index beginning January 3, 1990 and ending October 9, 2015, a total of 6493 observations. The EGARCH model for a sequence of asset returns combines a mixture model for volatility $v_{k,t}$ (volatility in state k on day t) with a normal mixture model for the shocks to asset returns ε_t. The model for volatility is

$$v_{kt} = \alpha_k v_{k,t-1} + \beta_k \left(|\varepsilon_{t-1}| - (2/\pi)^{1/2} \right) + \gamma_k \varepsilon_{t-1} \quad (k = 1, \ldots, K).$$

where

$$\alpha_k > 0,\ \beta_k > 0,\ \gamma_k > 0\ \ (k = 1, \ldots, K). \tag{2}$$

Conditional on volatility the model for the asset return is

$$y_t = \mu_Y + \sigma_Y \exp\left(\sum_{k=1}^{K} v_{kt}/2 \right) \varepsilon_t.$$

The normal mixture model for the shock to asset returns is

$$p(\varepsilon_t) = \sum_{i=1}^{I} p_i \phi\left(\varepsilon_t; \mu_i, \sigma_i^2\right)$$

where $\phi(x; \mu, \sigma^2)$ is the Gaussian density with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 > 0$ evaluated at x and

$$p_i > 0\,(i = 1, \ldots, I) \tag{3}$$

with

$$\sum_{i=1}^{I} p_i = 1. \tag{4}$$

This model is included in SABL.

The model as stated is not identified in the classical sense that a continuum of alternative parameter values implies exactly the same p.d.f. The problems are (1) μ_Y can be replaced by $\mu_Y - c$ and μ_i by $\mu_i + c$ $(i = 1, \dots, I)$ and (2) γ_k can be replaced by $d\gamma_k$ and β_k by $d\beta_k$ $(k = 1, \dots, K)$ and σ_i^2 by $d^2\sigma_i^2$ $(i = 1, \dots, I)$. This is not a formal problem for Bayesian inference with a proper prior distribution, but if one wishes to interpret the posterior distribution of parameters then formal identification is helpful; moreover, the M phase of the SABL algorithm may take longer without classical identification, so the problem merits interest even if one is solely interested in predictive distributions, which is often the case with asset return models. The normalization

$$\mathrm{E}\left(\varepsilon_t\right) = \sum_{i=1}^{I} p_i \mu_i = 0, \ \ \mathrm{var}\left(\varepsilon_t\right) = \sum_{i=1}^{I} p_i \left(\mu_i^2 + \sigma_i^2\right) = 1 \tag{5}$$

achieves classical identification. The constraints (4) and (5) need to be enforced, which can be accomplished by means of $p_i^* > 0$, μ_i^*, $\sigma_i^* > 0$ $(i1 =, \dots, n)$ followed by

1. $p_i = p_i^* / \sum_{j=1}^{I} p_J^*$ $(i = 1, \dots, J)$
2. $\mu_i^{**} = \mu_i^* - \sum_{j=1}^{I} p_j \mu_j^*$
3. $c = \left\{ \sum_{i=1}^{I} p_i \left[\mu_i^{**2} + \sigma_i^{*2} \right] \right\}^{-1/2}$
4. $\mu_i = c\left(\mu_i - \sum_{j=1}^{J} p_j \mu_j^*\right)$ and $\sigma_i = c\sigma_i^*$ $(i = 1, \dots, J)$

One way to handle the inequality constraints is to begin with the parameter vector $\theta \in \Theta = \mathbb{R}^k$, where $k = 2 + 3(K + I)$, $\theta' = \left(\theta^{(1)\prime}, \dots, \theta^{(8)\prime}\right)$ where $\theta^{(1)}$ and $\theta^{(2)}$ are scalars, $\theta^{(3)}$, $\theta^{(4)}$ and $\theta^{(5)}$ are each $K \times 1$, and $\theta^{(6)}$, $\theta^{(7)}$ and $\theta^{(8)}$ are each $I \times 1$. Then

$$\mu_Y = \theta^{(1)}/1000, \quad \sigma_Y = \exp\left(\theta^{(2)}\right),. \tag{6}$$

and for $k = 1, \dots, K$,

$$\alpha_k = \tanh\left(\theta_k^{(3)}\right), \quad \beta_k = \exp\left(\theta_k^{(4)}\right), \quad \gamma_k = \theta_k^{(5)}, \tag{7}$$

and for $i = 1, \dots, I$,

$$p_i^* = \tanh\left(\theta_i^{(6)}\right) + 1, \quad \mu_i^* = \theta_i^{(7)}, \quad \sigma_i^* = \exp\left(\theta_i^{(8)}\right). \tag{8}$$

The prior distribution is proper with support $\Theta = \mathbb{R}^k$. With the exception of $\theta^{(6)}$ the parameters are classically identified, and the proper prior assures that all of θ is identified in the posterior distribution.

The normalizations just described are not the only ones that might be used, and the support Θ need not be $\mathbb{R}^k$. For example, $(p_1, \ldots, p_I)$ might have a Dirichlet prior distribution, reducing the elements of θ by 1, and $(\sigma_1, \ldots, \sigma_k)$ might have identical independent gamma prior distributions. Both prior distributions are available in `SABL`. Situations like this arise regularly in complex models and two features of `SABL` deal with them.

1. The first feature is a mapping system for parameters. `SABL` distinguishes between the algorithm parameter vector θ and the model-specific parameter vector, which here consists of the parameters μ_Y, σ_Y, $(\alpha_k, \beta_k, \gamma_k)$ $(k = 1, \ldots, K)$ and (p_i, μ_i, σ_i) $(i = 1, \ldots, I)$. `SABL` incorporates a generic parameter mapping structure that transforms θ to the vector of model parameters. Most models[4] in `SABL` have default parameter maps, which for the EGARCH model is the one just described. The user may substitute a different parameter map, which would be done here to incorporate the alternatives described in the previous paragraph. New users need not even be aware of the parameter mapping feature to get started, especially with simple models.

2. The second feature is a system for combining prior distributions to build up a complete proper prior distribution for θ. In the example just provided, the prior distribution has one component, the multivariate normal distribution with scalar variance matrix for θ. In the alternative prior distribution mentioned there would be one component for the Dirichlet prior distribution with parameter vector $\theta^{(6)}$, K components for the gamma distribution for each element of $\theta^{(3)}$ accompanied by the parameter map $\sigma_k = \theta_k^{(3)}$ $(k = 1, \ldots, K)$, and a final component for the remaining elements of θ jointly. Most models have default prior distributions, and that for the EGARCH model is the one just presented.

SABL provides log marginal likelihood as a by-product; for details, see Durham and Geweke (2015), Algorithm 3.3, Section 3.3. This ability stems from the importance sampling in the C phase, and the fact that the average weight in importance sampling provides the marginal likelihood when the source distribution is the prior, the numerator of the weight function is the properly normalized likelihood function and the denominator of the weight function is the properly normalized prior density (Geweke, 2005, Section 8.2.2). The J groups of particles then facilitate the computation of numerical standard errors in the same way that they do for posterior moments.

Using the `SABL` default values of all algorithm parameters, including $J = 16$ groups of $N = 1024$ particles each, Bayesian inference for EGARCH models for all combinations $K \in \{1, \ldots, 4\}$ and $I \in \{1, \ldots, 6\}$ was conducted using `SABL`. For the EGARCH model

[4]Exceptions are very flexible models like the full linear simultaenous equation model, which always incoporates application-specific zero, cross-parameter and/or cross-equation restrictions. In these cases there is no useful default parameter map. The same qualification applies to the default prior distributions discussed in the next paragraph.

with the data described and given $I \in \{1, \ldots, 6\}$ marginal likelihood is greater for $K = 2$ than for any other $K \in \{1, \ldots, 4\}$. Given any $K \in \{1, \ldots, 4\}$ marginal likelihood is greater for $I = 6$ than for any other $I \in \{1, \ldots, 6\}$. Results for $K = 2$ and $I \in \{7, \ldots, 12\}$ yield marginal likelihoods not significantly (in the metric of numerical standard error) different from that for $K = 2$, $I = 6$. At the latter values the log marginal likelihood approximation is 21,430.53 and the accompanying numerical standard error is 0.16. Table 1 provides the difference between this value and log marginal likelihood for all other elements $(K, I) \in \{1, \ldots, 4\} \times \{1, \ldots, 6\}$, together with their numerical standard errors.

Table 1: Difference in log marginal likelihood, EGARCH(2,6) versus alternatives

	$K = 1$	$K = 2$	$K = 3$	$K = 4$
$I = 1$	-146.60 (0.22)	-103.20 (0.34)	-105.03 (0.29)	-104.10 (0.24)
$I = 2$	-48.35 (0.21)	-8.50 (0.22)	-8.85 (0.22)	-10.45 (0.32)
$I = 3$	-43.03 (0.21)	-5.11 (0.23)	-5.48 (0.23)	-6.86 (0.38)
$I = 4$	-39.88 (0.23)	-2.40 (0.20)	-3.62 (0.20)	-4.80 (0.25)
$I = 5$	-38.42 (0.22)	-0.86 (0.25)	-1.43 (0.34)	-3.06 0.36)
$I = 6$	-37.23 (0.29)	0.00	-0.86 (0.24)	-2.08 (0.22)

Except for the choice of GPU processor all algorithm parameters were `SABL` default values. Table 2 provides information about wallclock execution time in seconds. The left part of the panel shows how time varies with the complexity of the model. The relationship is very nearly ($\overline{R}^2 = 0.94$) log linear in I and K, the elasticity with respect to I is 0.37 and with respect to K is 0.76. Total execution time for all 24 models was 3.24 hours and produced a posterior sample of 16,384 nearly uncorrelated particles in each case.

All algorithm parameters were `SABL` default values, except for those governing the hardware configuration and utilization. The results in Table 1 all used a single NVIDIA K80 processor, which was state-of-the-art in late 2014. The left panel of Table 2 shows the wallclock execution time (elapsed time from start to finish) for the 24 combinations of I and K in Table 2. The relationship is very nearly ($\overline{R}^2 = 0.94$) log linear in I and K, the elasticity with respect to I is 0.37 and with respect to K is 0.76. Total execution time for all 24 models was 3.24 hours and produced a posterior sample of 16,384 nearly uncorrelated particles in each case.

The right panel of Table 2 documents wallclock execution time for some alternative hardware configuration and utilization. The configuration in instance (1) is a single GPU on the Amazon EC2 cloud, evidently of an earlier generation – it is about four times slower. In view of the rapid increase in number of cores in successive generations of GPU chips this is not surprising. The other three instances use CPU cores. In this environment all code executed is written in Matlab, using vector arithmetic across particles. Instance (2) is the `SABL` default, which does not require the Matlab Parallel Toolbox, particles are not explicitly allocated across cores; however, the Matlab compiler

Table 2: Wallclock execution time for various EGARCH models, various processors

	NVIDIA K80				AWS* $K=2$, $I=6$		
	$K=1$	$K=2$	$K=3$	$K=4$	Type	Instance	Time
$I=1$	122	249	388	615	1 GPU	(1) g2.2xlarge	1939
$I=2$	195	355	485	612			
$I=3$	239	435	560	687	CPU quadcore	(2) "1-core"	37,711
$I=4$	304	483	614	736	CPU quadcore	(3) 2-core	39,741
$I=5$	333	474	657	740	CPU quadcore	(4) 4-core	19,944
$I=6$	345	536	699	804			

*Amazon Web Services EC2, US East (Virginia) zone

still seeks to take advantage of the presence of four CPU cores when code is compiled. Execution takes 70 times as long as the K80 GPU and almost 20 times as long as the AWS GPU. `SABL` provides the option to explicitly allocate particles to CPU cores, by changing a single default parameter. Declaring two parallel streams (instance 3) leads to slightly slower execution, but declaring four streams reduces execution time by almost half from the default instance (2).

