Adaptive Sequential MCMC for Combined State and Parameter Estimation

Zhanglong Cao

Department of Mathematics & Statistics
University of Otago

19 July 2018

GPS Data

GPS units irregularly record time series data of a moving object. These data are in the form of

$$P = \{x_t, y_t, v_t, \omega_t, b_t, \cdots | t \in \mathbb{R}\}. \tag{1}$$

x	longitude
y	latitude
v	velocity
ω	bearing
b	boom status

A *trajectory* is a curve relating (not necessarily connecting) a sequence of successive positions.
Vehicle *tracking system* uses the GPS data to enable users to locate their vehicles with ease and in a convenient manner.

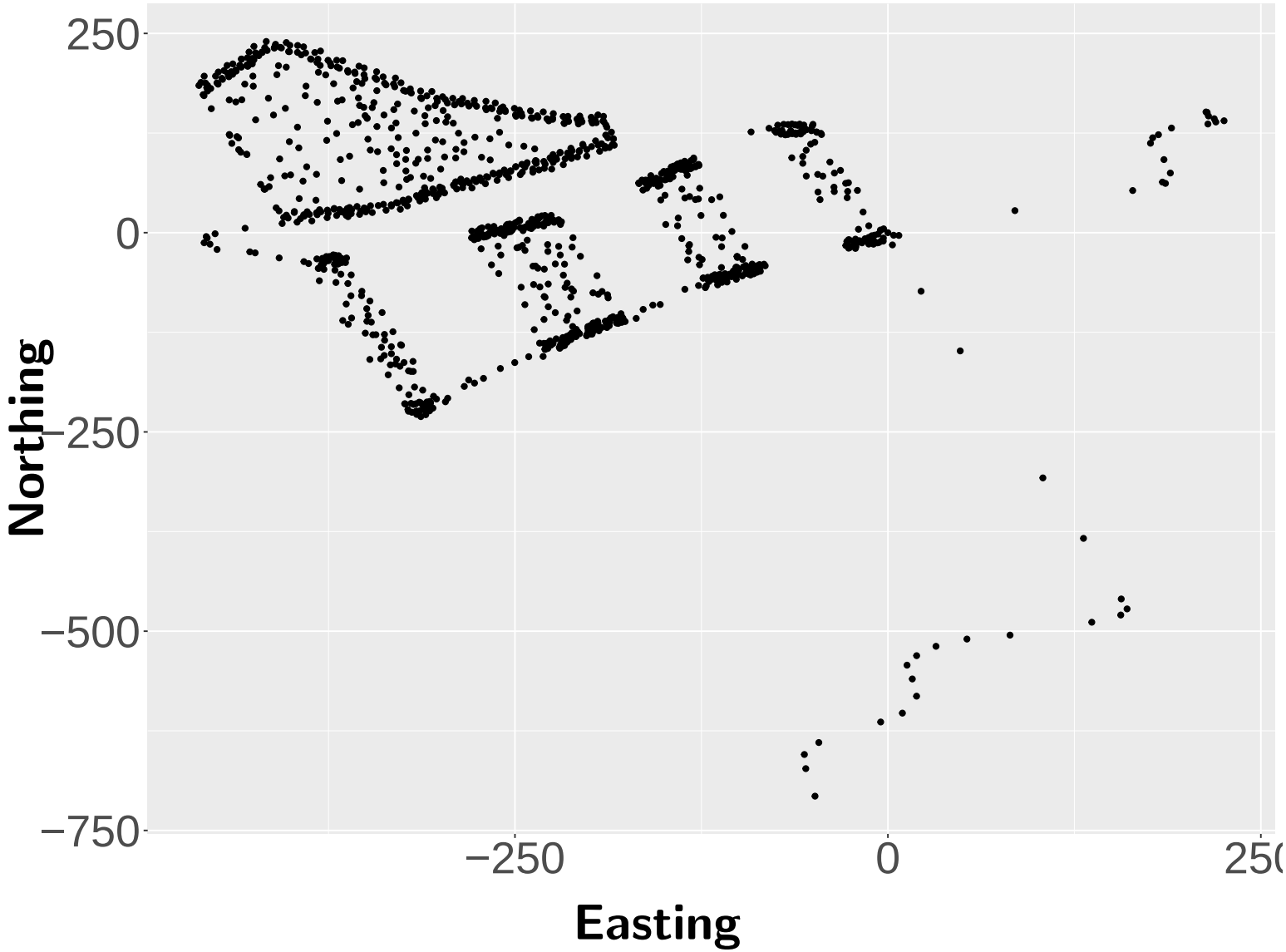


Figure: Examples of GPS data. Observed positions y_t are shown. In trajectory reconstruction, the y_t are combined with velocity information v_t and operating characteristics b_t to infer actual positions x_s, for times of interest s.

Filtering

The word “filtering” refers to the methods for estimating the state of a time-varying system, which is indirectly observed through noisy measurements.
A Bayes filter (Bayesian filter) is a general probabilistic approach to infer an unknown probability density function recursively over time using incoming measurements and a mathematical process model.

In a Hidden Markov model process

$$\begin{array}{ccccccccccc} \cdots & \rightarrow & x_{t-1} & \rightarrow & x_t & \rightarrow & x_{t+1} & \rightarrow & \cdots & & \text{truth} \\ \cdots & & \downarrow & & \downarrow & & \downarrow & & \cdots & & \\ \cdots & & y_{t-1} & & y_t & & y_{t+1} & & \cdots & & \text{observation} \end{array} \tag{2}$$

- Smoothing: $\mathrm{E}[x_k \mid y_{1:t}]$, where $1 \leq k < t$
- Filtering: $\mathrm{E}[x_t \mid y_{1:t}]$
- Prediction: $\mathrm{E}[x_s \mid y_{1:t}]$, where $s > t$

Bayesian Filter

$$
\begin{aligned}
p(x_t \mid y_{1:t}) &\propto p(y_t \mid x_t, y_{1:t-1}) p(x_t \mid y_{1:t-1}) \\
&= p(y_t \mid x_t) p(x_t \mid y_{1:t-1}) \\
&= p(y_t \mid x_t) \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1}
\end{aligned}
\tag{3}
$$

- Prior: The prior $p(x_t \mid y_{1:t-1})$ defines the knowledge of the model
- Likelihood: the likelihood $p(y_t \mid x_t)$ essentially determines the measurement noise model

- Kalman filter
  1. Gaussian
  2. Linear or linearized models
- Particle filter
  1. Arbitrary models (sampling required)
  2. A *sequential Monte Carlo* (SMC) based technique
  3. It is a combination of sequential importance sampling and resampling.

Sequential Monte Carlo

Sequential Monte Carlo method uses Monte Carlo approaches to estimate and to compute recursively. One of the attractive merits is in the fact that they allow on-line estimation by combining the powerful Monte Carlo sampling methods with Bayesian inference at an expense of reasonable computational cost.

Importance Sampling

Recall: to calculate $\int_X f(x)dp(x)$, Monte Carlo uses N i.i.d. samples $\{x^{(1)}, \ldots, x^{(N)}\}$ from $p(x)$

$$\mathrm{E}[f] = \mathrm{E}[\hat{f}] = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}). \tag{4}$$

The idea of importance sampling is to choose a proposal distribution $q(x)$ in place of the true probability distribution $p(x)$, which is hard-to-sample. Rewriting the integration problem as

$$\int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx. \tag{5}$$

Monte Carlo importance sampling is to use N i.i.d. drawn from $q(x)$ to obtain a weighted sum approximation

$$\hat{f} = \frac{1}{N} \sum_{i=1}^{N} w_i f(x^{(i)}), \tag{6}$$

where $w_i = \frac{p(x^{(i)})}{q(x^{(i)})}$ are called the importance weights (or importance ratios). Or

$$\hat{f} = \sum_{i=1}^{N} \tilde{w}_i f(x^{(i)}), \tag{7}$$

if the normalizing factor of $p(x)$ is not known and $\tilde{w}_i = \frac{w_i}{\sum w_i}$.

Sequential Importance Sampling

The importance weights w_t can be updated recursively in the way

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{q(x_t^{(i)} \mid x_{t-1}^{(i)}, y_{1:t})} \tag{8}$$

Issue: degeneracy. After several iterations, only few particles have non-zero importance weights. This phenomenon is called *weight degeneracy* or *sample impoverishment*.
Solution: resampling.

