Nearly Smooth Particle Filters for Likelihood Estimation with Multivariate Latent Variables

Anthony Lee

Department of Statistics & Oxford-Man Institute of Quantitative Finance
University of Oxford

Greek Stochastics Meeting, August 2009

Joint work with Arnaud Doucet, ISM

State-Space Models

- We focus on time-homogeneous Markovian state-space models with hidden states:
  - $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$, each $\mathbf{x}_t \in \mathcal{X}$

  and observations:
  - $\mathbf{y}_{0:T} = \{\mathbf{y}_0, \ldots, \mathbf{y}_T\}$, each $\mathbf{y}_t \in \mathcal{Y}$
- The model is given by

$$
\begin{array}{lll}
p(\mathbf{x}_0|\theta) & & \text{(initial state)} \\
p(\mathbf{x}_t|\mathbf{x}_{t-1}, \theta) & \text{for } 1 \leq t \leq T & \text{(evolution)} \\
p(\mathbf{y}_t|\mathbf{x}_t, \theta) & \text{for } 0 \leq t \leq T & \text{(observation)}
\end{array}
$$

  where $\theta \in \Theta$ are the parameters of the model.

Likelihood Evaluation

- Given data $\mathbf{y}_{0:T}$, we want to evaluate $p(\mathbf{y}_{0:T}|\theta)$ for any $\theta \in \Theta$.
- This is not straightforward in general. While we can compute $p(\mathbf{x}_{0:T}, \mathbf{y}_{0:T}|\theta)$,
  - $p(\mathbf{y}_{0:T}|\theta)$ usually cannot be computed analytically.
  - $p(\mathbf{y}_{0:T}|\theta) = \int_{\mathcal{X}^{T+1}} p(\mathbf{x}_{0:T}, \mathbf{y}_{0:T}|\theta) d\mathbf{x}_{0:T}$ is a very high-dimensional integral.
- Difficult to find good proposal densities $q(\mathbf{x}_{0:T}|\mathbf{y}_{0:T}, \theta)$.

Likelihood Decomposition

- We can decompose the likelihood $p(\mathbf{y}_{0:T}|\theta)$ as follows:

$$p(\mathbf{y}_{0:T}|\theta) = p(\mathbf{y}_0|\theta)\prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{y}_{0:t-1},\theta)$$

where

$$\begin{aligned}
p(\mathbf{y}_t|\mathbf{y}_{0:t-1},\theta) &= \int p(\mathbf{y}_t,\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1},\theta)d\mathbf{x}_{t-1} \\
&= \int p(\mathbf{y}_t|\mathbf{x}_{t-1},\theta)p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1},\theta)d\mathbf{x}_{t-1} \\
&= \int\int p(\mathbf{y}_t,\mathbf{x}_t|\mathbf{x}_{t-1},\theta)d\mathbf{x}_t p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1},\theta)d\mathbf{x}_{t-1} \\
&= \int\int p(\mathbf{y}_t|\mathbf{x}_t,\theta)p(\mathbf{x}_t|\mathbf{x}_{t-1},\theta)d\mathbf{x}_t p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1},\theta)d\mathbf{x}_{t-1}
\end{aligned}$$

Likelihood Decomposition II

- If we can sample $\mathbf{x}_{t-1}^{(i)} \sim p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1}, \theta)$ and $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \theta)$ for $i = 1, \ldots, N$ we can estimate $p(\mathbf{y}_t|\mathbf{y}_{0:t-1}, \theta)$ via

$$\hat{p}_N(\mathbf{y}_t|\mathbf{y}_{0:t-1}, \theta) = \frac{1}{N}\sum_{i=1}^{N} p(\mathbf{y}_t|\mathbf{x}_t^{(i)}, \theta)$$

- and $\log p(\mathbf{y}_{0:T}|\theta)$ via

$$\hat{\ell}_N(\mathbf{y}_{0:T}|\theta) = \log \hat{p}_N(\mathbf{y}_0|\theta) + \sum_{t=1}^{T} \log \hat{p}_N(\mathbf{y}_t|\mathbf{y}_{0:t-1}, \theta)$$

- SMC will allow us to sample from an empirical distribution $\hat{P}_N(d\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1}, \theta)$ which approximates $p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1}, \theta)$.