In this example (1) evaluation of the likelihood function is not trivial (though even more complicated cases are common), (2) there are no sufficient statistics, (3) there are thousands of observations and (4) likelihood function evaluation is embarrassingly parallel. This combination of characteristics, entailing well over a billion floating point operations over all 16,384 particles, is the kind that provides the greatest leverage for GPU computation. In less computationally intensive cases, including almost all textbook examples and even those in some published papers demonstrating posterior simulation and numerical optimization, GPU computation provides no real advantage, and indeed can be substantially slower than common CPU platforms due to the overhead entailed in GPU use. (Section 7 provides some more detail.) All of the other examples in this paper are in this latter category, and utilize CPU execution without explicit declaration of parallel streams – computation time for these examples is never more than 5 minutes.

4.2 Secular and cyclical behavior of GDP in an AR(3) model

This example is taken from Geweke (2015) and utilizes the `SABL` normal model in which the outcomes y_t $(t = 1, \ldots, T)$ are mutually independent conditional on (x_t, z_t) $(t = 1, \ldots, T)$:

$$y_t \sim N\left(\beta' x_t,\ \exp\left(\gamma' z_t\right)\right) \tag{9}$$

where x_t and z_t are vectors of covariates and y_t is the corresponding outcome. The model parameter vector is (β, γ). More specifically, in this example there is no heteroscedasticity and the econometric model is a straightforward third-order autoregression

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 y_{t-3} + \varepsilon_t,\ \ \varepsilon_t \overset{iid}{\sim} N\left(0, \sigma^2\right), \tag{10}$$

The entire analysis conditions on the initial values y_{-2}, y_{-1} and y_0 making the likelihood function is a specific case of the likelihood function for (9) with $x_t = (1, y_{t-1}, y_{t-2}, y_{t-3})$ and $z_t = 1$.

The individual components of β have no substantive interpretation and little if any interesting statistical interpretation. Geweke (1988) considered the case where the generating polylnomial $1 - \beta_1 z - \beta_2 z^2 - \beta_3 z^3$ has one real root r_1 and a conjugate pair of complex roots (r_2, r_3), and then transformed

$$\alpha_s = |r_1|^{-1}, \ \alpha_c = |r_2|^{-1}, \ p = 2\pi / \tan^{-1}\left(\mathrm{Im}\,(r_2) \,/\, \mathrm{Re}\,(r_2)\right), \tag{11}$$

with an interpretation motivated by Samuelson (1939): α_s is the secular amplitude, α_c is the cyclical amplitude, and p is the periodicity. In (11) $\tan^{-1}$ is the principle branch with range $[0, \pi)$, and (11) restricts (10) by requiring that the characteristic polynomial $1 - \sum_{j=1}^{3} \beta z^j$ have a complex conjugate pair of roots. The application in Geweke (1988) is to OECD annual real GDP series for quite a few countries for the period 1957 - 1983.

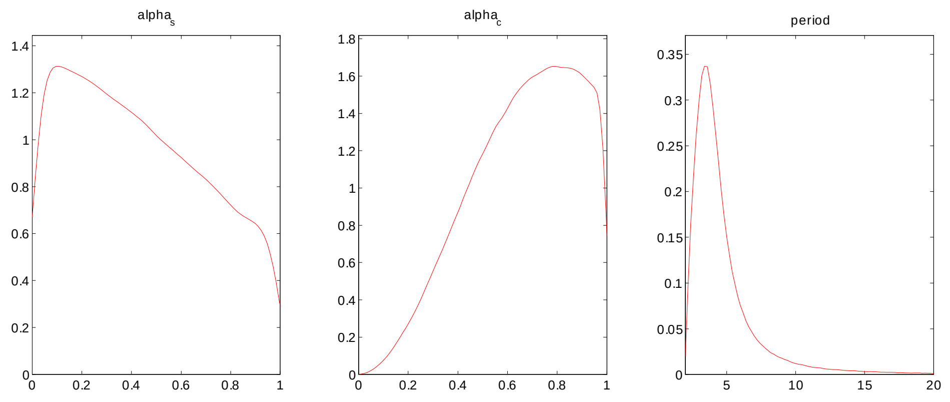


Figure 1: Implicit prior distributions in Geweke (1988)

Geweke (1988) used a conventional improper prior distribution $p(\beta, \sigma) \in \sigma^{-1}$ and reported the posterior distributions of the parameters of interest α_s, α_c, p, σ. Unrecognized in Geweke (1988) is the fact that this prior distribution is informative for β. Figure **1** shows prior densities for α_s and for α_s, each over the range $(0, 1)$ and for p over the range $(2, 20)$, $p = 2$ being the minimum possible. While "uninformative" is not a precisely defined property, few investigators would ascribe that characteristic to these distributions. (Moreover, the prior distribution for period is strikingly similar to the posterior distributions for this parameter reported in Geweke (1988).) The mechanics are well understood: since the transformations are nonlinear, the Jacobian varies with β and σ; of greater substantive importance is the fact that loose characterizations of prior distributions as "uninformative," if made as an escape from formulating a substantive prior distribution, are intellectually indefensible.

As a constructive alternative, restate the model exclusively in terms of parameters that have ready substantive interpretation and then apply a substantive proper prior distribution in each case. The alternative parameterization uses the half-life of the secular component, $h_s = \log(1/2)/\log(\alpha_s)$, the half-life of the cyclical component, $h_c = \log(1/2)/\log(\alpha_c)$, the period p and the shock standard deviation σ. The mapping from (h_s, h_c, p) to β is

$$\begin{aligned}
\beta_1 &= (1/2)^{1/h_s} + 2(1/2)^{1/h_c}\cos(2\pi/p), \\
\beta_2 &= -\left((1/2)^{1/h_s}(1/2)^{1/h_c}\right)\cos(2\pi/p) + \alpha_c^2, \\
\beta_3 &= (1/2)^{1/h_s}(1/2)^{2/h_c}.
\end{aligned}$$

The mapping from the algorithm parameter vector θ to the substantive parameters is

$$\beta_0 = \theta_1, h_s = \exp(\theta_2), h_c = \exp(\theta_3), p = \exp(\theta_4), \sigma = \exp(\theta_5).$$

The prior distribution is

Parameter	Distribution	Centered 90% interval
Intercept β_0 (10):	$\theta_1 = \beta_0 \sim N(10, 5^2)$	$\beta_0 \in (1.77, 18.22)$
Secular half-life h_s:	$\theta_2 = \log(h_s) \sim N(\log(25), 1)$	$h_s \in (5.591, 111.9)$
Cyclical half-life h_c:	$\theta_3 = \log(h_c) \sim N(\log(1), 1)$	$h_c \in (0.2237, 22.36)$
Period p:	$\theta_4 = \log(p) \sim N(\log(5), 1)$, $p > 2$	$p \in (2.316, 28.46)$
Shock σ (10)	$\theta_5 = \log(\sigma) \sim N(\log(0.025), 1)$	$\sigma \in (0.005591, 0.1118)$

Setting up the map from the algorithm parameter vector θ to the model parameter vector (β, σ) is straightforward in `SABL` as described in Section 4.1. The prior distribution in this case has a single component, the multivariate normal prior distribution subject to the linear constraint $\log(p) > \log(2)$ is also straightforward, as noted in Section 4.1, and also utilizing the `SABL` feature that provides for truncation of many prior distributions.

Geweke (2015) uses this model with OECD annual time series 1973 - 2014 (range of outcome variable) for real GDP converted to US dollars by means of purchasing power parity corrected exchange rates for a number of countries. Posterior distributions are strikingly similar across countries. Table 4 provides the posterior moments of $\log(h_s)$, $\log(h_c)$, $\log(p)$ and $\log(\sigma)$ for the United States and Figure **6.2** shows their posterior and prior distributions. (It would have been a simple matter to work with h_s, h_c, p and σ directly. The presentation chosen provides better scaling and also proves convenient in Section 6.2, which provides corresponding maximum likelihood estimates and asymptotic standard errors using the SABL algorithm for optimization.) Several features are notable.

1. Except for σ(for which the prior density is close to the horizontal axis), the prior distribution contributes substantial information to the posterior despite the fact that it is less precise than the subjective priors of many economists would likely be. The prior information is not as great as that in the data, but the difference

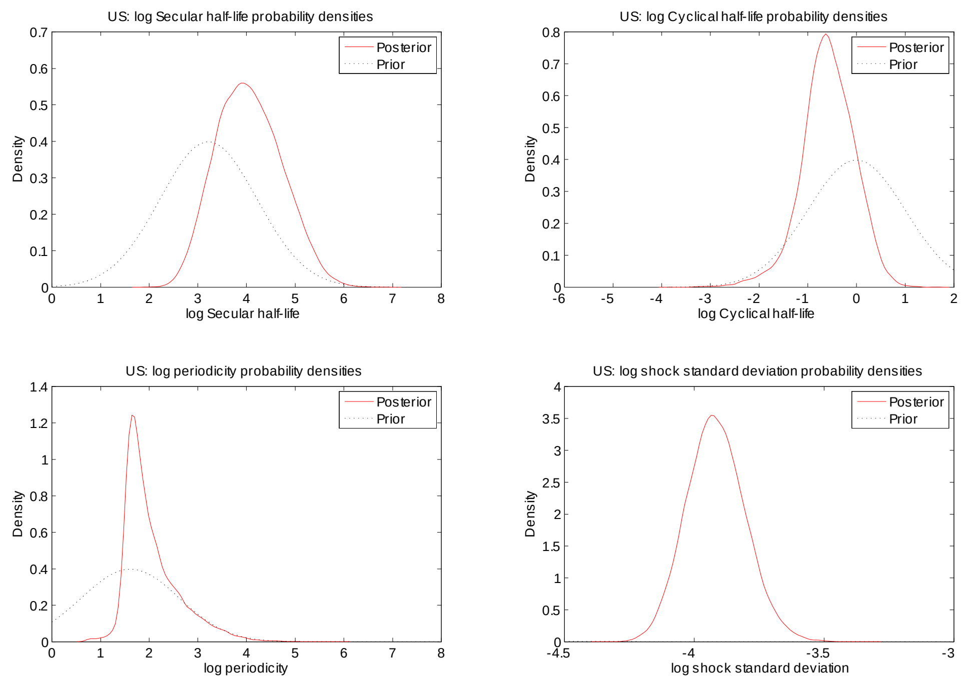


Figure 2: Posterior and prior distributions for four parameters

is small. This is not surprising in view of the fact that h_s is in the range 20 to 150 years, the period is in the range of 4 to 20 years, and there are only 42 years in the sample. On the other hand, under the model specification there are 42 independent observations to inform σ.

2. The four parameters are only weakly correlated in the posterior distribution. This is in marked contrast to the situation for the parameter vector β in (10), where there is large correlation due to multicollinearity in $(y_{t-1}, y_{t-2}, y_{t-3})$. This supports the efficacy of thinking about and working with the chosen parameterization.

3. Except for $\log(\sigma)$ the posterior distributions are more diffuse than the local behavior near the mode implies for a normal distribution. This is quite pronounced in the case of $\log(p)$. This feature, in turn, suggests that the classical asymptotic confidence regions for maximum likelihood estimates of these parameters might be considerably more concentrated than is the posterior distribution. Section 6.2 returns to this comparison.

5 The SABL algorithm for optimization

The SMC literature has recognized that those algorithms can be applied to optimization problems by concentrating what otherwise would be the likelihood function more and more tightly about its mode (Del Moral et al., 2006). Indeed, if the objective function is the likelihood function itself, the result would be a distribution of particles near the likelihood function mode. A straightforward way to do this is to apply power tempering in the C phase, but continue with cycles increasing the power r_ℓ well beyond the value $r_L = 1$ that defines the likelihood function itself. The principle (the inspiration) is sound, but like many principles working out the details and engineering for routine applications (the perspiration) is the greater challenge. To the best of our knowledge this latter agenda has not been pursued in the literature. This section sets for the details and engineering and the result is incorporated in the `SABL` toolbox.