Resampling

The idea of resampling is keeping the same size of particles, replacing the low weights particles with new ones. In the filtering problem,

$$p(x_t \mid y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta \left(x_t - x_t^{(i)} \right). \tag{9}$$

After resampling, it becomes

$$\tilde{p}(x_t \mid y_{1:t}) = \sum_{j=1}^{N} \frac{1}{N} \delta \left(x_t - x_t^{(j)} \right) = \sum_{i=1}^{N} \frac{n_i}{N} \delta \left(x_t - x_t^{(i)} \right), \tag{10}$$

where n_i represents how many times the new particles $x_t^{(j)}$ were duplicated from $x_t^{(i)}$.

State and Parameter Estimation

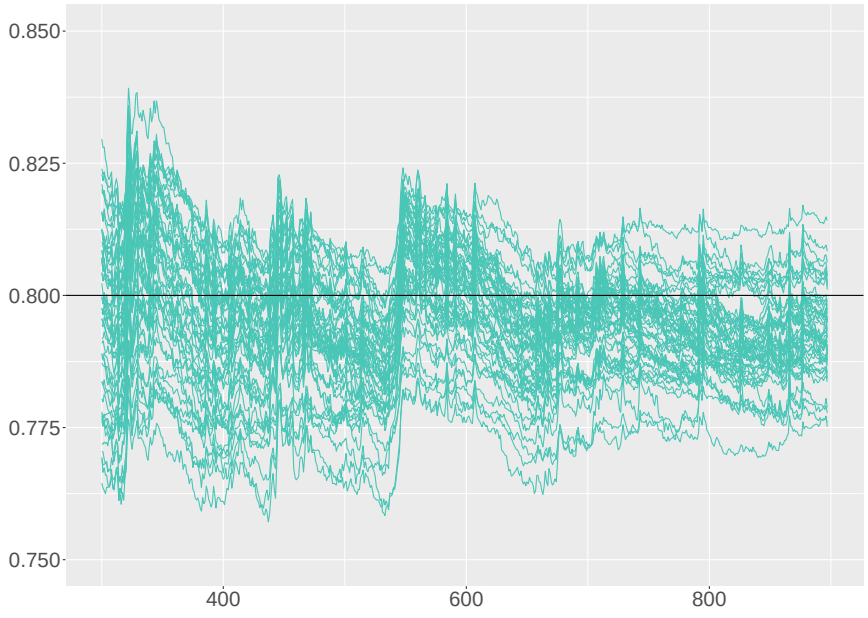
- Liu and West (2001) propose an improved particle filter to kill degeneracy, which is a common issue in static parameter estimation. They use a kernel smoothing approximation, with a correction factor to account for over-dispersion.
- Alternatively, Storvik (2002) propose a new filter algorithm by assuming the posterior depends on a set of sufficient statistics, which can be updated recursively.
- Particle learning approach, introduced by Carvalho (2010), uses sufficient statistics solely to estimate parameters and promises to reduce particle impoverishment. These particle-like methods use more or less sampling and resampling algorithms to update particles recursively.

- Stroud (2018) propose an SMC algorithm by using ensemble Kalman filter framework for high dimensional space models with observations.
- Polson (2008) rely on a fixed-lag length of data approximation to filtering and sequential parameter learning in a general dynamic state-space model.
- An adaptive MCMC method yields a quick and flexible way for estimating posterior distribution in parameter estimation (Haario 1999).

Consider a simple model:

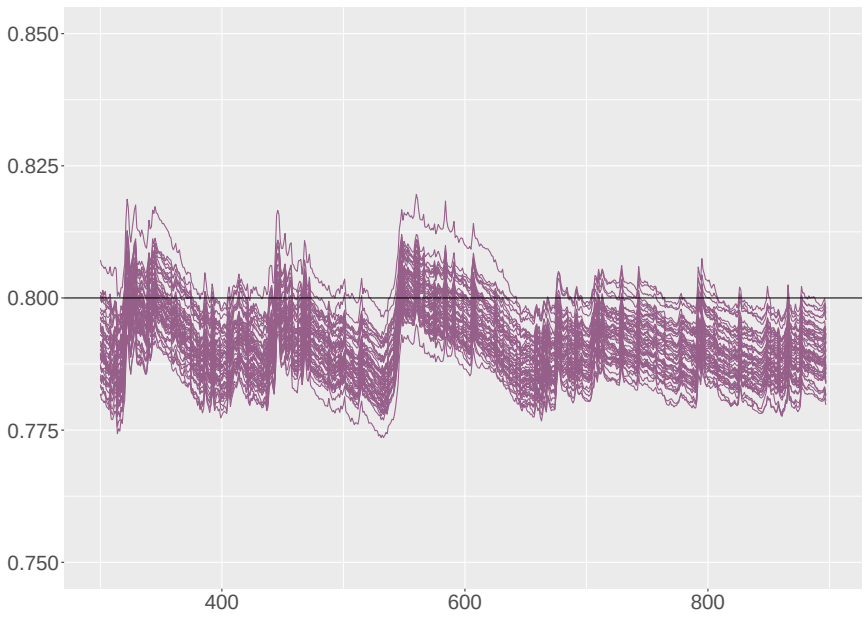
$$\begin{aligned} y_t &= Fx_t + \varepsilon_t, \\ x_t &= \phi x_{t-1} + w_t, \\ x_0 &\sim N(m_0, C_0), \end{aligned} \tag{11}$$

where $\varepsilon_t \sim N\left(0, \sigma^2\right)$ and $w_t \sim N\left(0, \tau^2\right)$, x_t are hidden status and y_t are observations. Assuming that $F = 1$, $\sigma^2 = 1$ and $\tau^2 = 1$. The initial value $x_0 = 0$. ϕ a single static parameter without unobserved state variable.



(a) Liu and West's filter

(b) Storvik Filter



(c) Particle Learning

On-line State and Parameter Estimation

The state transition density and the conditional likelihood function depend not only upon the dynamic state x_t, but also on a static parameter vector θ, which will be stressed by use of the notations $p(x_t \mid x_{t-1}, \theta)$ and $p(y_t \mid x_t, \theta)$.
Putting the algorithms on-line means to update the parameter and state instantly as new observations coming into the data stream.

Visualize Raw Data

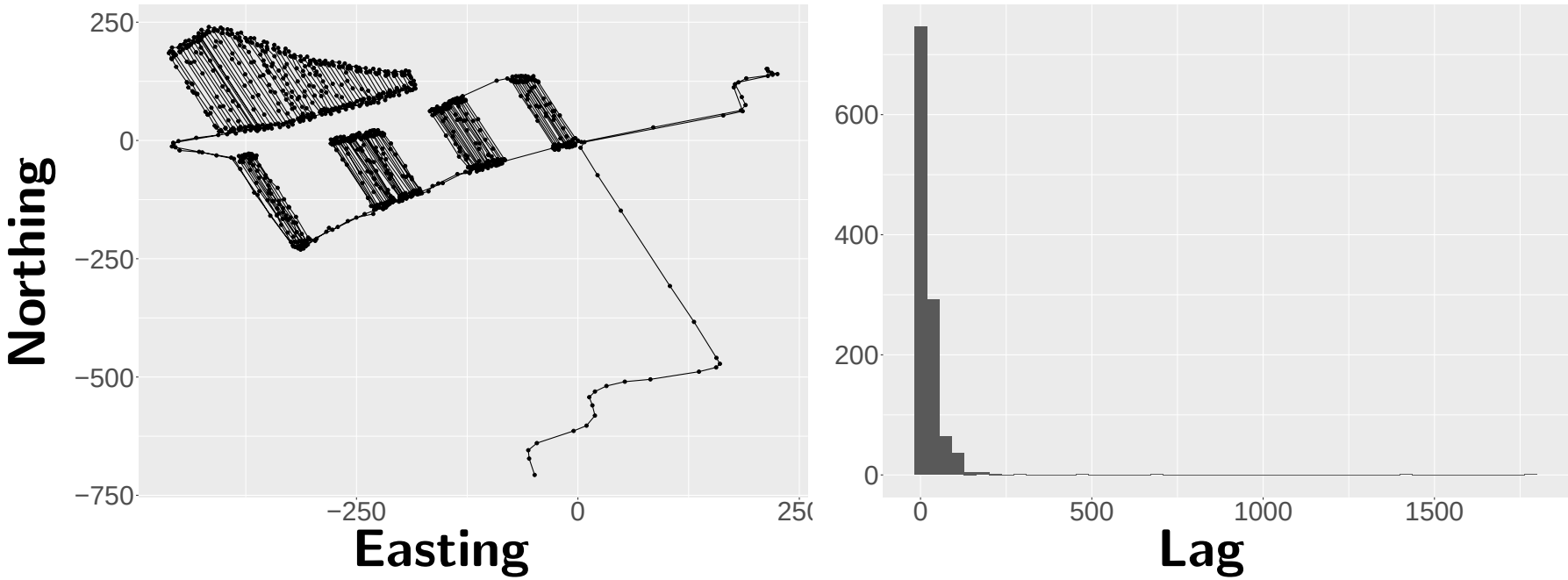


Figure: Demonstration of line-based trajectory of a moving tractor. The time lags (right side figure) obtained from GPS units are irregularly sampled.