Sequential Monte Carlo for State-Space Models

1. At time $t = 0$ (same except for sampling density)...
2. For times $t > 0$.
   - For $i = 1, \ldots, N$, sample $\tilde{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t | \mathbf{y}_t, \mathbf{x}_{t-1}^{(i)})$ and set $\tilde{\mathbf{x}}_{0:t}^{(i)} \stackrel{\text{def}}{=} (\mathbf{x}_{0:t-1}^{(i)}, \tilde{\mathbf{x}}_t^{(i)})$
   - For $i = 1, \ldots, N$, evaluate the importance weights:

$$w_t(\tilde{\mathbf{x}}_{0:t}^{(i)}) = w_{t-1}^{(i)} \frac{p(\mathbf{y}_t | \tilde{\mathbf{x}}_t^{(i)}) p(\tilde{\mathbf{x}}_t^{(i)} | \mathbf{x}_{t-1}^{(i)})}{q(\tilde{\mathbf{x}}_t^{(i)} | \mathbf{y}_t, \mathbf{x}_{t-1}^{(i)})}$$

   - For $i = 1, \ldots, N$, normalize the importance weights:

$$W_t^{(i)} = \frac{w_t(\tilde{\mathbf{x}}_{0:t}^{(i)})}{\sum_{j=1}^{N} w_t(\tilde{\mathbf{x}}_{0:t}^{(j)})}$$

   - Resample (with replacement) N particles $\{\mathbf{x}_{0:t}^{(i)} : i = 1, \ldots, N\}$ from $\{\tilde{\mathbf{x}}_{0:t}^{(i)} : i = 1, \ldots, N\}$ according to the importance weights $\{W_t^{(i)} : i = 1, \ldots, N\}$. Set $w_t^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.

Resampling

- In resampling we replace

$$\hat{P}_N(d\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = \sum_{i=1}^{N} W_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$$

with

$$\tilde{P}_N(d\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = \frac{1}{N} \sum_{i=1}^{N} n_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$$

where $n_t^{(i)} \in \{0, 1, \ldots, N\}$ and $\sum_{i=1}^{N} n_t^{(i)} = N$.

- Usually we use schemes such that $E[n_t^{(i)}|W_t^{(1:N)}] = NW_t^{(i)}$ so $\tilde{P}_N(d\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$ is an unbiased approximation of $\hat{P}_N(d\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$.
- In multinomial resampling, we have $n_t^{(1:N)} \sim \text{multinomial}(N, W_t^{(1:N)})$.

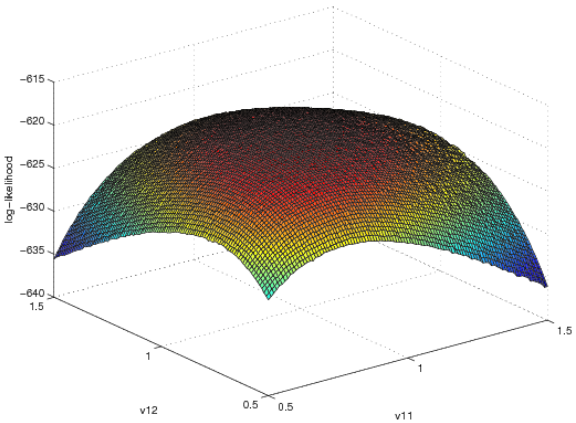
Smooth Likelihood Estimation

- We would like an estimator $\hat{L}_N(\theta)$ of $p(\mathbf{y}|\theta)$ to have three properties:
  1. Consistency: $\hat{L}_N(\theta) \xrightarrow{\text{P}} p(\mathbf{y}|\theta)$ as $N \to \infty$.
  2. Smoothness: $p(\mathbf{y}|\theta)$ is continuous in $\theta \Rightarrow \hat{L}_N(\theta)$ is continuous in θ.
  3. Tractability: $o(N^2)$ time complexity in general.
- Why smoothness?
  1. Captures the true nature of the likelihood.
  2. Better facilitates likelihood maximization:

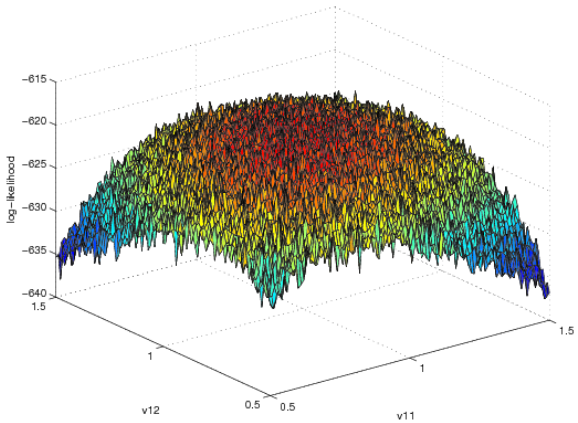
$$E[\hat{L}_N(\theta_2) - \hat{L}_N(\theta_1)] \xrightarrow{\text{P}} p(\mathbf{y}|\theta_2) - p(\mathbf{y}|\theta_1)$$

$$\text{var}(\hat{L}_N(\theta_2) - \hat{L}_N(\theta_1)) = \text{var}(\hat{L}_N(\theta_1)) + \text{var}(\hat{L}_N(\theta_2)) - 2\text{cov}(\hat{L}_N(\theta_1), \hat{L}_N(\theta_2))$$

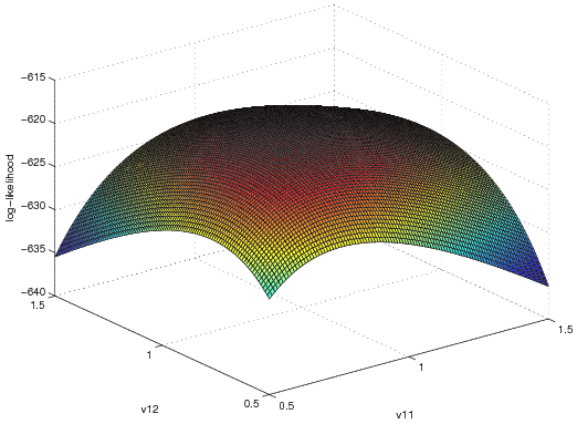
Smooth Likelihood Estimation (visualization)



(a) weighted binary tree



(b) vanilla



(c) true

Figure: 2D Gaussian State-Space Model Log-Likelihood

Common Random Numbers

- One way to achieve positive correlation is to use common random numbers (CRN).

$$p(\mathbf{y}|\theta) = \int p(\mathbf{y}, \mathbf{x}|\theta) d\mathbf{x} = \int p(\mathbf{y}|\mathbf{x}, \theta) \frac{p(\mathbf{x}|\theta)}{q(\mathbf{x}|\theta)} q(\mathbf{x}|\theta) d\mathbf{x}$$

  so each 'CRN' $\mathbf{x}_i \sim q(\mathbf{x}|\theta)$ gives the Monte Carlo estimate:

$$\hat{I}(\theta) = \frac{1}{N} \sum_{i=1}^{N} p(\mathbf{y}|\mathbf{x}_i, \theta) \frac{p(\mathbf{x}_i|\theta)}{q(\mathbf{x}_i|\theta)}$$