5.1 The optimization problem

This section changes the notation slightly to conform with that of the global optimization literature, and specifically the simulated annealing literature that has taken a similar approach and upon which SABL provides a very substantial improvement. Adopting the notation of much of the global optimization literature the objective function is $h(x) : X \to \mathbb{R}$, x replacing the vector θ and X replacing Θ in the notation of Section 2. We require that $h(x)$ be Riemann integrable on X. Then the associated Boltzman kernel for the optimization problem is

$$k^*\left(x; h, p^{(0)}, r\right) = p^{(0)}(x) \exp\left(r \cdot h(x)\right)$$

and this takes the place of the target kernel $k^*(\theta)$ from Section 2.

The initial density $p^{(0)}(x)$ is a technical device required for SABL as it is for simulated annealing and steepest ascent algorithms, but the final result of these algorithms, if they are successful, does not depend on $p^{(0)}$ whereas the posterior distribution clearly depends on the prior distribution. In the optimization literature $p^{(0)}(x)$ is sometimes called the instrumental distribution. Borel measure is denoted by μ.

Fundamental to the both the details and the engineering is the *upper contour set* $X(\varepsilon, h)$ defined for all $\varepsilon > 0$:

1. If $\overline{h} = \sup_X h(x) < \infty$, $X(\varepsilon; h) = \left\{x : h(x) > \overline{h} - \varepsilon\right\}$;
2. If $\overline{h} = \infty$, $X(\varepsilon; h) = \{x : h(x) > 1/\varepsilon\}$.

Closely related to $\overline{h}$ is the *modal set* $\overline{X} = \lim_{\varepsilon \to 0} X(\varepsilon; h)$.

Condition 1 *Three conditions are basic to the procedure:*
(a) $\mu[X(\varepsilon; h)] > 0 \ \forall \ \varepsilon > 0$*;*
(b) $0 < \underline{p} = \inf_{x \in X} p^{(0)}(x) \leq \sup_{x \in X} p^{(0)}(x) = \overline{p} < \infty$*;*
(c) For all $r \in [0, \infty)$*,* $k^*\left(x; h, p^{(0)}, r\right)$ *is finitely integrable on* X*.*

Departures from Condition 1(a) are possible if one tailors SABL to the particular attributes of h that violate this condition: for example, determine a set $X \subseteq \mathbb{R}^m$, $m < k$, for which the upper contour set $X^*(\varepsilon; h)$ satisfies Condition 1(a); and for all $x \notin S^*$ and some $\varepsilon > 0$, $h(x) < \overline{h} - \varepsilon$. A requirement similar to Condition 1(a) accompanies other numerical methods for optimization, for example steepest ascent, evolutionary and simulated annealing algorithms. These special treatments may be necessary for some applications but they are unusual and by definition cannot be addressed at the level of generality of this paper.

It seems unlikely that the instrumental distribution would not have a finite upper bound. That is used here solely as a technical device in one of the proofs and perhaps could be weakened but there is no compelling case for doing so. Condition 1(b) is also used as a technical device but might be more difficult to avoid and imposes practical constraints. The issue does not arise in the standard global optimization problem, which imposes $\mu(X) < \infty$. In econometrics it is familiar as a constraint that often appears in maximum likelihood theory (and often ignored in practice).The situation here is similar to the latter context, in that clearly $\mu(X) = \infty$ is inconsistent with $\underline{p} > 0$, and so formally one has to impose $\mu(X) < \infty$. This situation is not entirely satisfactory, but at least it is familiar. Once $\mu(X) < \infty$,Condition 1(c) is satisfied trivially if $\overline{h} < \infty$, a further condition commonly imposed in the global optimization and M-estimation literature.

These conditions are easy to violate when the optimization problem is maximum likelihood. A classic example is maximum likelihood estimation in a full mixture of normals model in which the likelihood function is unbounded when one component of the mean is exactly equal to one of the data points and the variance of that component approaches zero. This is not merely an academic nicety because when SABL is applied to optimization problems it really does isolate multiple modes and modes that are unbounded. (These problems are avoided in some approaches to optimization, including many implementations of the EM algorithm, precisely because those approaches do not successfully find the true modes.) More generally, this section carefully develops these and other conditions because they are essential to fully understanding what happens when SABL successfully maximizes objective functions that violate standard textbook assumptions but arise quite regularly in economics and econometrics.

Any system of equations of the form $e(x) = 0$ can be converted trivially to an optimization problem by taking $h(x)$ to be a monotone increasing transformation of $-e(x)' e(x)$, e.g. $h(x) = -\log\left[e(x)' e(x)\right]$ or $h(x) = -e(x)' e(x)$. For example, one can use this approach to study general equilibrium and in this context SABL is especially well suited to multiple, countable and uncountable equilibria compared with other approaches.

A few more definitions are required before stating the basic result and proceeding further. These definitions presume Condition 1. Associated with the objective function $h(x)$, its Boltzman transformation, and the initial density $p^{(0)}$,

1. The *Boltzman density* is $p\left(x; h, p^{(0)}, r\right) = k^*\left(x; h, p^{(0)}, r\right) / \int_X k^*\left(x; h, p^{(0)}, r\right) dx$;

2. The *Boltzman probability* is $P\left(S; h, p^{(0)}, r\right) = \int_S p\left(x; h, p^{(0)}, r\right) dx$ for all Borel sets $S \subseteq X$;

3. The *Boltzman random variable* is $\widetilde{x}\left(h, p^{(0)}, r\right)$ with $P\left(\widetilde{x}\left(h, p^{(0)}, r\right) \in S\right) = P\left(S; h, p^{(0)}, r\right)$ for all Borel sets $S \subseteq X$.

The basic result is:

Proposition 2 *Limiting particle concentration. Given Condition 1,*

$$\lim_{r \to \infty} P_r\left[X\left(\varepsilon; h\right)\right] = 1 \ \forall \ \varepsilon > 0.$$

Proof. See Section 8. ■

In interpreting the results of the SABL algorithm in situations that do not satisfy the standard conditions of a bounded continuous objective function with a unique global mode discretely greater than the value of $h\left(x\right)$ at the most competitive local mode it is important to keep Proposition 2 in mind precisely. For example it does not address the distribution across multiple global modes or neighborhoods of those modes, be they finite, countable or uncountable. It identifies precisely the subsets of X that will be identified in the limit; the relationship of these subsets to the global mode(s) themselves is a separate matter and many possibilities are left open under Condition 1. This is intentional: SABL succeeds in these situations when other approaches can fail, but the definition of success – which is Proposition 2 – is critical.

5.2 Convergence of SABL to the global mode

This section turns to conditions under which SABL provides a sequence of particles converging to the true mode. Basic to this enterprise is

Condition 2 *Singleton global mode*
(a) $\overline{X} = x^*$
(b) For all $\delta > 0 \ \exists \ \varepsilon > 0 : X\left(\varepsilon; h\right) \subseteq B\left(x^*; \delta\right) = \left\{x : \left(x - x^{*\prime}\left(x - x^*\right) < \delta^2\right)\right\}$

and

Proposition 3 *Consistency of mode determination. Given Conditions 1 and 2,*

$$\lim_{r \to \infty} P_r\left[B\left(x^*; \delta\right)\right] = 1 \forall \delta > 0.$$

Proof. None required ■

Condition 2(a) is essential to the possibility that $\lim_{r \to \infty} P\left(X; h, p^{(0)}, r\right)$ collapses about a singleton x^*. Given Condition 2(a) it is still possible that the competitors to x^* belong to a set far removed from x^*; more technically, for all $\varepsilon > 0$ the closure of $\left\{x : \overline{h} - \varepsilon < h\left(x\right) < \overline{h}\right\}$ might not include x^*. A simple example is $X = (0, 1)$, $h\left(1/2\right) = 1$, $h\left(x\right) = 1 - \left|x - (1/2)\right| \ \forall \ x \in (0, 1/2) \cup (1/2, 1)$.

Condition 2(b) is not necessary for SABL to provide constructive information about $\overline{X}$. Indeed when $\overline{X}$ is not a singleton it performs smoothly whereas it is difficult to learn about $\overline{X}$ using some alternative optimization methods. Here are two examples. In both cases the consequences follow from Proposition 2.

1. Consider the family of functions $h(x) = (x - x^*)' A (x - x^*)$, where $k > 1$ and A is a negative semidefinite matrix of rank $r < k$. Then $\overline{X}$ is an entire Euclidean space of dimension $k - r$. Particle variation orthogonal to this space vanishes as $r \to \infty$, but the distribution of particles within this space depends on $p^{(0)}(x)$.

2. As another class of examples, suppose $X = R^k$ and $h(x) = h(|x|) \; \forall \; x \in X$ satisfies Condition 2 on $X^+ = R^{k+}$. The function h displays 2^k reflected images and $\overline{X}$ is the union of 2^k points. The associated Boltzman distribution collapses about all 2^k points, with the relative probabilities depending on $p^{(0)}(x)$. In particular, as $r \to \infty$ the number of particles associated with the modes follows a multinomial distribution with $J \cdot N$ trials and probabilities according to $p^{(0)}(x)$ evaluated at the modes if $p^{(0)}(x)$ is continuous.

With additional regularity conditions familiar to any econometrician,

Condition 3 *Mode properties:*
(a) The initial kernel $p^{(0)}(x)$ is continuous at $x = x^*$;
(b) There is an open set S such that $x^* \in S \subseteq X$;;
(c) At all $x \in X$, $h(x)$ is three times differentiable and the third derivatives are uniformly bounded on X;
(d) At $x = x^*$, $\partial h(x) / \partial x = 0$, $H = \partial^2 h(x) / \partial x \partial x'$ is negative definite, and the third derivatives of $h(x)$ are continuous,

we have a result that, again, is perhaps unsurprising but has substantial practical value.

Proposition 4 *Asymptotic normality of particles. Given Conditions 1, 2 and 3, as $r \to \infty$, $\widetilde{u}\left(h, p^{(0)}, r\right) = r^{1/2} \left[\widetilde{x}\left(h, p^{(0)}, r\right) - x^*\right] \xrightarrow{d} N\left(0, -H^{-1}\right)$.*

Proof. See Section 8. ■

Denote the sample variance of the particles at power r by V_r. Proposition 4 shows that given Condition 3 $\lim_{r\to\infty} V_r = -H^{-1}$. For $\widehat{V}_{r,n}$ the corresponding sample variance of particles, by ergodicity of the algorithm $\lim_{n\to\infty} \lim_{r\to\infty} r^{-1} \widehat{V}_{r,n} = -H^{-1}$. The result has significant practical implications for maximum likelihood estimation, because it provides the asymptotic variance of the estimator as a costless by-product of the SABL algorithm. Condition 3 and the classical conditions for the asymptotic variance of the maximum likelihood estimator have much in common due to the similarity of the derivations, but the results do not: whereas the classical conditions are sufficient for the properties of the variance of the maximum likelihood estimator in the limit as sample size increases, Condition 3 is sufficient for $\lim_{n\to\infty}\lim_{r\to\infty} r^{-1}\widehat{V} = -H^{-1}$ regardless of sample size.

SABL requires fewer problem-specific investments than do competing methods like steepest ascent. It requires the correct statement and coding of the objective function. Steepest ascent requires, in addition, either the derivation and coding of an analytical derivatives, the computation of numerical derivatives with sufficient accuracy for the purposes at hand, or some combination of these two. (The number of function evaluations for numerical second derivatives can be comparable to the number of function evaluations in SABL when k is substantial, and steepest ascent with numerical derivatives cannot attain the accuracy demonstrated in the examples of Section 6.1) In addition SABL is robust to objective functions with pathologies like multiple modes and discontinuities, whereas steepest ascent is not.

5.3 Rates of convergence

One can also determine the limiting (as $\ell \to \infty$) properties of the power tempering sequence $\{r_\ell\}$. This requires no further conditions. The result shows that the sequence of *power increase ratios* $\rho_\ell = (r_\ell - r_{\ell-1})/r_{\ell-1}$ converges to a limit that depends on k but is otherwise independent of the optimization problem. Recall that x is a $k \times 1$ vector.