Ornstein-Uhlenbeck Process

The forward map for the states is based on an Ornstein-Uhlenbeck process,

$$\begin{cases} \mathrm{d}u_t = -\gamma u_t \mathrm{d}t + \lambda \mathrm{d}W_t', \\ \mathrm{d}x_t = u_t \mathrm{d}t + \xi \mathrm{d}W_t, \end{cases} \tag{12}$$

so that γ^{-1} is roughly the time scale over which the velocity remains informative in the absence of subsequent observations. In our application, $\gamma^{-1} \approx 60\mathrm{s}$.

$$\begin{bmatrix} y_t \\ v_t \end{bmatrix} \sim N\left(\begin{bmatrix} x_t \\ u_t \end{bmatrix}, \begin{bmatrix} \sigma^2 & 0 \\ 0 & \tau^2 \end{bmatrix} \right). \tag{13}$$

Consequently, the parameter θ of an entire Ornstein-Uhlenbeck process is a set of five parameters from both hidden status and observation process, which is represented as $\theta = \{\gamma, \xi^2, \lambda^2, \sigma^2, \tau^2\}$.

Objective Function

If we assume that, given x_{t-1}, x_t is conditionally independent of states at all other times and all observations, then

$$p(x_t \mid y_{1:t}, \theta) = \int p(x_t \mid x_{t-1}, \theta) p(x_{t-1} \mid y_{1:t}, \theta) dx_{t-1}, \tag{14}$$

where $y_{1:t} = \{y_1, \ldots, y_t\}$ is the observation information up to time t. Then, given the posterior distribution $p(\theta \mid y_{1:t})$ for the parameters at time t, we have

$$p(x_t \mid y_{1:t}) = \int p(x_t \mid y_{1:t}, \theta) p(\theta \mid y_{1:t}) d\theta. \tag{15}$$

Generally, researchers would like to find the combined estimation for x_t and θ at time t by

$$p(x_t, \theta \mid y_{1:t}) = p(x_t \mid y_{1:t}, \theta) p(\theta \mid y_{1:t}). \tag{16}$$

Differently, from the target equation (15), the state inference containing N samples is a mixture Gaussian distribution in the following form

$$p(x_t \mid y_{1:t}) = \int p(x_t \mid y_{1:t}, \theta) p(\theta \mid y_{1:t}) d\theta \doteq \frac{1}{N} \sum_{i=1}^{N} p\left(x_t \mid \theta^{(i)}, y_{1:t}\right). \tag{17}$$

Posterior of θ

Additionally, the marginal posterior distribution for the parameter can be written in two different ways:

$$p(\theta \mid y_{1:t}) \propto p(y_{1:t} \mid \theta)p(\theta), \tag{18}$$

$$p(\theta \mid y_{1:t}) \propto p(y_t \mid y_{1:t-1}, \theta)p(\theta \mid y_{1:t-1}). \tag{19}$$

The above formula (18) is a standard Bayesian inference requiring a prior distribution $p(\theta)$. It can be used in off-line methods, in which $\hat{\theta}$ is inferred by iterating over a fixed observation record $y_{1:t}$. By contrast, formula (19) is defined in a recursive way over time depending on the previous posterior at time $t-1$, which is known as on-line method. $\hat{\theta}$ is estimated sequentially as a new observation y_{t+1} becomes available.

Joint Distribution

Rearrange x and u such that $X_{1:t} = \{x_1, u_1, x_2, u_2, \ldots, x_t, u_t\}$, $Y_{1:t} = \{y_1, v_1, y_2, v_2, \ldots, y_t, v_t\}$ and the precision matrix Σ^{-1} looks like

$$\Sigma^{-1} = \begin{bmatrix} A_t & -B_t \\ -B_t^\top & B_t \end{bmatrix}, \tag{20}$$

where B_t is a $2t \times 2t$ diagonal matrix

$$B_t = \begin{bmatrix} \frac{1}{\sigma^2} & \cdot & \cdot & \cdot & \cdot \\ \cdot & \frac{1}{\tau^2} & \cdot & \cdot & \cdot \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \cdot & \cdot & \cdot & \frac{1}{\sigma^2} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \frac{1}{\tau^2} \end{bmatrix}. \tag{21}$$

Posterior Distribution

The covariance matrix is the inverse of the precision matrix

$$\Sigma_t = \begin{bmatrix} (A_t - B_t)^{-1} & (A_t - B_t)^{-1} \\ (A_t - B_t)^{-1} & \left(I_t - A_t^{-1} B_t\right)^{-1} B_t^{-1} \end{bmatrix} \triangleq \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}. \tag{22}$$

Therefore, the posterior distribution of θ is

$$p\left(\theta \mid y_{1:t}\right) \propto p\left(y_{1:t} \mid \theta\right) p\left(\theta\right) \propto \exp\left(-\frac{1}{2} y_{1:t} \Sigma_{YY}^{-1} y_{1:t}\right) \sqrt{\det \Sigma_{YY}^{-1}}\, p\left(\theta\right). \tag{23}$$

The Forecast Distribution

It is known that

$$p(Y_{1:t-1}, \theta) = N\left(0, \Sigma_{YY}^{(t-1)}\right) \tag{24}$$

$$p(Y_t, Y_{1:t-1}, \theta) = N\left(0, \Sigma_{YY}^{(t)}\right) \tag{25}$$

$$p(Y_t \mid Y_{1:t-1}, \theta) = N\left(\bar{\mu}_t, \bar{\Sigma}_t\right) \tag{26}$$

where the covariance matrix of the joint distribution is $\Sigma_{YY}^{(t)} = \left(I_t - A_t^{-1} B_t\right)^{-1} B_t^{-1}$.
To be clear, the matrix B_t is short for the matrix $B_t(\sigma^2, \tau^2)$, which is $2t \times 2t$ diagonal matrix with elements $\frac{1}{\sigma^2}, \frac{1}{\tau^2}$ repeating for t times on its diagonal.

By using the Sherman-Morrison-Woodbury formula and after massive calculations, K_t and b_t are updated in a recursive way.
Thus

$$\begin{aligned} \bar{\mu}_t &= \Phi_t K_{t-1} B_1 \bar{\mu}_{t-1} + \Phi_t \left(I_t - K_{t-1} B_1\right) Y_{t-1} \\ \bar{\Sigma}_t &= \left(B_1 K_t B_1\right)^{-1} \end{aligned} \tag{27}$$

where $K_1 = B_1^{-1} - A_1^{-1} = \begin{bmatrix} \frac{\sigma^4}{\sigma^2 + L_x^2} & 0 \\ 0 & \frac{\tau^4}{\tau^2 + L_u^2} \end{bmatrix}$ and $\Phi_t = \begin{bmatrix} 1 & \frac{1 - e^{-\gamma \Delta_t}}{\gamma} \\ 0 & e^{-\gamma \Delta_t} \end{bmatrix}$ is the forward map.

The Estimation Distribution

The joint distribution of X_t and $Y_{1:t}$ is

$$X_t, Y_{1:t} \mid \theta \sim N\left(0, \begin{bmatrix} C_t^\top (A_t - B_t)^{-1} C_t & C_t^\top (A_t - B_t)^{-1} \\ (A_t - B_t)^{-1} C_t & \left(I_t - A_t^{-1} B_t\right)^{-1} B_t^{-1} \end{bmatrix}\right), \quad (28)$$

where $C_t^\top = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$ is a $2 \times 2t$ matrix. Thus,

$$X_t \mid Y_{1:t}, \theta \sim N\left(\mu_t^{(X)}, \Sigma_t^{(X)}\right), \quad (29)$$

recursively through

$$\mu_t^{(X)} = K_t B_1 \bar{\mu}_t + (I_t - B_1 K_t) Y_t \quad (30)$$

$$\Sigma_t^{(X)} = B_1^{-1} - K_t. \quad (31)$$

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) refers to constructing Markov chains that move in the unobserved quantity space and produce a sequence samples from the posterior distribution. After the chain has been run long enough, the sequence is considered as an approximation to the posterior distribution.