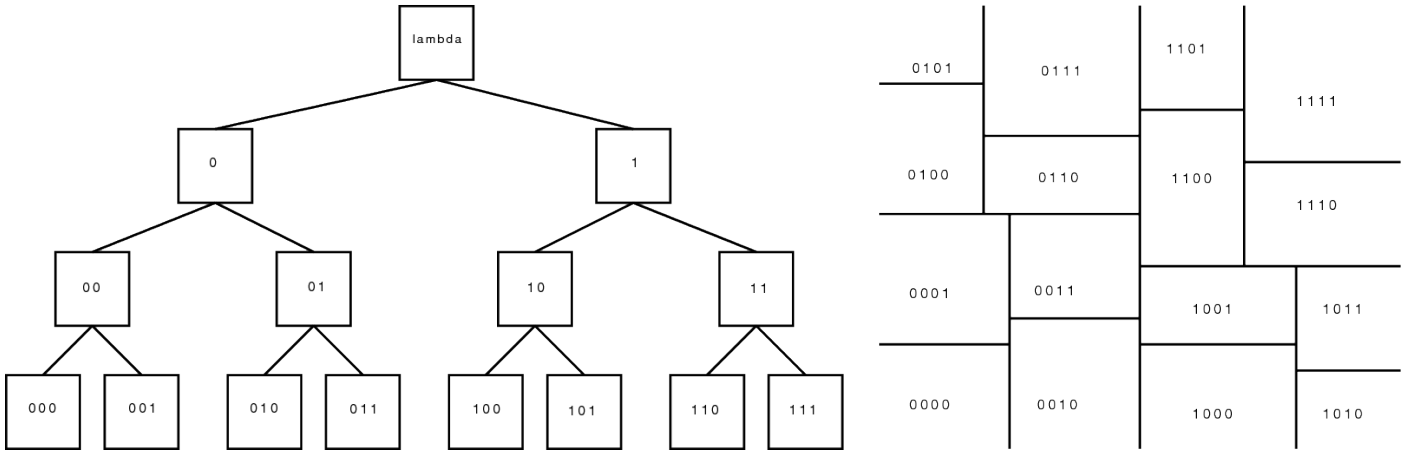
- $\mathbf{z}_i$'s are common and $\mathbf{x}_i = f(\mathbf{z}_i; \theta) \Rightarrow \mathbf{x}_i \sim q(\mathbf{x}_i|\theta)$, where f is continuous in θ.
- Still, we shouldn't use proposal distributions of this form.

CRN for Particle Filters

- Transition:
  - Use common $\{\mathbf{z}_t^{(i)}\}_{i=1}^N$ to produce $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{y}_t, \mathbf{x}_{t-1}^{(i)}, \theta)$
- Resampling:
  - Use common $\{u_t^{(i)}\}_{i=1}^N$ to sample $\{n_t^{(i)}\}_{i=1}^N$.
- Problem:
  - $n_t^{(1:N)} \sim$ multinomial$(N, W_t^{(1:N)})$.
  - The weights $W_t^{(1:N)}$ are dependent on $\theta \Rightarrow n_{t,\theta}^{(1:N)} \neq n_{t,\theta'}^{(1:N)}$.
- Key observation:
  - $\hat{F}_{t,N}(j) = \sum_{i=1}^{j} W_t^{(i)}$.
  - When weights change, we will pick a particle with a 'close' index.
  - Can we make particles with close indices be close themselves?
  - For 1D state variables, $\hat{F}_{t,N}(x) = \sum_{x_t^{(i)} \leq x} W_t^{(i)}$ will work [Pitt '02].
- What can we do when state variables are not 1D?

Tree-Based Resampling

- Imagine we want to sample from $p(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^2$.
- If we can compute the median of any truncated version of p and sample Bernoulli(0.5) rv's, we can recursively split regions along their medians and pick either subregion with probability 0.5.



(a) tree representation (b) partition representation

Figure: Recursive binary partitioning of the space

- The limit of the region contains a single point (almost surely) as the length of the random binary string goes to ∞.