Proposition 5 *Given Condition 3, the limiting value of the power increase ratio ρ_ℓ for the power tempering sequence defined by (1) and the equation $f(r_\ell) = \eta^*$ for the RESS criterion $RESS = \eta^*$ is*

$$\rho = \lim_{\ell\to\infty} \rho_\ell = \eta^{*-2/k} - 1 + \left[\left(\eta^{*-2/k} - 1\right)\eta^{*-2/k}\right]^{1/2}. \qquad (12)$$

Proof. See Section 8. ■

This result is key to determining whether the variance matrix of particles is a reliable for the approximation $-H^{-1} \approxeq r_\ell \cdot \widehat{V}_{r_\ell}$. Since the validity of this approximation also implies $\rho_\ell \approxeq \rho$, it is clearly not advisable to use it unless the latter condition is verified. In applications where Condition 3 is satisfied this point is typically easy to identify by successive cycles ℓ in which ρ_ℓ fluctuates around ρ: the approximation of $-H^{-1}$changes little over these cycles. Thus for applications to which Conditions 1, 2 and 3 apply, a generic convergence criterion for the approximation of the Hessian emerges as a by-product of the algorithm.

A straightforward reading of Proposition 5 suggests these conditions will persist as ℓ increases beyond the values where they are first verified. In fact this does not happen: typically ρ_ℓ decreases substantially beyond a certain point with values of ρ_ℓ less than (say) $\rho/2$ occurring after perhaps a dozen more cycles. This is due to the digital evaluation of functions which is always noisy due to the finite number (usually 52) of mantissa bits used in floating point representation. If x_1 and x_2 are sufficiently close this becomes important in the comparison of $h(x_1)$ and $h(x_2)$, especially near the mode x^* of h since $\partial h(x)/\partial x \mid_{x=x^*} = 0$. (The problem is compounded for second derivatives – that is why machine approximation of the Hessian at the mode is not straightforward and reliable algorithms and code have value.) As cycles proceed, function evaluation is increasingly corrupted by digital noise because the particles are close, and the discrepancy between the conditions set down here and those that exist in the machine function evaluations at the particles increases. The underlying problem is that the digital evaluation of the weights in the C phase no longer adequately reflects the actual weights.

Fortunately, at the point where ρ_ℓ begins to depart from ρ the standard deviation of particles is typically sufficiently small that the value of x^* can be asserted reliably to more significant figures than are generally reported – it is usually possible to obtain at least 8 significant figures. When the units of $h(x)$ have substantive interpretation – like dollars – one can do even better by using the distribution of particles to determine that the standard deviation of $h(x)$ across particles is substantively trivial and further iteration of SABL would be pointless. These considerations underlie the convergence metrics provided as a by-product by SABL, from which the user may select for the purpose at hand; and it is also easy for the user to construct a custom convergence metric.

6 Optimization with SABL: Examples

6.1 Test functions from the global optimization literature

Irregular global optimization problems arise routinely in business and applied science. The simulated annealing algorithm (Kirkpatrick et al., 1983; Černý, 1985) has proved competitive for many global optimization problems in $\mathbb{R}^k$ (Černý, 1984; Vanderbilt and Louie, 1984; Geman and Hwang, 1986; Aluffi-Pentini et al., 1985; Dekkers and Aarts, 1991; Belisle, 1992). Variants of the algorithm using multiple values of the argument vector x simultaneously emerged only slightly later (Aarts et al., 1986a, 1986b; Xue, 1994; Hamma et al., 2000). More recent contributions (Molvalioglu et al., 2007, 2009; Zhou and Chen, 2013) use both multiple values of x and Metropolis steps.

The most recent work, including Zhou and Chen (2013) names the algorithm "simulated annealing / sequential Monte Carlo" (SA/SMC). This could be described as the SABL algorithm without adaptation – in particular the power function $\{r_\ell\}$; the variance matrix for the Metropolis random walk, the number of steps in the M phase, and the number of cycles are all deterministic, with no formal feedback from particles. (The sequence $\{r_\ell^{-1}\}$ is known as the temperature reduction schedule in the simulated annealing literature, and declaring this series at the outset is well recognized as the key problem in application.) There is some informal feedback in the sense that the first specification of these constants is almost certain to produce quite poor results, setting up an iteration between the results from the last attempt and revision of the constants by the investigator – tedious tuning, tinkering, trial and error. Perhaps in order to limit these iterations the SA/SMC algorithm confines attention to

$$r_\ell = \lambda \cdot \frac{10 \log(\ell+2)}{\max_{i=1,\ldots,n} h(x_i) + 10^{-9} \log(\ell+2)} \approxeq \lambda \cdot \frac{10 \log(\ell+2)}{\max_{i=1,\ldots,n} |h(x_i)|} \tag{13}$$

in the C phase.[5] In the M phase the Metropolis random walk source distribution for the candidate $\widetilde{x}$ is

$$\widetilde{x} \sim N\left(x, \alpha\beta^{-(\ell+1)} I_k\right) \tag{14}$$

and the random walk is not iterated. The number of particles n and the number of cycles L are also set in advance. Thus the manual design task involves setting the five parameters λ, α, β, n and L.[6]

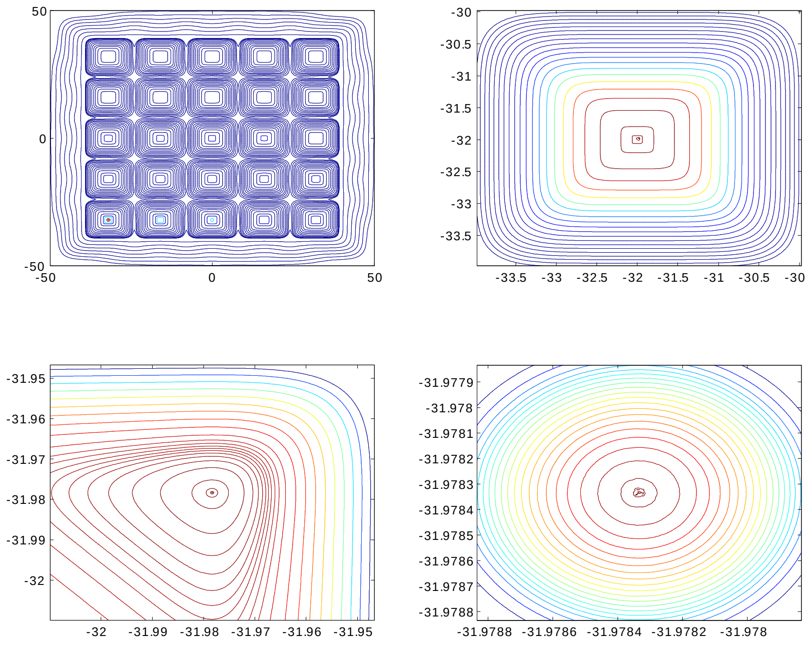


Figure 3: Contours of Dejong's 5th function

Zhou and Chen (2013) presents results for six test problems from the global optimization literature. That paper also demonstrates that their procedure provides results superior to those achieved in existing variants of simulated annealing, which makes it an attractive means for comparing SABL with simulated annealing more generally. The first test problem is

[5] We do not understand the presence of the term $\max_{i=1,\ldots,n} |h(x_i)|$ in (13): addition of an abitrary constant to h does not change the problem by can have a dramatic effect in (13).

[6] These details are not all in Zhou and Chen (2013). They are taken from the code for the work in that paper, which the authors generously shared.

- Test Problem 1: Dejong's 5^{th} function with $k = 2$,

$$h(x) = -\left[0.002 + \sum_{i=1}^{25} \frac{1}{i + \sum_{j=1}^{k} (x_j - a_{ij})^6}\right]; \tag{15}$$

  $X = [-50, 50]^2$, $a_{1\cdot} = e \otimes v$ and $a_{2\cdot} = v \otimes e$, where $e = (1, 1, 1, 1, 1)$ and $v = (-32, -16, 0, 16, 32)$.

The function has 25 local modes, each near – but not exactly the same as – a point of the form (v_i, v_j). The modes are not equal and the unique global mode is near the point $(-32, -32)$. Since this function has only two arguments, it is possible to study the behavior of the SABL optimization algorithm graphically, whereas the other examples have many more dimensions. Figure 3 (page 26) provides contours of $h(x)$ on X_d in the upper left panel. Successive panels show the finer structure that emerges from reducing ε in the upper contour set $X(h, \varepsilon)$ defined in Section 5.1.

The initial distribution is uniform on X, as it is in all of the six test problems both in this paper and in Zhou and Chen (2013). The `SABL` optimization algorithm terminated after 34 cycles. (This number is random, but varies little on re-execution.) Figure 4 (page 4) depicts the distribution of the particles at selected cycles. In each panel the axes coincide with the full range of particles so all particles are shown. The particles in the set $\overline{X}_\ell = \left\{x_i : h\left(x_i^{(\ell)}\right) = \max_{j=1,\ldots,n} h\left(x_j^{(\ell)}\right)\right\}$ are overplotted in red. The similarity of the particle distributions in Figure 4 to the contour plots in Figure 3is apparent: cycles 2 and 4 evoke the upper left panel, cycle 8 is similar to the upper right panel, cycle 16 exhibits the shape shown in the lower left panel, cycle 24 the lower right panel.

By cycle 28 the objective function $h(x)$ attains its maximum possible value $\overline{h}$ (to machine precision) for some of the particles. These are the particles overplotted in red in Figure 4 for cycles 28 and beyond. The remaining cycles gradually eliminate particles x_i for which $h(x_i)$ is less than this value. By cycle 34 they account for fewer than half the particles and the algorithm terminates. These results are consistent across independent executions of the algorithm, although the precise assortment of particles in $\overline{X}$ changes because there are many more values of x consistent with $\overline{h}$ (expressed in 64-bit floating point) than there are particles.

The other five test problems in Zhou and Chen (2013) are

- Test problem 2: Powel singular function $(k = 20)$

$$\begin{aligned} h(x) &= -\sum_{i=2}^{k-1} \left[(x_{i-1} + 10x_i)^2 + 5(x_{i+1} - x_{i+2})^2 \right. \\ &\quad \left. + (x_i - 2x_{i+1})^4 + 10(x_{i-1} - x_{i+2})^4\right] - 0.01 \end{aligned}$$

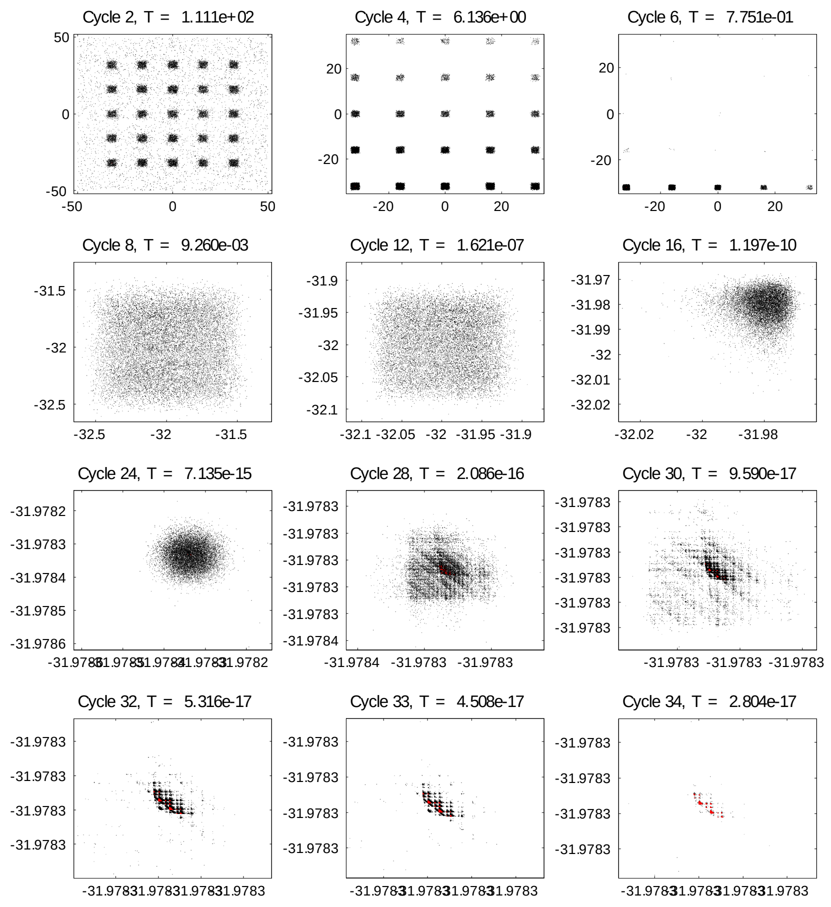


Figure 4: Distribution of particles in selected cycles, Dejong's 5th functino

- Test problem 3: Rosenbrock function ($k = 20$)

$$h(x) = -\sum_{i=1}^{k-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right] - 1$$