Metropolis-Hastings Algorithm

Given essentially a probability distribution $\pi(\cdot)$ (the target distribution), MH algorithm provides a way to generate a Markov chain $x_1, x_2, \ldots, x_N$, who has the target distribution as a stationary distribution, for the uncertain parameters x requiring only that this density can be calculated at x.

Suppose that we can evaluate $\pi(x)$ for any x. The transition probabilities should satisfy the detailed balance condition

$$\pi\left(x'\right) q\left(x', x^{(t)}\right) \alpha\left(x', x^{(t)}\right) = \pi\left(x^{(t)}\right) q\left(x^{(t)}, x'\right) \alpha\left(x^{(t)}, x'\right). \quad (32)$$

In this way, a tentative new state x' is generated from the proposal density $q\left(x'; x^{(t)}\right)$ and it is accepted or rejected according to acceptance probability

$$\alpha = \frac{\pi\left(x'\right)}{\pi\left(x^{(t)}\right)} \frac{q\left(x^{(t)}, x'\right)}{q\left(x', x^{(t)}\right)}. \quad (33)$$

If $\alpha \geq 1$, the new state is accepted. Otherwise, the new state is accepted with probability α.

Random Walk

In a random walk, the proposal density function $q(\cdot)$ can be chosen for some suitable normal distribution, and hence $q\left(x' \mid x^{(t)}\right) = N\left(x' \mid x^{(t)}, \epsilon^2\right)$ and $q\left(x^{(t)} \mid x'\right) = N\left(x^{(t)} \mid x', \epsilon^2\right)$ cancel in the above equation (33).
To decide whether to accept the new state, we compute the the probability of accepting the new state by

$$\alpha = \min\left\{1, \frac{\pi\left(x'\right) q\left(x^{(t)} \mid x'\right)}{\pi\left(x^{(t)}\right) q\left(x' \mid x^{(t)}\right)}\right\} = \min\left\{1, \frac{\pi\left(x'\right)}{\pi\left(x^{(t)}\right)}\right\}. \tag{34}$$

If the proposed value is accepted it becomes the next current value $x^{(t+1)} = x'$; otherwise the current value is left unchanged $x^{(t+1)} = x^{(t)}$.

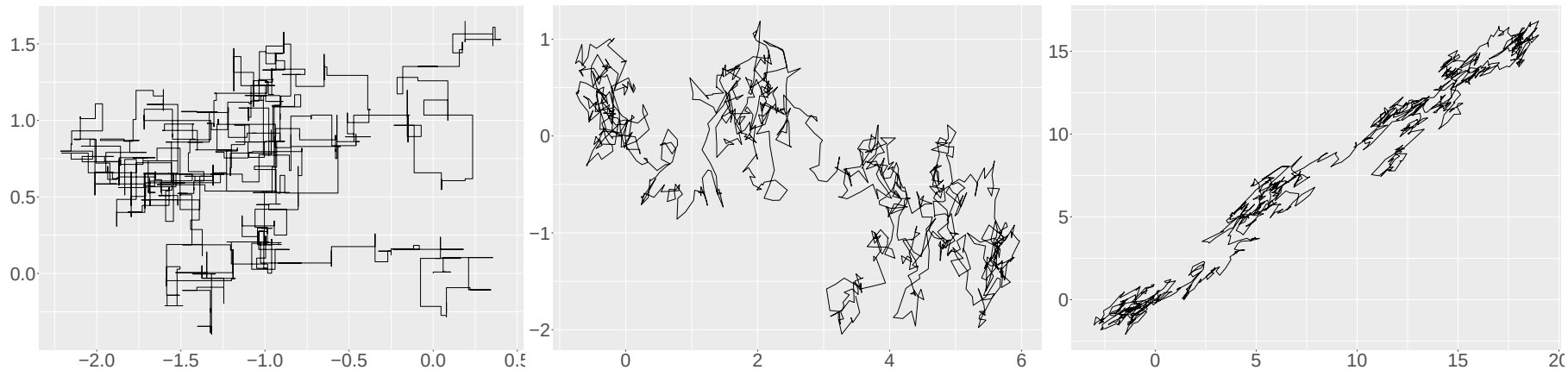
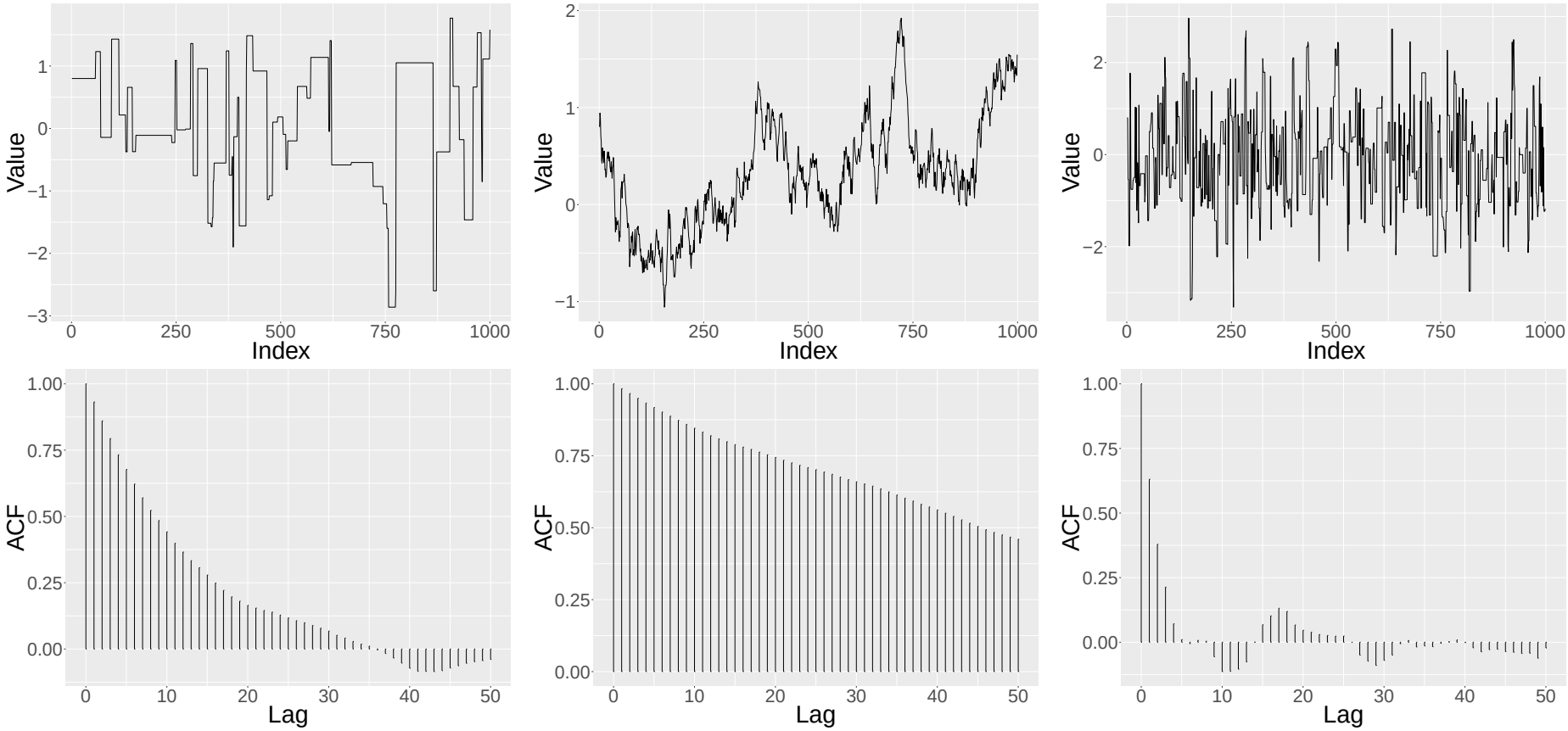


Figure: Examples of 2-Dimensional random walk Metropolis-Hastings algorithm.

Plenty of work has obtained many interesting results for the d-dimensional spherical multivariate normal problem with symmetric proposal distributions, including that the optimal scale is approximately $2.4/\sqrt{d}$ times the scale of target distribution, which implies optimal acceptance rates of 0.44 for $d = 1$ and 0.23 for $d \to \infty$.



(a) a large step size (b) a small step size (c) a proper step size

Figure: Metropolis-Hastings sampler for a single parameter with: 5a a large step size, 5b a small step size, 5c an appropriate step size.