Theoretical Median-Cutting Algorithm $f_n(u)$

- Draw $u \sim U[0,1]$ and compute $f_n(u)$ as follows:

1.
   - Set $a_1^{(1)} = a_2^{(1)} = \cdots = a_d^{(1)} = -\infty$ and $b_1^{(1)} = b_2^{(1)} = \cdots = b_d^{(1)} = \infty$
   - Let $\mathcal{X}_{-1}^{(1)} \stackrel{\text{def}}{=} \prod_{i=2}^{d}[a_i^{(1)}, b_i^{(1)}]$
   - Compute t_1 satisfying $\int_{a_1^{(1)}}^{t_1} \int_{\mathcal{X}_{-1}^{(1)}} p(\mathbf{x}) dx_{2:d} dx_1 = 0.5$
   - For $i = 1, \ldots, d$: set $a_i^{(2)} = a_i^{(1)}$ and $b_i^{(2)} = b_i^{(1)}$
   - If $u < 0.5$, set $b_1^{(2)} = t_1$ and $u = 2u$, else set $a_1^{(2)} = t_1$ and $u = 2(u - 0.5)$
2. For $j = 2, \ldots, n$
   - Let $k = j \mod d$. If $k = 0$, let $k = d$.
   - Let $\mathcal{X}_{-k}^{(j)} \stackrel{\text{def}}{=} \prod_{i=1}^{k-1}[a_i^{(j)}, b_i^{(j)}] \prod_{i=k+1}^{d}[a_i^{(j)}, b_i^{(j)}]$
   - Compute t_j satisfying $2^{j-1} \int_{a_k^{(j)}}^{t_j} \int_{\mathcal{X}_{-k}^{(j)}} p(\mathbf{x}) dx_{1:k-1} dx_{k+1:d} dx_k = 0.5$
   - For $i = 1, \ldots, d$: set $a_i^{(j+1)} = a_i^{(j)}$ and $b_i^{(j+1)} = b_i^{(j)}$
   - If $u < 0.5$, set $b_k^{(j+1)} = t_j$ and $u = 2u$, else set $a_k^{(j+1)} = t_j$ and $u = 2(u - 0.5)$
3. Return $\mathcal{X}^{(n+1)} = \prod_{i=1}^{d}[a_i^{(j+1)}, b_i^{(j+1)}]$.

Practical Algorithm I: Unweighted Binary Trees

- Given N particles and weights, construct a tree as follows:
  - Compute the weighted median of the particles in one dimension and split the particles into two sets (children).
  - For each child, compute the weighted median of its particles in the next dimension and split the particles into two sets.
  - Repeat, cycling through the dimensions.
- We can pick a particle with a common random number $u \in \mathbb{R}$, traversing the tree as in the theoretical version.
  - Think of the path selected as a binary string β where each bit is independent of the weights.
- We need to replicate a particle at each split to make the weight of each child equal.
- Total cost is usually in $O(N \log N)$ for constructing the tree and sampling N particles.

Practical Algorithm II: Weighted Binary Trees

- Given $N = 2^k$ particles and weights, construct a tree as follows:
  - Compute the (unweighted) median of the particles in one dimension and split the particles into two sets (children).
  - For each child, compute the (unweighted) median of its particles in the next dimension and split the particles into two sets.
  - Repeat, cycling through the dimensions.
- The weight associated with each node is the sum of the weights of its constituent particles.
- We can pick a particle with a common random number $\mathbf{u} \in \mathbb{R}^k$, traversing the tree according to the weights.
  - Think of the path selected as a binary string β where each bit is dependent on the weights.
  - Further improvement can be attained by using $\mathbf{u} \in \mathbb{R}^d$.
- Total cost is in $O(N \log N)$ for constructing the tree and sampling N particles.