- Test problem 4: Griewank function ($k = 20$)

$$h(x) = -\left[\frac{1}{4000} \sum_{i=1}^{k} x_i^2 - \prod_{i=1}^{k} \cos\left(i^{-1/2} x_i \right) + 1 \right]$$

- Test problem 5: Trigonometric function ($k = 10$)

$$h(x) = -1 - \sum_{i=1}^{k} \left[8 \sin^2 \left(7 (x_i - 0.9)^2 \right) + 6 \sin^2 \left(14 (x_i - 0.9)^2 \right) + (x_i - 0.9)^2 \right]$$

- Test problem 6: Pintér's function ($k = 10$)

$$\begin{aligned} h(x) &= -\left[\sum_{i=1}^{k} i x_i^2 + \sum_{i=1}^{k} 20i \cdot \sin^2\left(x_{i-1}\sin x_i - x_i + \sin x_{i+1}\right)\right. \\ &\quad \left. + \sum_{i=1}^{k} i \log_{10}\left(1 + i\left(x_{i-1}^2 - 2x_i + 3x_{i+1} - \cos x_i + 1\right)^2\right)\right] - 10^{-15} \end{aligned}$$

In all cases $X = [-50, 50]^k$ and the initial distribution is uniform on this set. In each case the global model is a singleton in $(-50, 50)^k$, but the set of highly competitive arguments and local modes is challenging to eliminate – that is why these test problems have persisted in the global optimization literature. Yet all of these functions satisfy Condition 3 and so Propositions 4 and 5 apply. The consequences are clear in Figure **5**. In every case the power increase ratio $\{\rho_\ell\}$ goes through three stages.

1. In the first stage r_ℓ is not sufficiently great that the particles are yet limited to a neighborhood of the global mode in which the quadratic approximation is valid.

2. The second stage conforms with Proposition 5.

3. Consistent with the discussion in Section 5.3, in the third and final stage the ratio drops below the asymptotic value ρ as the evaluation of the objective function is increasingly dominated by the limitations of machine precision in evaluating $h(x)$.

In the case of Test Problem 1 (upper left panel of Figure **5**) this interpretation can be verified by comparison with Figure **??**.

For each of the six test problems Table 3 (page 43) summarizes some aspects of the problem (Rows 1-3), provides some characteristics of the SABL solution (Rows 4-11), and records some aspects of the solution achieved by the SA/SMC algorithm (rows 12-25). In Row 3 there is only one value for the solution because in each of these test problems the maximizing arguments x_i^* $(i = 1, \ldots, k)$ are all the same.

All of the results for `SABL` used the default number of particles, 16,384. With the exception of test problem 5 the algorithm parameters (e.g. the relative effective sample size in the C phase and the target Metropolis acceptance rate in the M phase) were the `SABL` default values. In test problem 5 the M phase used the default blocked Metropolis algorithm mentioned in Section 4. `SABL` was executed using CPU cores, not a GPU, because the function evaluations are so simple that there is little gain from moving to GPU execution; run times are fast anyway, ranging from under a minute to about five minutes. The results for SABL are taken from the last cycle L of the algorithm, in turn defined by the condition that the maximum value of the objective function was the same evaluated across at least half of the particles.

The results in row 4 can be explained readily using the fact that the ratio of any two numbers is an integer multiple of 2^{-52} because the mantissa has 52 bits. In row

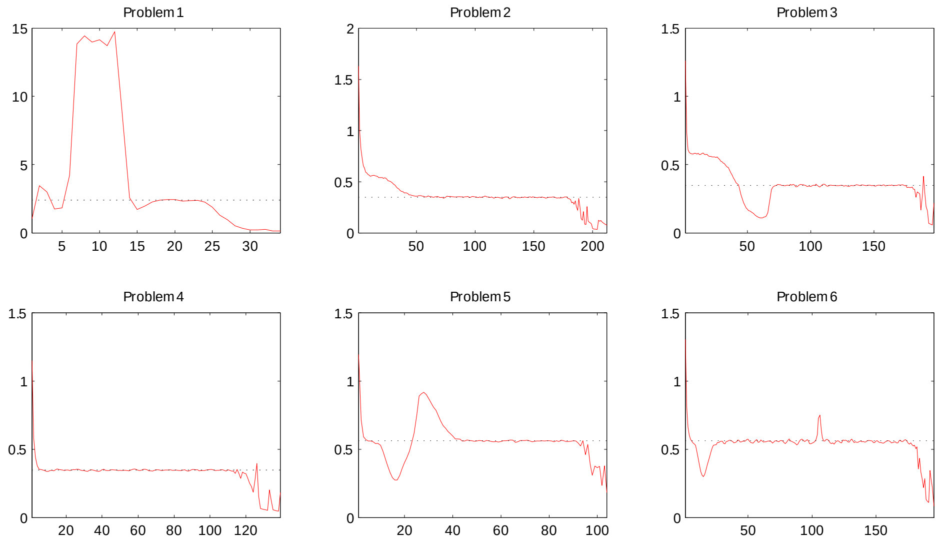


Figure 5: Power increase ratio ρ_ℓ in the test problems (solid line) and the limiting value ρ of Proposition 4 (horizontal dotted line)

4, the evaluation of h^* is exact if it is wholly divisible by 2^{-52} and otherwise it is less than $\left(2^{-52}\right) h^*$ in absolute value. The error bound in row 5 is the difference between h^* and the smallest value of h across all particles. From row 5 it may be deduced that at the end of cycle L the objective function h had only two unique values $h^{(1)} > h^{(2)}$, across all particles, and $\exp\left(h^{(1)}\right) / \exp\left(h^{(2)}\right) - 1 \approxeq 2^{-52}$, in each test problem. For each test problem the range in row 6 is $k^{-1} \sum_{i=1}^{k} \left[\max\left(x_i\right) - \min\left(x_i\right)\right]$, where the minimum and maximum are taken over all particles at the end of the last cycle. These values are much larger than the h^* error bound because h is locally quadratic at the maximum x^*: a straightforward application of Taylor's theorem shows that the evaluation of $h\left(x_1^*, \ldots, x_{i-1}^*, x_i, x_{i=1}^*, \ldots, x_k^*\right)$ is indiscernible from that of $h\left(x^*\right)$, in floating point representation, if and only if $\left|x_i - x^*\right| \precapprox 2^{-51/2}\left[\partial^2 h / \partial x_i^2\right]^{1/2}$. To a rough approximation this implies that ranges on the order of the square root of the h^* error bound might be anticipated, and row 6 is consistent with this.

Rows 12 through 16 document the choices of the 5 free algorithm parameters made for the SA//SMC algorithm for each test problem. Recall that the entire procedure was repeated 100 times in each case. Rows 17 and 18 provide the mean and standard deviation of the error in approximating h^* over these 100 repetitions, and rows 20 and 21 do the same for the first argument of the objective function. The double asterisk (rows 19 and 22) denotes the single particle x^{**} that produced the largest value h^{**} of the objective function over all cycles and repetitions. These results show that the SA/SMC algorithm is grossly inaccurate compared with the SABL algorithm. The final rows for each method, in Table 3, record the number of function evaluations entailed, a standard method of comparing computational efficiency in the global optimization literature, and the execution time on the same quadcore CPU for each. The geometric mean of the ratio of SABL evaluations to SA/SMC evaluations across the six test problems is 0.275; for execution times it is 0.240.

A major factor in the relatively quite poor performance of SA/SMC is the power increase schedule in (13): the power increase ratio is approximately $\rho_\ell = (r_\ell - r_{\ell-1})/r_\ell = \log(\ell+1)/\log(\ell) - 1 \to 0$, whereas in SABL the limiting value is ρ given in (12). The consequences are evident in the final power r_L of the SA/SMC algorithm (row 23 of Table 3): this, alone, precludes accurate determination of the mode. Another major inefficiency in the SA/SMC algorithm is due to the fact that the Metropolis variance matrix (14) collapses much faster than is warranted by the rate of power increase, with the consequence that very quickly this step does not mix the particles that emerge from resampling. An effective non-adaptive SA/SMC algorithm could not be designed without first understanding the theory developed in Section 5.3. By employing a generic sequentially adaptive strategy that applies to a wide set of problems of which global optimization is only one, SABL achieves efficiency without the theory developed in Section 5 – indeed, in our work the theory in Section 5 was motivated by the observed ability of SABL to solve a variety of optimization problems to machine accuracy.

6.2 Maximum likelihood in the AR(3) GDP model

The section reviews the performance of the SABL optimization model as applied to maximum likelihood for the model and data of Section 4.2. Recall from Section 5.2 that in this setting the asymptotic variance matrix of the maximum likelihood estimator emerges as a by-product from SABL under the classical regularity conditions for the likelihood function.

Figure 6 (page 32) documents the performance of SABL for this problem. The top panel documents the same three different stages of the algorithm discussed in Section 6.1. In the second stage, between cycles 18 and 50, the power increment ratio $\rho_\ell = \left(\rho_\ell - \rho_{\ell-1}\right)/\rho_{\ell-1}$ fluctuates about the asymptotic value $\rho = 0.9688$ (for $k = 5$) of Proposition 5. From the middle panel, the maximum likelihood estimates are constant up to pixellation. This condition persists through the third stage as well as particles contract even more tightly about the mode (though at a slower rate than in the second stage). The bottom panel provides the asymptotic standard errors – square roots of the diagonal elements of $r_\ell \cdot \widehat{V}_\ell$ at each iteration. These are also constant in the second stage of cycles. They rise in the third stage as $\widehat{V}_\ell$ increasingly reflects noise due to the limitations of floating point representation of likelihood function evaluations.

Figure 7 (page 33) tracks the distribution of particles through some representative cycles. In the upper left panel $r_5 \approxeq 1$ and so the distribution is very close to that in Figure 2 (page 19), lower left panel. In the second stage of cycles inferred from ρ_ℓ, and a very few adjacent ones as well, the density conforms very well with that inferred from $r_\ell \cdot \widehat{V}_\ell$ ($\ell = 14, 30, 46, 50$). The deterioration due to the limitations of machine arithmetic become apparent in the last two panels. Indeed, in the final panel ($\ell = 70$) one can

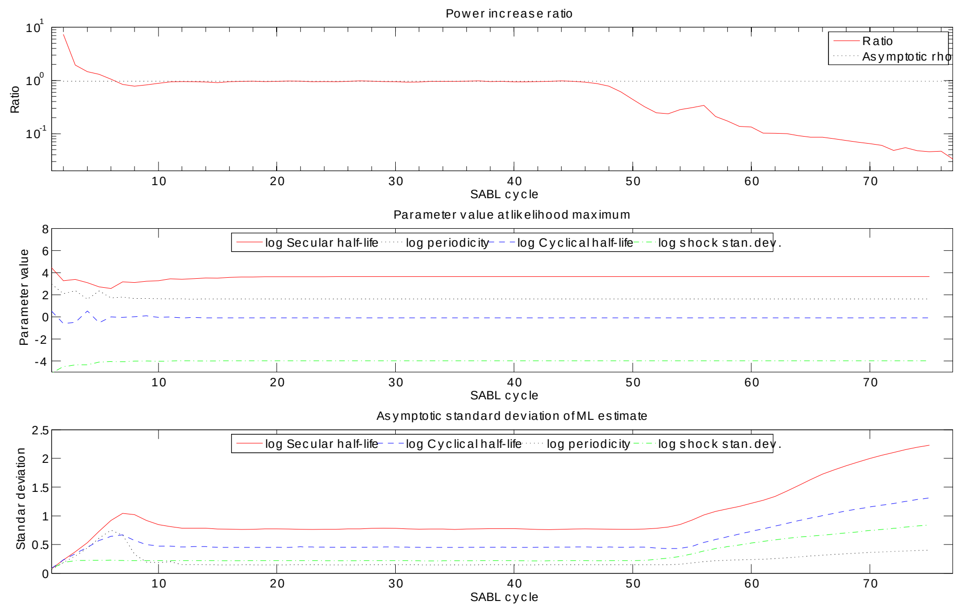


Figure 6: Some aspects of SABL applied to maximum likelihood

discern the emergence of a step function, three different values of the likelihood function reflecting changes in the two lowest order bits of floating point representation.