Self-tuning Random Walk

Algorithm 1: Self-tuning Random Walk Metropolis-Hastings Algorithm

1 Initialization: Given an arbitrary positive step size $s_i^{(1)}$ for each parameter. Set up a value for b and find a by using the formula. Set up a target acceptance rate α_i for each parameter, where $i = 1, \ldots, d$.

2 Run sampling algorithm: **for** *k from 1 to N* **do**

3 Randomly select a parameter $\theta_i^{(k)}$, propose a new one by $\theta_i' \sim N\left(\theta_i^{(k)}, \epsilon s_i^{(k)}\right)$ and leave the rest unchanged.

4 Accept θ_i' with probability $\alpha = \min\left\{1, \frac{\pi(\theta')q\left(\theta^{(k)},\theta'\right)}{\pi\left(\theta^{(k)}\right)q\left(\theta',\theta^{(k)}\right)}\right\}$.

5 If it is accepted, tune step size to $s_i^{(k+1)} = s_i^{(k)} e^a$, otherwise $s_i^{(k+1)} = s_i^{(k)} / e^b$.

6 Set $k = k + 1$ and move to step 3 until N.

7 Take n samples out from N with equal spaced index for each parameter being a new sequence.

Delayed-Acceptance Metropolis-Hastings Sampling

Cui, T., Fox, C., & O'Sullivan, M. J. (2011) proposed a two-stage Metropolis-Hastings algorithm in which, typically, proposed parameter values are accepted or rejected at the first stage based on a computationally cheap surrogate $\hat{\pi}(x)$ for the likelihood $\pi(x)$. In stage one, the quantity α_1 is found by a standard MH acceptance formula

$$\alpha_1 = \min\left\{1, \frac{\hat{\pi}(x')q\left(x^{(t)}, x'\right)}{\hat{\pi}(x^{(t)})q\left(x', x^{(t)}\right)}\right\}, \tag{35}$$

where $\hat{\pi}(\cdot)$ is a cheap estimation for x and a simple form is $\hat{\pi}(\cdot) = N\left(\cdot \mid \hat{x}, \epsilon\right)$. Once stage one is accepted with α_1, the process goes into stage two and the acceptance probability α_2 is

$$\alpha_2 = \min\left\{1, \frac{\pi(x')\hat{\pi}\left(x^{(t)}\right)}{\pi\left(x^{(t)}\right)\hat{\pi}(x')}\right\}, \tag{36}$$

where the overall acceptance probability $\alpha_1\alpha_2$ ensures that detailed balance is satisfied with respect to $\pi(\cdot)$; however if a rejection occurs at stage one, the expensive evaluation of $\pi(x)$ at stage two is unnecessary.

Efficiency

To explore the efficiency of a MCMC process, it is suggested that the efficiency of Markov chains (Eff) is

$$e_g(\sigma) \propto \left(\tau_g \mathrm{Var}_\pi[g(X)]\right)^{-1}. \tag{37}$$

where τ_g *integrated autocorrelation time*.
However, the disadvantage of their method is that the measurement of efficiency is highly dependent on the function g. Instead, an alternative approach uses *effective sample size* (ESS) which is defined in in the following form of

$$\mathrm{ESS} = \frac{n}{1 + 2\sum_{k=1}^{\infty} \rho_k(X)} \approx \frac{n}{1 + 2\sum_{k=1}^{k_{\mathrm{cut}}} \rho_k(X)} = \frac{n}{\tau}, \tag{38}$$

Functions IAT and ESS in the package LaplacesDemon are available in *R* (Christen, J.A. and Fox, C. (2010)).

Moreover, a wide support among both statisticians and physicists use the following cost of independent samples to evaluate the performance of MCMC, that is

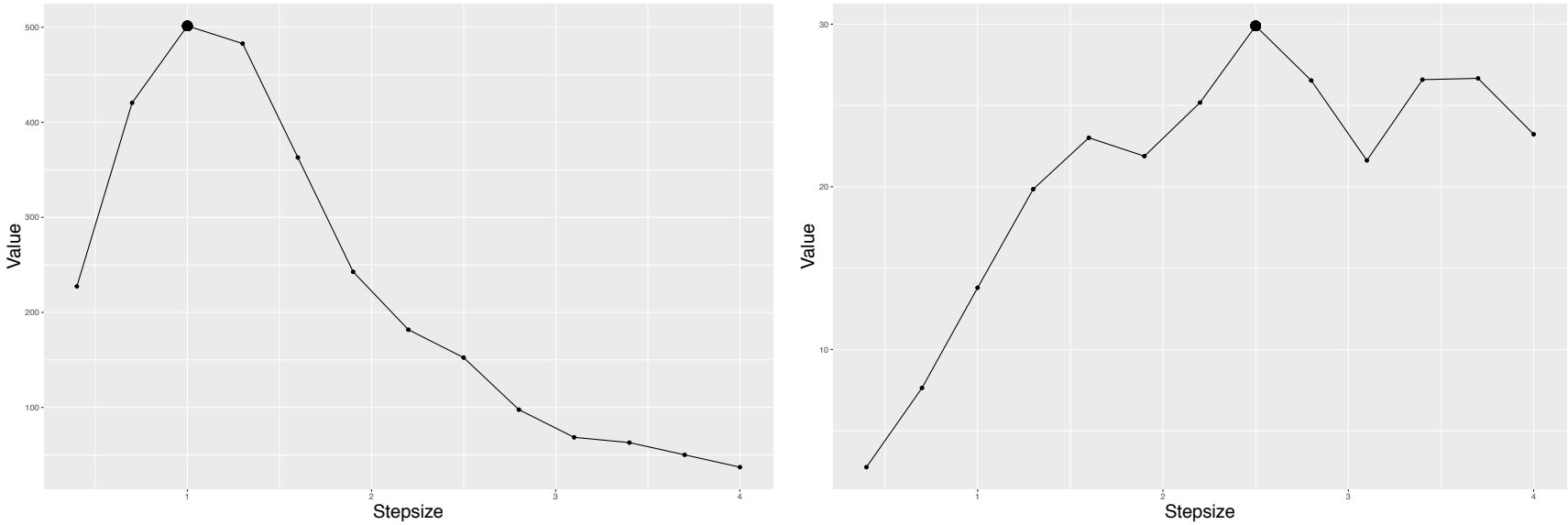
$$\frac{n}{\text{ESS}} \times \text{cost per step} = \tau \times \text{cost per step}. \tag{39}$$

Being inspired by their research, we now define the Efficiency in Unit Time (EffUT) and ESS in Unit Time (ESSUT) as follows:

$$\text{EffUT} = \frac{e_g}{T}, \tag{40}$$

$$\text{ESSUT} = \frac{\text{ESS}}{T}, \tag{41}$$

where T represents the computation time, which is also known as running time.



(a) ESS against different step sizes (b) ESSUT against different step sizes

Figure: Influences of different step sizes on effective sample size (ESS) and effective sample size in unit time (ESSUT)

Step Size	Length	Time	Eff	EffUT	ESS	ESSUT
1.0	1 000	3.48	0.0619	0.0178	69.4549	19.9583
1.3	1 400	3.40	0.0547	0.0161	75.3706	22.1678
	1 000*	3.40	0.0813	0.0239	72.5370	21.3344
2.2	5 000	3.31	0.0201	0.0061	96.6623	29.2031
	1 000*	3.31	0.0941	0.0284	94.2254	28.4669
2.5	7 000	3.62	0.0161	0.0044	112.3134	31.0258
	1 000*	3.62	**0.1095**	**0.0302**	**113.4063**	**31.3277**

Table: Comparison of Eff, EffUT, ESS and ESSUT values with different step size. The 1000* means taking 1 000 samples from a longer chain, such as taking 1 000 out of 5 000 sample chain. The computation time is measured in seconds.

Application to Real GPS Data

An application is to track the position of a moving tractor on a farm. The original GPS data set is plotted in Figure 1. In a 2-dimensional trajectory filtering problem, we use the same parameter $\theta = \{\gamma, \xi^2, \lambda^2, \sigma^2, \tau^2\}$ for both easting and northing directions. The observations on these two directions are denoted as Y_E and Y_N respectively. The hidden states on easting and northing directions are X_E and X_N.