Correctness

- For any tree structure with properly weighted nodes we have

$$\Pr[\text{select index } i] = \frac{\text{weight of particle } i}{\text{sum of all weights}}$$

- Let the set of particles in the node at level $j+1$ reached by a given $\beta_{1:j} \in \{0,1\}^j$ be denoted $S_j(\beta_{1:j})$

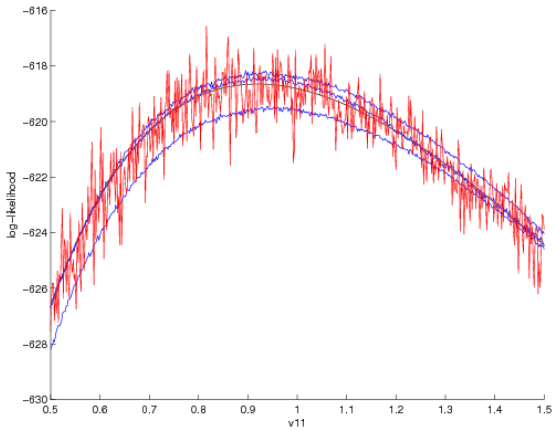
$$\Pr[\beta_{1:j}] = \frac{\text{weight of } S_j(\beta_{1:j})}{\text{sum of all weights}}$$

- With respect to smoothness
  - Nodes define similar regions when θ changes.
  - There is a common (possibly null) prefix of $\boldsymbol{\beta}$ between different runs.
    $\Rightarrow$ Particles tend to be close even when θ changes.

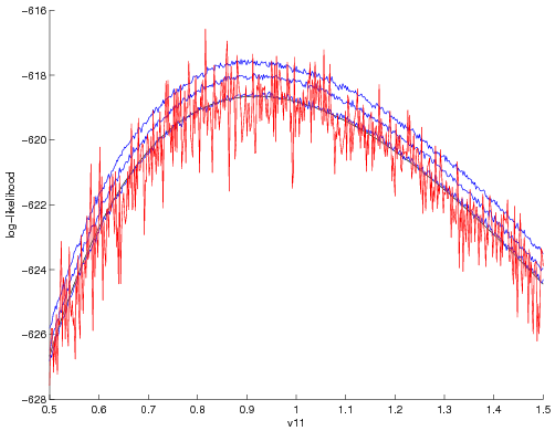
Comparing the binary trees

- For the weighted binary tree
  - The random string β depends on the weights.
  - Set membership for particles does not depend on the weights.
- For the unweighted binary tree
  - The random string β is independent of the weights.
  - Set membership for particles does depend on the weights.
- It is hard to tell which is better in theory for finite N.
- For the unweighted tree, as $N \to \infty$, the particle returned for u converges in probability to the almost surely unique particle in $\lim_{n\to\infty} f_n(u)$.
- The weighted binary tree is easier to implement.