Table 4 (page 43) presents the maximum likelihood estimates of the four parameters of interest and their asymptotic standard errors in the columns headed "direct ML". The two columns to the left provide the posterior means and standard deviations from the Bayesian inference example in Section 4.2. The maximum likelihood results differ substantially from the Bayesian inference results, especially for $\log(p)$: the posterior mean has a classical p-value of 2.086×10^{-3}, whereas the MLE is well within a posterior highest credible interval of any conventional size. The mechanics of this difference are readily traced to the decidedly non-quadratic likelihood as a function of $\theta_4 = \log(p)$ evident in Figure 7, upper left panel, and Figure 2, lower left panel. The differences for $\log(h_s)$ and $\log(h_c)$ have similar origins, though the contrast is not so marked as it is for $\log(p)$.

Yet a third approach is to use the relatively quite simple AR(3) likelihood function directly. The consequences of this approach appear in the two far right columns in Table 4. Mapping the OLS estimate of β and the corresponding estimation of σ^2 into θ by means of (11) then provides the MLE, which of course must be the same as the estimate found by the SABL optimization algorithm. Just what "the" asymptotic standard errors are for this procedure is a matter of definition. These could be based on the gradients of the mapping from β to θ at the MLE. Those in the last column of Table 4 are obtained by transforming values of β and σ, simulated from the asymptotic normal distribution, to θ and then computing the sample variance matrix. On the whole, they are much closer

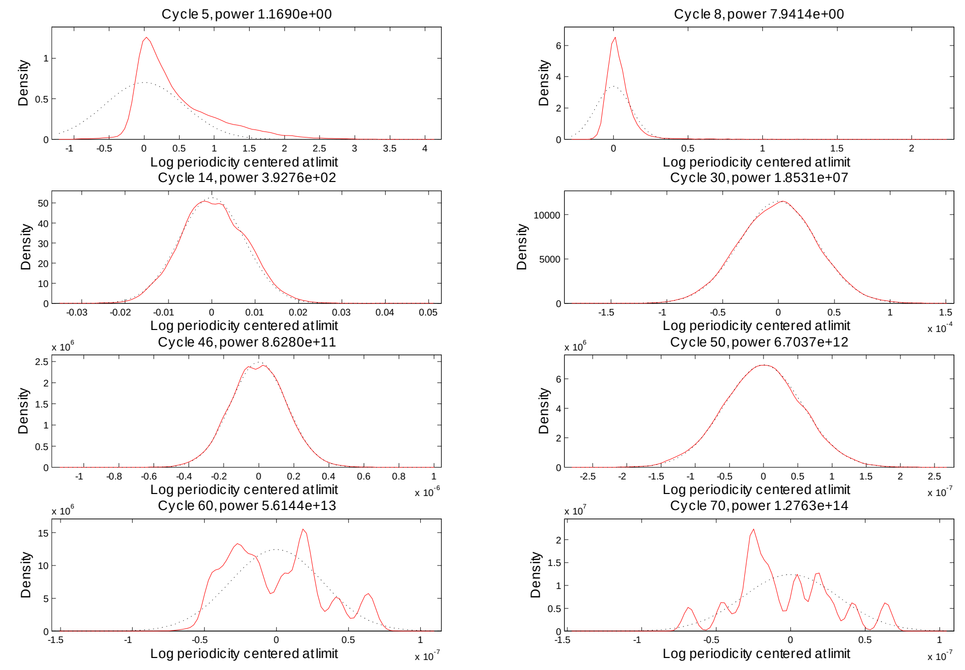
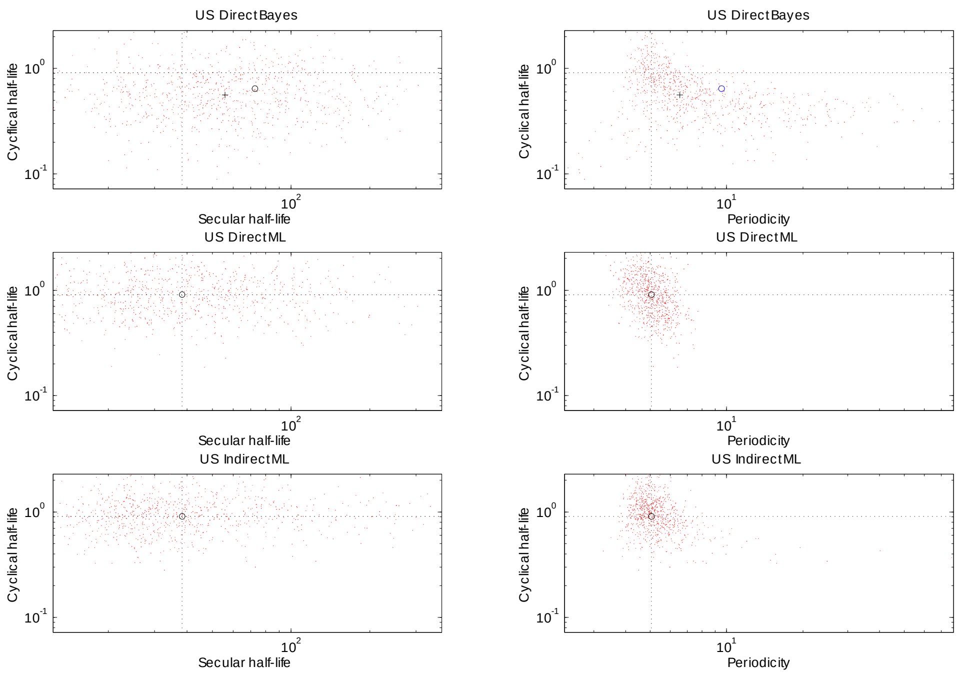


Figure 7: Distribution of θ_3 (period) particles in selected cycles ℓ (solid line). The normal density consistent with standard deviation is the dotted line. The horizontal axis is $\theta_3 + 0.0962319609$.

to the direct ML asymptotic standard errors than to the posterior standard deviations.

Figure 6.2 provides another perspective on the comparison between the three approaches to inference, for the joint distribution of secular and cyclical half-life (left panels) and the joint distribution of period and cyclical half-life (right panels). To facilitate comparison the axes are identical in each column and were selected so as to exclude the 0.25% smallest and 0.25% largest points out of 820 selected from the 16,348 particles in the posterior distribution. The number of points plotted in the last two rows is almost 820 since the dispersions there are smaller. The horizontal and vertical lines are the same in each column, intersecting at the maximum likelihood estimate, which is indicated by the circle in the last two rows. In the top pair of panels the circle is the mean of the posterior distribution and the cross is the median. The smaller dispersion of the parameters under the asymptotic ML distributions, especially for $\log(p)$, is striking in these figures. So, too, are the differences in shape. The joint distribution under the direct ML asymptotic expansion is Gaussian by construction; as already documented and well-understood, the posterior distribution is not; but the indirect ML distribution is also non-Gaussian because the transformation from (β, σ) to θ is nonlinear.

The indirect ML asymptotic expansion is very close to the exact posterior distribution using the conventional improper prior distribution $p(\beta.\sigma) \propto \sigma^{-1}$, sometimes called the



Jeffreys prior on (less precisely) “diffuse” or “uninformative.” This is in fact not a Jeffreys prior for linear regression except when it reduces to mean estimation. Whereas the prior in β is flat, the implied prior for θ is not due to the Jacobian implied by the nonlinear transformation. The fact that “diffuse” or “uninformative” prior distribution can be vague, slippery or vacuous concepts is sometimes missed by Bayesian econometricians, like Geweke (1988).

7 `SABL` toolbox

Matlab is proprietary software developed and leased by The Mathworks. It is widely used in industry and academia. For academics interested in an individual license substantial academic discounts are available. Many research universities have site licenses for Matlab and in many cases it is freely available to university staff. Matlab operates under Linux, OS and Windows operating systems. Matlab has constantly improved and expanded since its inception almost 30 years ago and this continues. The Mathworks has a large PhD staff including several substantial contributors to scientific computing with independent academic reputations.

Matlab supports a wide variety of basic mathematical operations in is core software. This core is supplemented by over fifty proprietary Matlab toolboxes with specialized capabilities, for example Statistics and Machine Learning, Parallel Computing, and Econometrics. Third parties have developed hundreds of additional toolboxes, most of them nonproprietary. After installation toolboxes become an intimate part of Matlab, extending its capabilities by anything from a few functions to provision of entire platforms specific kinds of work.

`SABL` is a third-party nonproprietary toolbox developed at the University of Technology Sydney, Australia. It provides an entire platform that makes it straightforward for users to apply the SABL algorithm to problems in Bayesian inference and optimization, and for third parties to contribute new models or variations on the SABL algorithm presented in this paper. `SABL` requires the Matlab Statistics and Machine Learning Toolbox, and for GPU computing or explicitly parallel CPU computing it requires the Matlab Parallel Toolbox as well. For GPU computing it is also necessary to have the C/CUDA compiler installed. Matlab and `SABL` support GPU computing only with Nvidia chips, which are commonly found in university linux clusters and many clouds. `SABL` can be installed under the same operating systems as Matlab; currently, however, GPUs cannot be installed on Macs. The *SABL Handbook* provides details on all of these matters.

`SABL` is designed to meet several objectives

1. Permit users whose objective is to solve a particular problem at hand to exploit the power of the SABL algorithm and inexpensive, fast, massive parallel computing easily, without having to involve themselves in the arcane difficulties of GPU computation or the tedious tuning, tinkering, trial and error that so often accompany posterior simulation or optimization.

2. Provide a platform with ready ports of entry for new models and new objective functions developed by third-party researchers.

3. Minimize the time and effort required to make future improvements in SMC algorithms available for practical day-to-day work and expand the toolkit of researchers addressing challenging problems in Bayesian inference and optimization.

`SABL` incorporates a wide variety of features in order to meet these objectives. Here are some of the more important ones.

1. The specification of the algorithm is governed by 50 parameters (e.g., groups J and number N of particles in each group, RESS criterion in the C phase, criteria for termination of the Metropolis steps in the M phase). These parameters all have default values that work well for most applications.

    (a) In this respect `SABL` is "hands free", avoiding the tedious tuning, tinkering, trial and error that is so often encountered with new models or even new applications using alternatives like importance sampling and Markov chain Monte Carlo for Bayesian inference or steepest ascent and simulated annealing for optimization.

   (b) Modellers can set alternative default values for users and the user can change any of the default values. It is the rare user who would need to know about all of these parameters, much like most smart phone users remain unaware of all of the options to tailor their device.

2. `SABL` exploits the fact that the SABL algorithm is embarrassingly parallel in the C and M phases, and the S phase accounts for a tiny fraction of execution time. The algorithm is pleasingly parallel overall.

   (a) It is therefore well suited to execution using graphics processing units, which provide massively parallel computing with thousands of cores at rates of well under a dollar per hour for occasional users by means of cloud computing and a few cents per hour for intensive users by means of local installation. It is impossible to reduce the advantages of GPU to a single speed-up factor, primarily because this varies greatly by application and the goal of the application, but also because the number of installed cores per GPU is currently increasing at rates exceeding Moore's law. Spedup factors of 10 to 100 are common, and there are specific instances in which alternatives with other algorithms would require eons.