Recall that:

$$p(X_t \mid Y_{1:t}) = \int p(X_t \mid Y_{1:t}, \theta) p(\theta \mid Y_{1:t}) d\theta \doteq \frac{1}{N} \sum_{i=1}^{N} p\left(X_t \mid \theta^{(i)}, Y_{1:t}\right). \tag{42}$$

and

$$p(\theta \mid Y_{1:t}) \propto p(Y_{1:t} \mid \theta) p(\theta), \tag{43}$$

$$p(\theta \mid Y_{1:t}) \propto p(Y_t \mid Y_{1:t-1}, \theta) p(\theta \mid Y_{1:t-1}). \tag{44}$$

Learning Phase

In the learning phase, a cheap Gaussian surrogate $\hat{p}(\theta)$ is obtained that will be used for a *delayed-acceptance Metropolis-Hastings* sampler in the estimation phase.
Specifically, a self-tuning random walk Metropolis-Hastings algorithm, in which the parameters are updated one at a time, is used to obtain the mean and covariance structure of $\hat{p}(\theta)$

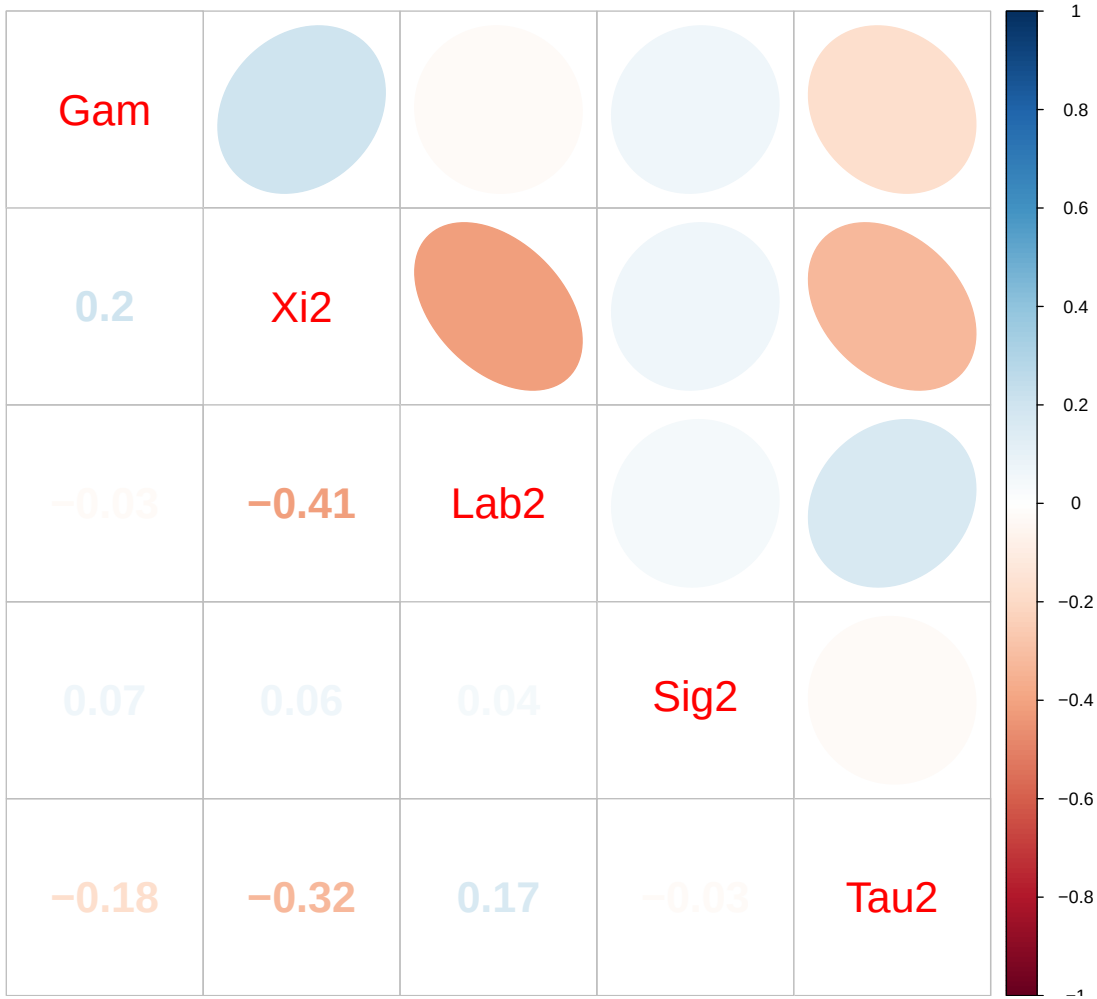
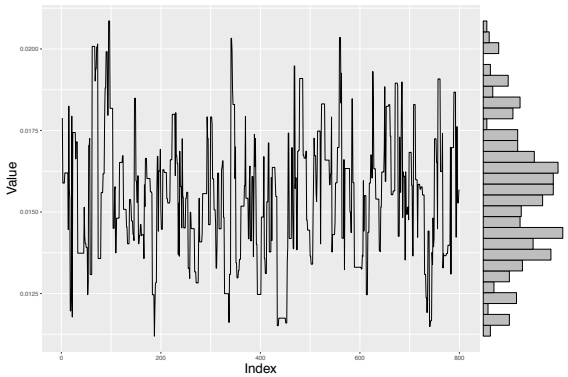
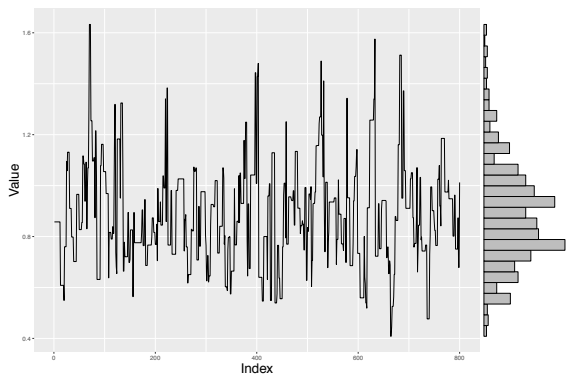


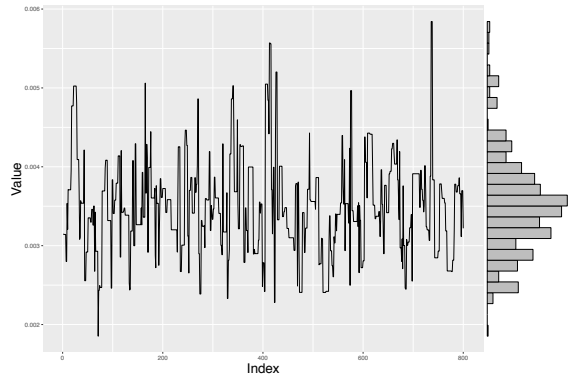
Figure: Visualization of the parameters correlation matrix, which is found in learning phase.



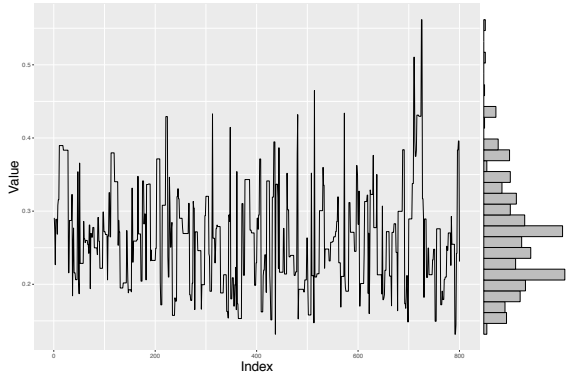
(a) Trace plot of γ



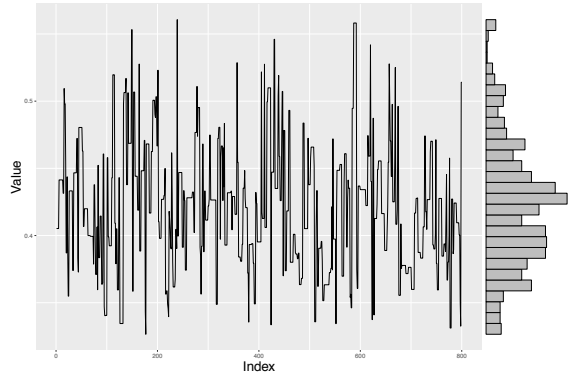
(b) Trace plot of ξ^2



(c) Trace plot of λ^2



(d) Trace plot of σ^2



(e) Trace plot of τ^2

Figure: Trace plots of θ after taking 1 000 burn-in samples out from 5 000 from the learning phase.