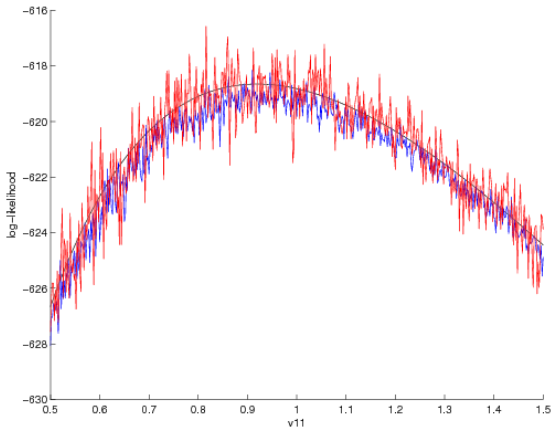
2D Gaussian State-Space Model



(a) weighted binary tree



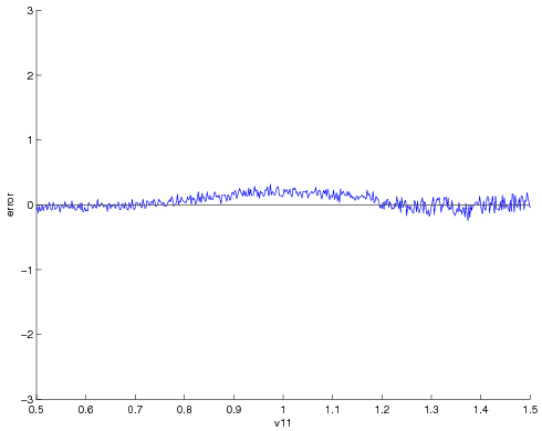
(b) unweighted binary tree



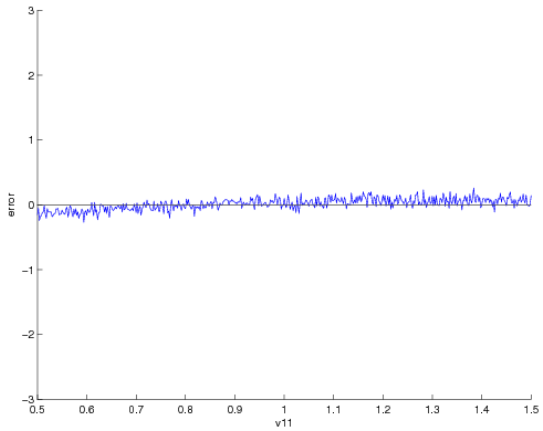
(c) CRN vanilla

Figure: $N = 1024$ for tree-based filters, $N = 1536$ for vanilla filters

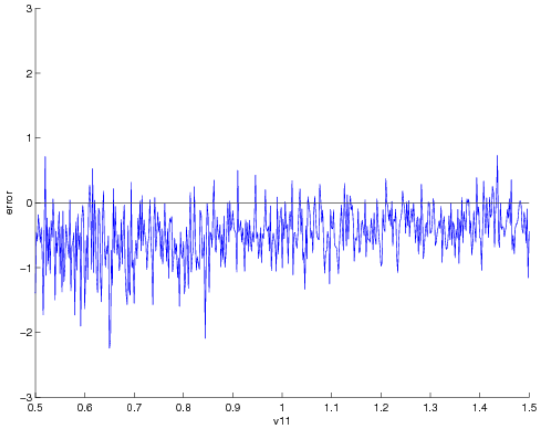
2D Gaussian State-Space Model Errors



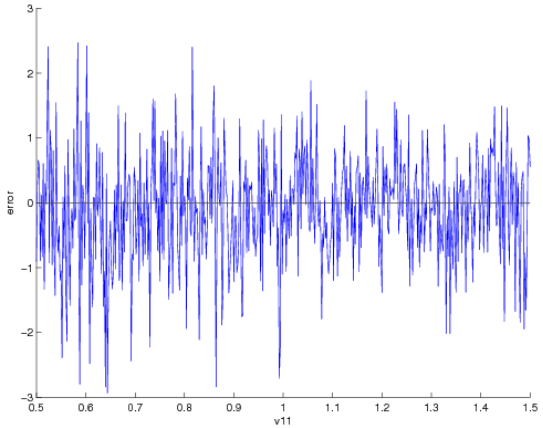
(a) weighted binary tree



(b) unweighted binary tree



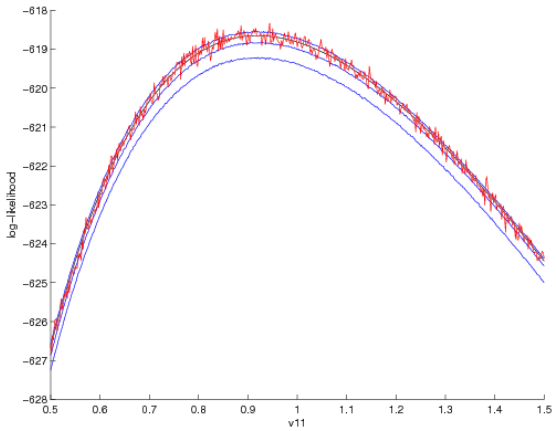
(c) CRN vanilla