   (b) `SABL` is also well suited to CPU environments with dozens of cores, which are becoming more common. Because Matlab efficiently exploits options in C for parallel loop execution, `SABL` can achieve speedup relative to single core computing by a factor approaching the number of CPU cores even without the Matlab Parallel Toolbox installed. This is not nearly as fast as GPU execution but is still a substantial improvement on conventional quadcore computing.

3. `SABL` is completely self-documenting, with an on-line **`help`** command system, an extension of the Matalb **`help`** command system to documentation and the system of global structures employed by `SABL` as well as the functions that comprise `SABL`.

4. `SABL` executes in both CPU and GPU environments, including multiple GPUs. Users switch from one to the other using a single parameter. While many common operations (e.g., allocating space for an array or computing standard deviation) are sensitive to the type and number of processors, these differences are all managed by an extensive set of `SABL` utilities. They are invisible to the modeller or user, who can write a single set of code that executes efficiently in all of these environments.

5. `SABL` provides all of the common prior distributions, a systematic way to truncate prior distributions, and a systematic way to mix a continuous prior distribution with a discrete distribution. Augmenting this set of prior distributions is straightforward. Both modellers and users can provide customized prior distributions and `SABL` will incorporate these for the model or application at hand in the same was as the common prior distributions.

6. Because incorporation of incremental information is part and parcel of SABL and all SMC algorithms, the work required to update the posterior distribution given new information is incremental, unlike the case in IS or MCMC where the work must be repeated almost in its entirety. Thus SMC algorithms are tailored to real-time posterior updating in a way that other posterior simulators are not. `SABL` makes this very simple for the user, who must only specify two file names, one for the particles of the SABL algorithm without the updating information set and one to record the particles after updating.

7. `SABL` incorporates the two-pass variant of the SABL algorithm discussed in Section 3.1, simply by the change of a single algorithm parameter.

8. Because the marginal likelihood is a generic by-product of the SABL algorithm, `SABL` provides the approximation of the log marginal likelihood and its numerical standard error in all cases. The accuracy of these evaluations is excellent in comparison with competitors.

9. `SABL` provides a great deal of additional information in the process of execution. For example, with complete data tempering (one observation introduced each cycle) one has access to every updated posterior distribution, which together with the data make it possible to assess all predictive likelihoods in a single sweep. `SABL` passes control to the user at many points in each cycle, where the user can harvest this intermediate information for the purposes at hand. The basic `SABL` user need not even be aware of this possibility, which provides a wide range of opportunities for the more experienced Bayesian econometrician or statistician.

8 Proofs of propositions

Proposition 1. The function $f(r_\ell)$ in (1) is monotone decreasing for $r_\ell \in [r_{\ell-1}, \infty)$. Let m be the number of (i,j) for which $k\left(\theta_{ij}^{(\ell-1)}\right) = \max_{i,j} k\left(\theta_{ij}^{(\ell-1)}\right)$. If $RESS^* \in (m/JN, 1)$ then $f(r_\ell) = RESS^*$ has exactly one solution $r_\ell \in (r_{\ell-1}, \infty)$; otherwise it has none.

Proof: To simplify notation replace $r_\ell - r_{\ell-1}$ with $r \geq 0$ and the arguments $\theta_{jn}^{(\ell-1)}$ with θ_i $(i = 1, \ldots, n)$, and write $k(\theta_i) = k_i$. Then (1) becomes

$$f(r) = \frac{\left(\sum_{i=1}^n k_i^r\right)^2}{n \sum_{i=1}^n k_i^{2r}} = \frac{\left(n^{-1} \sum_{i=1}^n k_i^r\right)^2}{n^{-1} \sum_{i=1}^n k_i^{2r}}.$$

By inspection $f(0) = 1$. Since only relative values of k_i matter, any positive multiplicative renormalization of the k_i is permissible. Adopting the normalization $\max_{i=1,\ldots,n}(k_i) = 1$,

$$\lim_{r\to\infty} f(r) = \frac{m^2}{nm} = \frac{m}{n},$$

hence the requirement $RESS^* \in (m/JN, 1)$. Since $f(r)$ is continuous this establishes the existence of at least one solution $r_\ell \in (r_{\ell-1}, \infty)$.

Suppose that there are u positive values of k_i. Re-order the k_i so that $k_i > 0$ $(i = 1, \ldots, u)$ and write

$$f(r) = \frac{u}{n} \cdot \frac{\left(u^{-1}\sum_{i=1}^{u} k_i^r\right)^2}{u^{-1}\sum_{i=1}^{u} k_i^{2r}} = \frac{u}{n} \cdot \frac{g(r)}{h(r)}.$$

It suffices to show $h'(r) > g'(r) > 0 \; \forall \; r > 0$. For this purpose adopt the normalization $\min_{i=1,\ldots,u}(k_i) = 1$. We have

$$g'(r) = 2\left(u^{-1}\sum_{i=1}^{u} k_i^r\right) \cdot \left(u^{-1}\sum_{i=1}^{u} k_i^r \log k_i\right), \quad h'(r) = 2u^{-1}\sum_{i=1}^{u} k_i^{2r} \log(k_i)$$

and

$$h'(r) - g'(r) = 2\text{cov}\left(k_i^r, k_i^r \log k_i\right) > 0.$$

Proposition 2 (Limiting particle concentration) Given Condition 1,

$$\lim_{r\to\infty} P_r\left[X(\varepsilon; h)\right] = 1 \; \forall \; \varepsilon > 0.$$

Proof: If $\mu\left[x : h(x) < \overline{h}\right] = 0$ then the result is trivial. Otherwise there exists $\varepsilon^* > 0$ such that for all $\varepsilon \in (0, \varepsilon^*)$, $P\left[X(\varepsilon; h)^c; h, p^{(0)}, r\right] > 0$. For all $\varepsilon \in (0, \varepsilon^*)$ and for $\overline{h} < \infty$,

$$\begin{aligned}
\frac{P\left[X(\varepsilon); h, p^{(0)}, r\right]}{P\left[X(\varepsilon; h)^c; h, p^{(0)}, r\right]} &\geq \frac{P\left[X(\varepsilon/2; h); h, p^{(0)}, r\right]}{P\left[X(\varepsilon; h)^c; h, p^{(0)}, r\right]} \geq \frac{\int_{X(\varepsilon/2;h)} p^{(0)}(x)\,dx}{\int_{X(\varepsilon;h)^c} p^{(0)}(x)\,dx} \cdot \frac{\exp\left[r \cdot \left(\overline{h} - \frac{\varepsilon}{2}\right)\right]}{\exp\left[r \cdot \left(\overline{h} - \varepsilon\right)\right]} \\
&= \frac{\int_{X(\varepsilon/2;h)} p^{(0)}(x)\,dx}{\int_{X(\varepsilon;h)^c} p^{(0)}(x)\,dx} \cdot \exp(r \cdot \varepsilon/2) \longrightarrow \infty.
\end{aligned}$$

For $\overline{h} = \infty$, replace $\exp\left[r \cdot \left(\overline{h} - \frac{\varepsilon}{2}\right)\right] / \exp\left[r \cdot \left(\overline{h} - \varepsilon\right)\right]$ with $\exp\left(r/(\varepsilon/2)\right)/\exp(r/\varepsilon)$.

Proposition 4. Asymptotic normality of particles. Given Condition 3, as $r \to \infty$, $\widetilde{u}\left(h, p^{(0)}, r\right) = r^{1/2}\left[\widetilde{x}\left(h, p^{(0)}, r\right) - x^*\right] \xrightarrow{d} N\left(0, -H^{-1}\right)$.

Proof. We show that the p.d.f. of $\widetilde{u}\left(h, p^{(0)}, r\right)$ converges pointwise to the normal density $p_N\left(u; 0, -H^{-1}\right)$ and the result then follows by Scheffe's Lemma.

For any $u \in \mathbb{R}^k$ consider the sequence of points $x_r = x^* + u \cdot r^{-1/2}$. Applying Taylor's Theorem with the Lagrange form of the remainder, the corresponding sequence of Boltzman kernels evaluated at z is

$$\begin{aligned}
& p^{(0)}\left(x^* + ur^{-1/2}\right) \cdot \exp\left[rh\left(x^* + ur^{-1/2}\right)\right] \\
= \; & p^{(0)}\left(x^* + ur^{-1/2}\right) \cdot \exp\left[rh(x^*)\right] \cdot \exp\left(u'Hu/2\right) \cdot \exp\left[r \cdot c\left(ur^{-1/2}\right)\right].
\end{aligned}$$

From condition **??** $c\left(ur^{-1/2}\right) = O\left(r^{-3/2}\right)$, so $\lim_{r\to\infty} r \cdot c\left(r,u\right) = 0$. Hence the limiting kernel, unique up to normalization, is

$$\begin{aligned}
&\lim_{r\to\infty} \frac{p^{(0)}\left(x^* + u\cdot r^{-1/2}\right)\exp\left[rh\left(x^*\right)\right]\exp\left(u'Hu/2\right)}{p^{(0)}\left(x^*\right)\exp\left[rh\left(x^*\right)\right]} \\
= &\lim_{r\to\infty} \frac{p^{(0)}\left(x^* + u\cdot r^{-1/2}\right)}{p^{(0)}\left(x^*\right)}\exp\left(u'Hu/2\right) = \exp\left(u'Hu/2\right).
\end{aligned}$$

Proposition 5. Given Condition 3, the limiting value of the power increase ratio ρ_ℓ for the power tempering sequence defined by (1) and the equation $f\left(r_\ell\right) = \eta^*$ for the $RESS$ criterion $RESS = \eta^*$ is

$$\rho = \lim_{\ell\to\infty} \rho_\ell = \eta^{*-2/k} - 1 + \left[\left(\eta^{*-2/k} - 1\right)\eta^{*-2/k}\right]^{1/2}.$$

Proof. The power tempering sequence satisfies

$$\begin{aligned}
&\left\{\frac{1}{n}\sum_{i=1}^{n}\exp\left[h\left(x_i^{(\ell-1)}\right)\right]^{(r_\ell - r_{\ell-1})}\right\}^2 - \eta^*\cdot\frac{1}{n}\sum_{i=1}^{n}\exp\left[h\left(x_i^{(\ell-1)}\right)\right]^{2(r_\ell - r_{\ell-1})} = 0 \iff \\
&\left\{\frac{1}{n}\sum_{i=1}^{n}\exp\left[\left(r_\ell - r_{\ell-1}\right)h\left(x_i^{(\ell-1)}\right)\right]\right\}^2 - \eta^*\cdot\frac{1}{n}\sum_{i=1}^{n}\exp\left[2\left(r_\ell - r_{\ell-1}\right)h\left(x_i^{(\ell-1)}\right)\right] = 0 \\
&\iff \exp\left[\left(r_\ell - r_{\ell-1}\right)h\left(x^*\right)\right]^{-2}\left\{\frac{1}{n}\sum_{i=1}^{n}\exp\left[\left(r_\ell - r_{\ell-1}\right)h\left(x_i^{(\ell-1)}\right)\right]\right\}^2 \\
&-\eta^*\cdot\exp\left[\left(r_\ell - r_{\ell-1}\right)h\left(x^*\right)\right]^{-2}\sum_{i=1}^{n}\exp\left[2\left(r_\ell - r_{\ell-1}\right)h\left(x_i^{(\ell-1)}\right)\right] = 0. \qquad (16)
\end{aligned}$$

Turning to the first term,

$$\begin{aligned}
\exp\left[\left(r_\ell - r_{\ell-1}\right)h\left(x_i^{(\ell-1)}\right)\right] &= \exp\left[\left(r_\ell - r_{\ell-1}\right)h\left(x^*\right)\right] \\
\cdot\exp\left[\frac{\left(r_\ell - r_{\ell-1}\right)}{2}\left(x_i^{(\ell)} - x^*\right)' H\left(x_i^{(\ell)} - x^*\right)\right] &\cdot \exp\left[\left(r_\ell - r_{\ell-1}\right)\cdot O\left(r_{\ell-1}^3\right)\right]. \qquad (17)
\end{aligned}$$