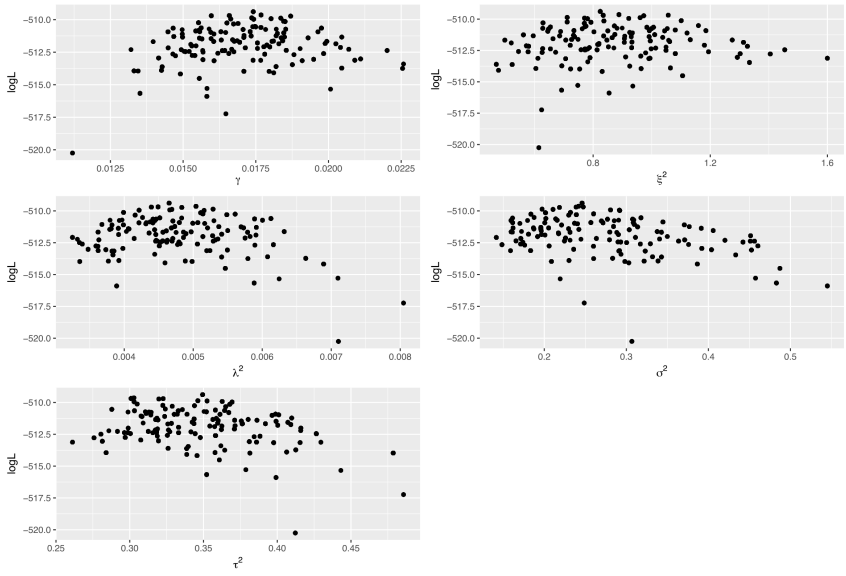
Estimation Phase

To maintain sampling speed in on-line mode, we use all the latest date with truncating the first few history ones

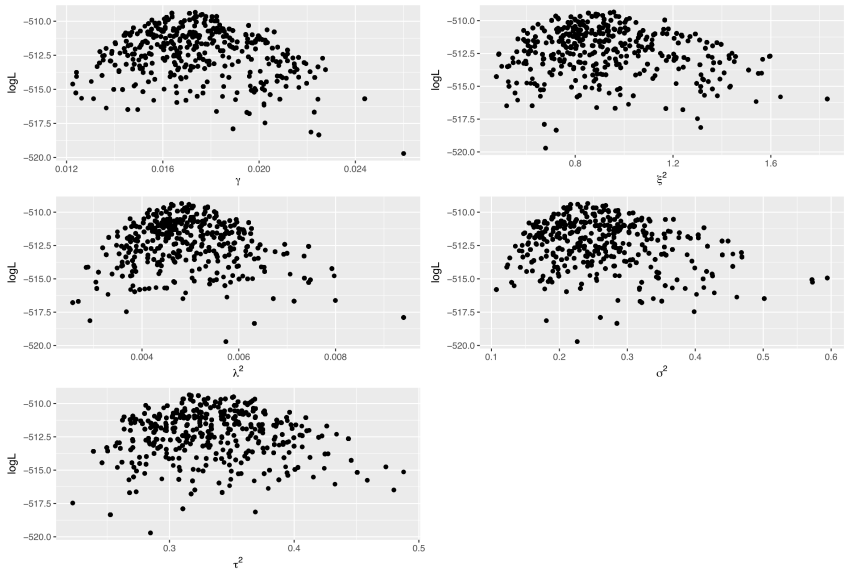
$$p(X_t \mid Y_{t-L+1:t}) \doteq \frac{1}{N} \sum_{i=1}^{N} p(X_t \mid Y_{t-L+1:t}, \theta^{(i)}). \tag{45}$$

We name this method the *Sliding Window Sequential Parameter Learning Filter*. This is also advantageous in our application as the parameters are often slowly varying in time.

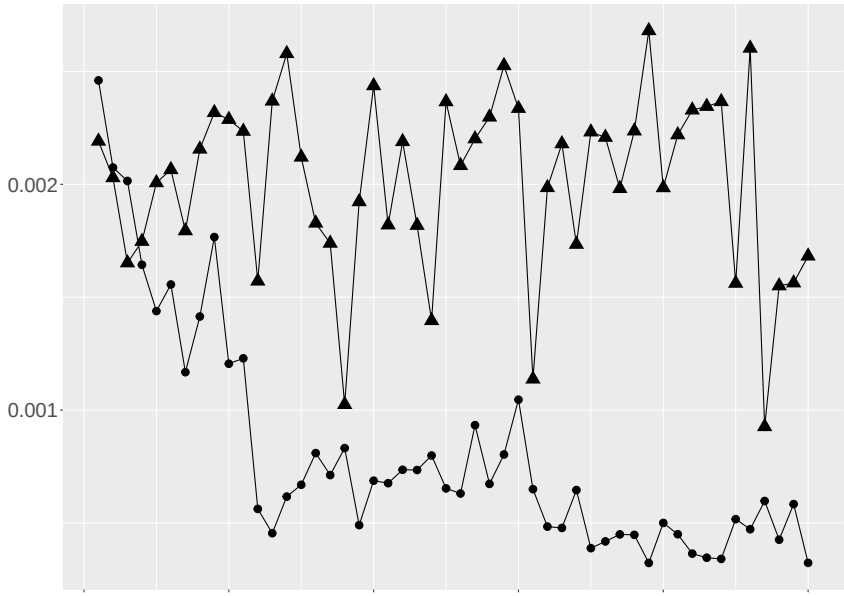
To avoid estimation bias, which is caused by sampling degeneracy, we are introducing a *threshold* criterion and a *cutting-off* value. In a certain circumstance, the cheap $\hat{\pi}(\cdot)$ is not accurate and is replaced by a new one $\hat{\pi}_{\text{new}}(\cdot)$.
The *cutting-off* procedure stops the algorithm when a large Δ_t occurs in the progress. A large time gap indicates a break of the vehicle at a time point and it causes irregularity and bias. A smart way is to stop the process and to wait for new data coming in.