So long as $\lim_{\ell\to\infty}\left[\left(r_\ell - r_{\ell-1}\right)/r_{\ell-1}^3\right] = 0$ the last term can be ignored. Since the proof goes through after imposing a positive upper bound on $\left(r_\ell - r_{\ell-1}\right)/r_{\ell-1}^2$ and since $\lim_{\ell\to\infty}\left(r_\ell - r_{\ell-1}\right)/r_{\ell-1}^2 = 0$, this is harmless. The first term on the right side of (17) vanishes in the re-normalization in (16), leaving just the middle term. Substituting $u = r_{\ell-1}^{1/2}\left(x_i^{(\ell)} - x^*\right)$, this term is

$$\exp\left[\frac{\left(r_\ell - r_{\ell-1}\right)}{2r_{\ell-1}}u'Hu\right] = \exp\left(\rho_\ell u'Hu/2\right).$$

From Proposition 3

$$\begin{aligned}\lim_{\ell\to\infty} E_{(\ell-1)}\left[\exp\left(\rho_\ell u'Hu/2\right)\right] &= E_{u\sim N(0,-H^{-1})}\left[\exp\left(\rho_\ell u'Hu/2\right)\right] \\ &= E_{u\sim N(0,H^{-1})}\left\{\exp\left[-\rho_\ell u'Hu\right]\right\}.\end{aligned}$$

Using the Choleski decomposition $J'J = H^{-1}$ make the standard transformation $z = Ju \sim N(0, I_k)$ and then

$$\begin{aligned}E_{u\sim N(0,H^{-1})}\left\{\exp\left[-\rho_\ell u'Hu/2\right]\right\} &= E_{z\sim N(o,I_k)}\left\{\exp\left[-\rho_\ell z'z/2\right]\right\} \\ &= \prod_{i=1}^{k} E_{N(0,1)} \exp\left[-\rho_\ell z_i^2/2\right]. \qquad (18)\end{aligned}$$

The term inside the product is

$$\int_{-\infty}^{\infty} \exp\left[-\rho_\ell z^2/2\right](2\pi)^{-1/2}\exp\left(-\frac{z^2}{2}\right)dz = (2\pi)^{-1/2}\int_{-\infty}^{\infty}\exp\left[-\left(1+\rho_\ell\right)z^2/2\right]dz = \left(1+\rho_\ell\right)^{-1/2}.$$

Since the particles are ergodic, we have for the first term in (16)

$$\lim_{n\to\infty}\lim_{\ell\to\infty}\frac{\left\{\frac{1}{n}\sum_{i=1}^{n}\exp\left[\left(r_\ell - r_{\ell-1}\right)h\left(x_i^{(\ell-1)}\right)\right]\right\}^2}{\exp\left[\left(r_\ell - r_{\ell-1}\right)h\left(x^*\right)\right]^2} = \left(1+\rho_\ell\right)^{-k}.$$

To deal with the other term proceed in exactly the same way, substituting $2\left(r_\ell - r_{\ell-1}\right)$ for $\left(r_\ell - r_{\ell-1}\right)$ in (17), which removes the division by 2 in (18), and then $\left(1+2\rho_\ell\right)^{-1/2}$ is the last term in (8). Thus for the second term in (16) we have

$$\lim_{n\to\infty}\lim_{\ell\to\infty}\eta^*\cdot\frac{\frac{1}{n}\sum_{i=1}^{n}\exp\left[2\left(r_\ell - r_{\ell-1}\right)h\left(x_i^{(\ell-1)}\right)\right]}{\exp\left[2\left(r_\ell - r_{\ell-1}\right)h\left(x^*\right)\right]} = \eta^*\left(1+2\rho_\ell\right)^{-k/2}$$

The limiting value ρ satisfies

$$\begin{aligned}(1+\rho)^{-k} &= \eta^*\cdot(1+2\rho)^{-k/2} \iff (1+\rho)^2 = \eta^{*-2/k}(1+2\rho) \\ &\iff \rho^2 + 2\left(1-\eta^{*-2/k}\right)\rho + \left(1-\eta^{*-2/k}\right) = 0 \\ &\iff \rho = \eta^{*-2/k} - 1 \pm\left[\left(\eta^{*-2/k}-1\right)^2 + \left(\eta^{*-2/k}-1\right)\right]^{1/2}.\end{aligned}$$

The RESS target $\eta^* \in (0,1)$ and hence $\eta^{*-2/k} - 1 > 0$. Since $\rho > 0$ the pertinent branch is

$$\rho = \eta^{*-2/k} - 1 + \left[\left(\eta^{*-2/k}-1\right)^2 + \left(\eta^{*-2/k}-1\right)\right]^{1/2}.$$

References

Aarts, EHL, de Bont, FMJ., Habers, JHA. and van Laarhoven, PJM. 1986a. A Parallel Statistical Cooling Algorithm. Proc. STACS 86, Springer Lecture Notes in Computer Science, 210, 87-97.

Aarts, EHL., de Bont, FMJ., Habers, JHA. and van Laarhoven, PJM. 1986b. Parallel Implementations of the Statistical Cooling Algorithm. Integration 4: 209-238.

Belisle CJP. 1992. Convergence theorems for a class of simulated annealing algorithms on $\mathbb{R}^d$. Journal of Applied Probability 29: 885 - 895.

Černý V. 1984. Minimization of Continuous Functions by Simulated Annealing, Research Institute for Theoretical Physics, University of Helsinki, preprint No. HU-TFT-84-51.

Černý V. 1985. Thermodynamical approach to the travelling salesman problem: An efficient solution algorithm. Journal of Optimization Theory and Applications 45:41 - 51.

Chopin N. 2002. A sequential particle filter method for static models. Biometrika 89: 539 - 551.

Creal D. 2012. A survey of sequential Monte Carlo methods for economics and finance. Econometric Reviewss 31: 245–296.

Dekkers A, Aarts E. 1991. Global optimization and simulated annealing. Mathematical Programming 50: 367 - 393.

Del Moral P, Doucet A, Jasra A. 2006. Sequential Monte Carlo samplers. Journal of the Royal Statistical Society Series B 68: 411 - 436.

Del Moral P, Doucet A, Jasra A. 2012. On adaptive resampling strategies for sequential Monte Carlo methods. Bernoulli 18: 252-278.

Douc R, Cappe P, Moulines E. 2005. Comparison of resampling schemes for particle filtering. 4th International Symposium on Image and Signal Processing and Analysis `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.126.9118`

Douc R, Moulines E. 2008. Limit theorems for weighted samples with applications to sequential Monte Carlo methods. Annals of Statistics 36: 2344–2376.

Doucet A, Godsill S, Andrieu C. 2000. On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing 10: 197 - 208.

Duan J, Fulop A. Density tempered marginalized sequential Monte Carlo samplers. Journal of Business and Economic Sstatistics 33: 192–202.

Durham G, Geweke J. 2015. Adaptively sequential posterior simulators for massively parallel computing environments. In: Ivan Jeliazkov , Dale J. Poirier (eds.) Bayesian Model Comparison (Advances in Econometrics, Volume 34) Emerald Group Publishing Limited, Chapter 1, 1-44.

Geman S, Hwang CR. 1986. Diffusions for global optimization. SIAM Journal on Control and Optimization 24: 1031–1043.

Geweke J, 1988. The secular and cyclical behavior of real GDP in nineteen OECD countries, 1957-1983. Journal of Business and Economic Statistics 6: 479-486.

Geweke J. 1989. Bayesian inference in econometric models using Monte Carlo integration. Econometrica 57:1317–1340.

Geweke J. 2005. Contemporary Bayesian Econometrics and Statistics. Wiley.

Geweke J. 2015. The secular and cyclical behavior or real GDP: Likelihood-based inference in a nonlinear model using SABL.

Goffe WL, Ferrier GD, Rogers J. (1994). Global optimization of statistical functions with simulated annealing. Journal of Econometrics 60: 65-99

Hamma B, Viitanen S, Törn A. 2000. Parallel continuous simulated annealing for global optimization. Optimization Methods and Software 13: 95-116.

Kirkpatrick S, Gelatt CD, Vecchi MP. 1983. Optimization by simulated annealing. Science 220: 671 - 680.

Liu JS, Chen R. 1998. Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association 93: 1032 - 1044.

Molvalioglu O, Zabinsky ZB, Kohn W. 2007. Models and algorithms for global optimization in: Optimization and its Applications, pp 215-222. New York: Springer.

Molvalioglu O, Zabinsky ZB, Kohn W. 2009. The interaction particle algorithm with dynamic heating and cooling. Journal of Global Optimization 43: 329 - 356.

Samuelson PA (1939). Interactions between the multiplier analysis and the principle of acceleration. The Review of Economics and Statistics 21: 75 - 78.

Vanderbilt, D, Louie SG. 1984. A Monte Carlo simulated annealing approach to optimization over continuous variables. Journal of Computational Physics 36: 259-271.

Xue, G. 1994. Molecular conformation on the CM-5 by parallel two-level simulated annealing. Journal of Global Optimization 4:187-208.

Zhou E, Chen X. 2013. Sequential Monte Carlo simulated annealing. Journal of Global Optimization 55: 101 - 124.

Table 3: Performance of the SA/SMC and SABL optimization algorithms

	Test problem	1	2	3	4	5	6
1	k	2	20	20	20	10	10
2	h^*	$\approxeq -0.998$	-0.01	-1	0	-1	-1e-15
3	x^*	$\approxeq -31.978$	0.00	1	0	0.9	0
	SABL optimization						
4	h^* error	*	2e-19	0	0	0	8.0e-32
5	h^* error bound	2.2e-16	1.7e-18	2.2e-16	2.2e-16	2.2e-16	2.0e-31
6	Max x_i^* range	3.3e-6	5.7e-9	3.3e-9	9.5e-7	1.8e-8	1.4e-16
7	L	33	232	201	139	104	264
8	r_1	4.3e-3	1.5e-9	3.3e-9	9.5e-7	1.8e-8	1.4e-16
9	r_L	2.7e16	3.2e19	7.1e16	8.2e16	2.4e16	6.8e32
10	Evaluations	1.4e7	5.4e7	8.0e7	3.6e7	3.4e7	4.4e7
11	Time	62.78	299.48	230.46	168.66	175.47	217.59
	SA/SMC optimization						
12	# particles	200	200	1000	200	1000	200
13	# cycles	200	2000	5000	5000	4000	20,000
14	Annealing λ	0.1	0.001	0.001	0.001	0.001	0.001
15	Metropolis α	10	10	10	10	10	10
16	Metropolis β	0.995	0.995	0.998	0.998	0.998	0.998
17	Mean h^* error	-1.2e-8	-0.0025	-3.657	-0.0010	-0.446	-0.648
18	Std h^* error	2.8e-8	0.0016	0.768	0.0103	0.386	4.559
19	h^{**} error	<1e-8	-0.0005	-0.0008	-1.1e-7	-0.0002	-3.8e-12
20	Mean x_1^* error	-0.0181	-0.0091	-0.3589	-0.062	0.0205	-0.0003
21	Std x_1^* error	0.0302	0.0703	0.7680	0.628	0.1776	0.0073
22	x_1^{**} error	-0.0011	-0.0105	-0.0004	-0.0001	0.0066	6.6e-8
23	r_L	53	5.9e3	1.6e3	4.6e8	57	9.6e9
24	Evaluations	4.0e6	4.0e7	5.0e8	1.0e8	4.0e8	4.0e8
25	Time	118.32	553.46	1289.1	540.27	1049.5	3011.4

*No exact (analytical) solution available for comparison

Table 4: Posterior moments and maximum likelihood estimates

	Posterior		Direct ML, θ		OLS indirect ML	
Parameter	mean	st. dev.	MLE	st. err.	MLE	st. err.
Secular log (h_s)	4.056	0.667	3.646	0.767	3.646	0.866
Cyclical log (h_c)	-0.584	0.548	-0.096	0.457	-0.096	0.406
Period log (p)	2.045	0.565	1.621	0.148	1.621	0.231
Shock log (σ)	-3.917	0.112	-3.984	0.109	-3.984	0.117