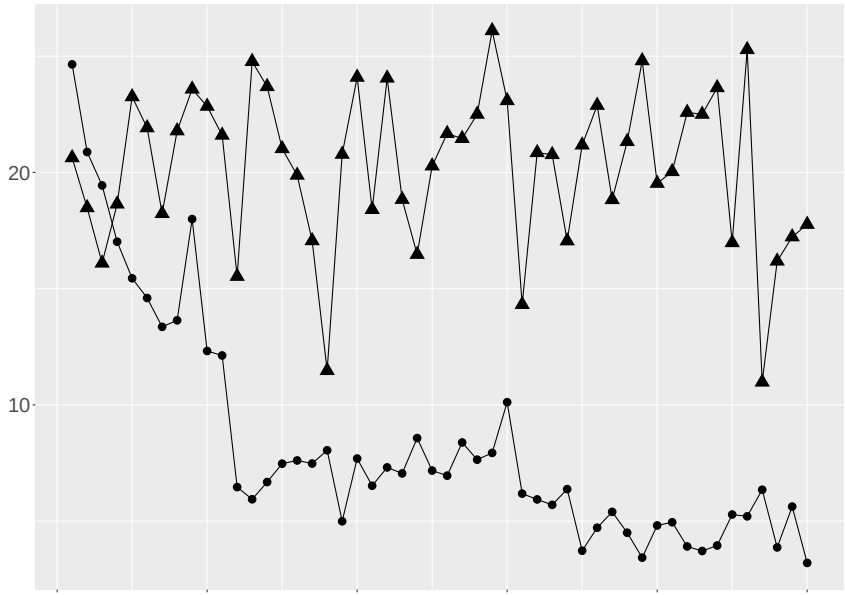
(a) ln L surfaces: not-updating-mean



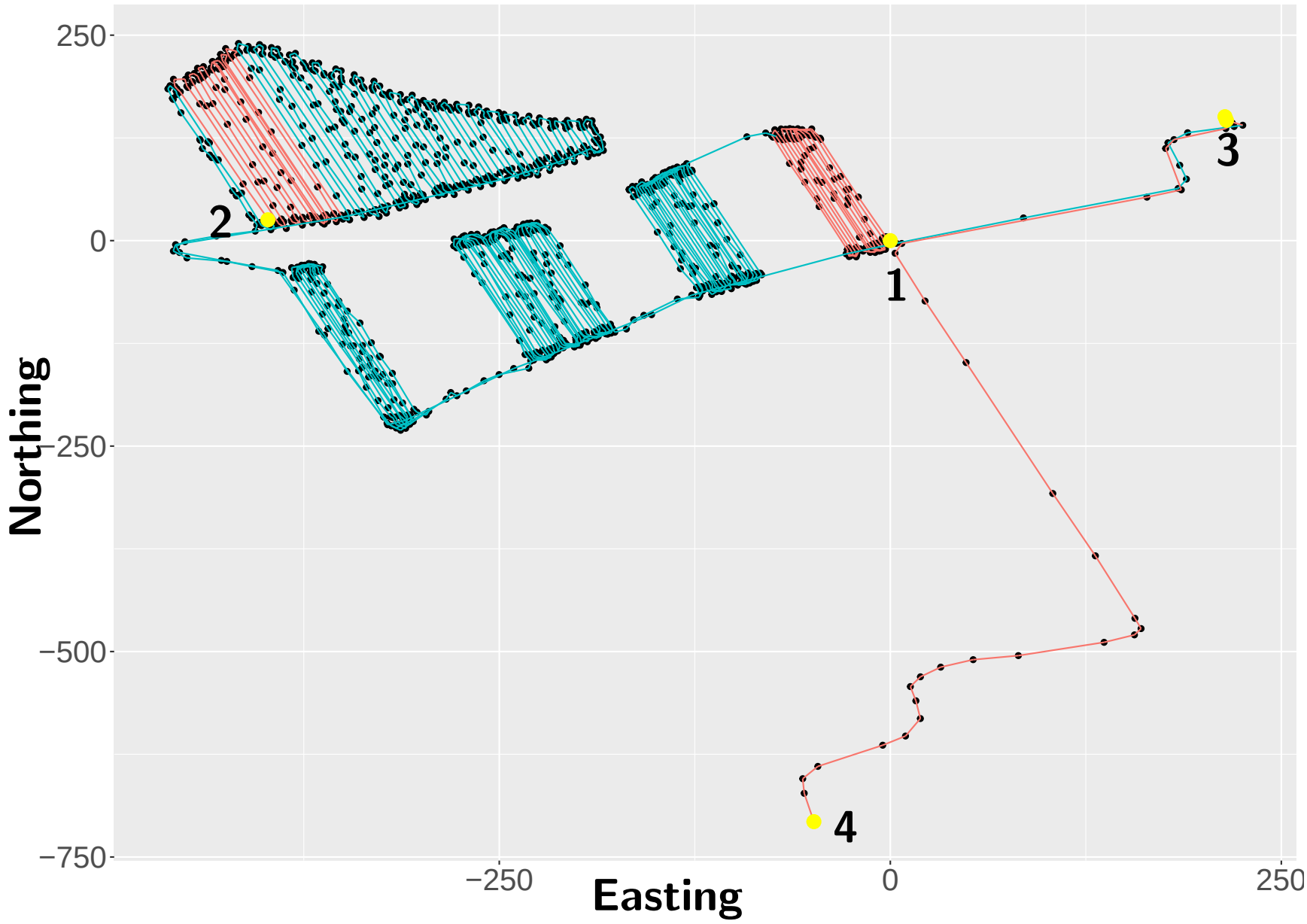
(b) ln L surfaces: updating-mean



(a) Comparing EffUT



(b) Comparing ESSUT


250
0
−250
−500
−750
Northing
−250
0
250
Easting
1
2
3
4

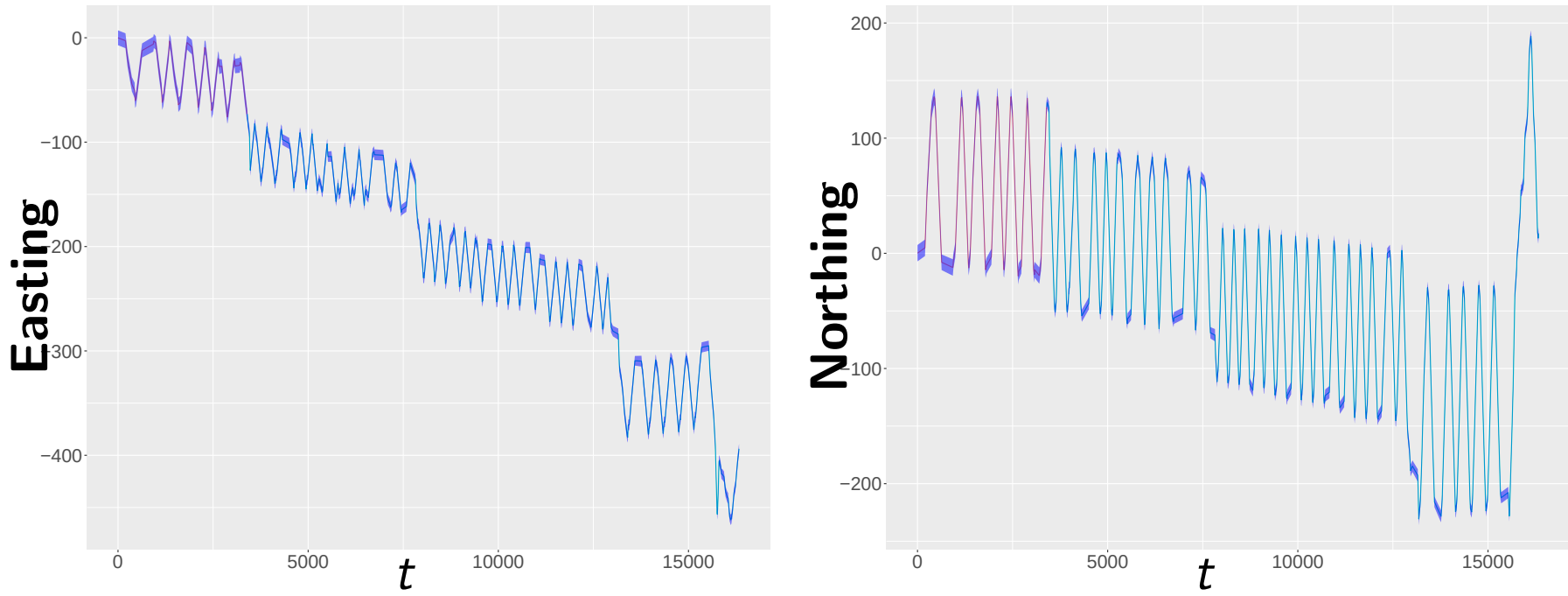


Figure: Uncertainties on easting and northing directions before the first cutting-off procedure. The means of uncertainties on each direction are about 0.5 meters.

Further video at:
Adaptive MCMC

Discussion

1. An adaptive MCMC algorithm is proposed for estimating combined state and parameter in a homogeneous linear state-space model (OU-process).
2. The whole process is split into two phases: learning phase and estimation phase.
3. In on-line mode, the algorithm is adaptive to maintain sampling efficiency and uses a sliding window approach to maintain sampling speed.
4. Advantages: easy to understand and to implement in practice. In contrast, Particle Learning algorithm is highly efficient, however, the sufficient statistics are not available at all times.

Future Work

- Boom-dependent sampling proposal
- Speeding up MCMC by parallel computation (such as population MCMC) and data sub-sampling in high dimensional space
- Bayesian inference for intractable distribution $p(y \mid \theta)$

Acknowledgment

- Ministry of Business, Innovation & Employment
  – Grant UOOX 1208
- TracMap
- Dr Matthew Parry
- Prof David Bryant

References

Anderson, J. L. (2001).

An ensemble adjustment Kalman Filter for data assimilation.

Stroud, J. R., Katzfuss, M., and Wikle, C. K. (2018)

A Bayesian adaptive ensemble Kalman filter for sequential state and parameter estimation.

Chen, Z. (2003).

Bayesian Filtering: from Kalman filters to particle filters, and beyond.

Andrieu, C., Doucet, A., and Tadic, V. B. (2005).

On-line parameter estimation in general state-space models.

Christen, J. A. and Fox, C. (2005).

Markov chain Monte Carlo using an approximation.

Christen, J. A. and Fox, C. (2010).

A general purpose sampling algorithm for continuous distributions (the t-walk).

Cappé, O., Godsill, S. J., and Moulines, E. (2007).

An overview of existing methods and recent advances in sequential Monte Carlo.

References

Liu, J. and West, M. (2001).

Combined parameter and state estimation in simulation based filtering.

Storvik, G. (2002).

Particle filters for state-space models with the presence of unknown static parameters.

Lopes, H. F. and Tsay, R. S. (2011).

Particle filters and Bayesian inference in financial econometrics.

Andrieu, C. and Thoms, J. (2008).

A tutorial on adaptive MCMC. Statistics and Computing

Polson, N. G., Stroud, J. R., and Müller, P. (2008).

Practical filtering with sequential parameter learning.

Kantas, N., Doucet, A., Singh, S. S., and Maciejowski, J. M. (2009).

An overview of sequential Monte Carlo methods for parameter estimation in General State-Space Models.

Quiroz, M., Tran, M.-N., Villani, M., and Kohn, R. (2018).

Speeding up MCMC by delayed acceptance and data subsampling.