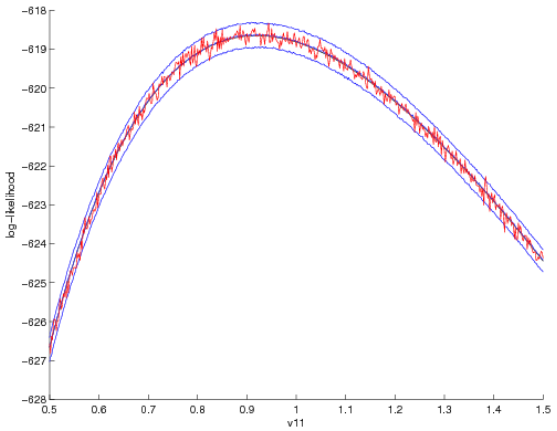
(d) vanilla

Figure: $N = 1024$ for tree-based filters, $N = 1536$ for vanilla filters

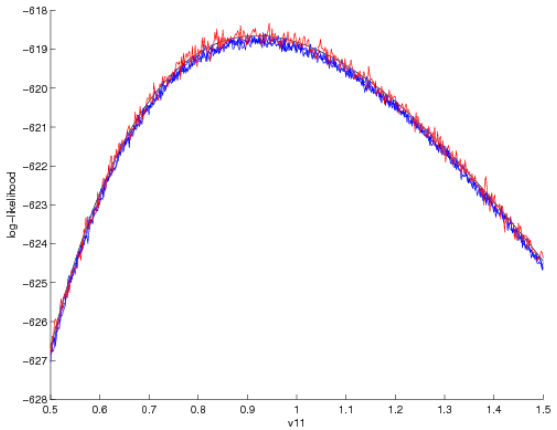
2D Gaussian State-Space Model (locally optimal)



(a) weighted binary tree



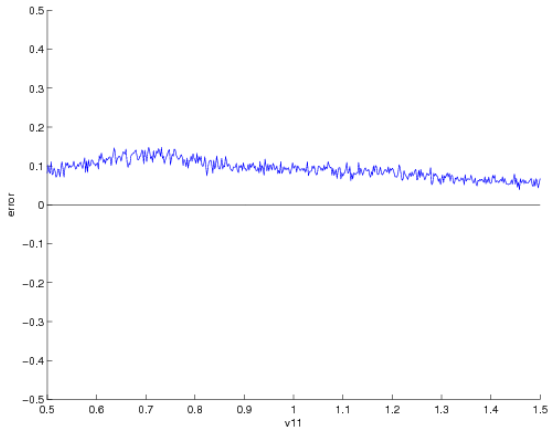
(b) unweighted binary tree



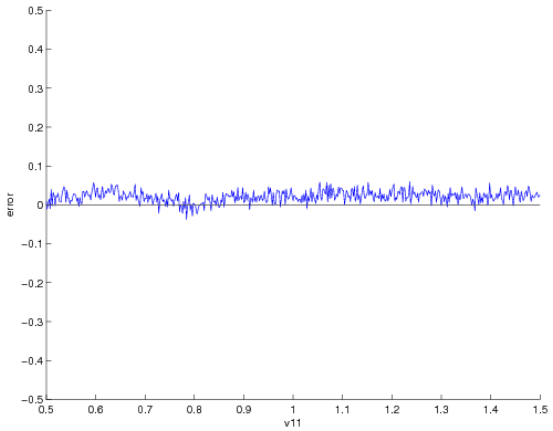
(c) CRN vanilla

Figure: $N = 1024$ for tree-based filters, $N = 1536$ for vanilla filters

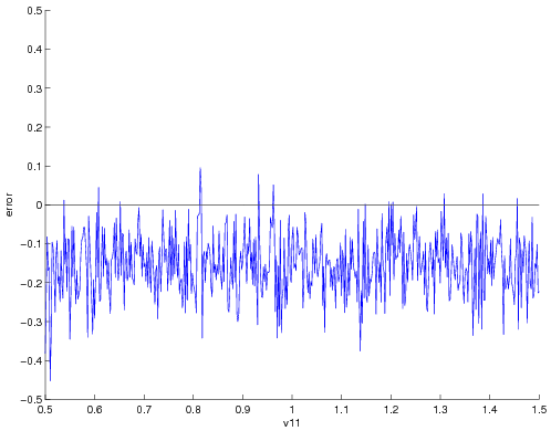
2D Gaussian State-Space Model Errors (locally optimal)



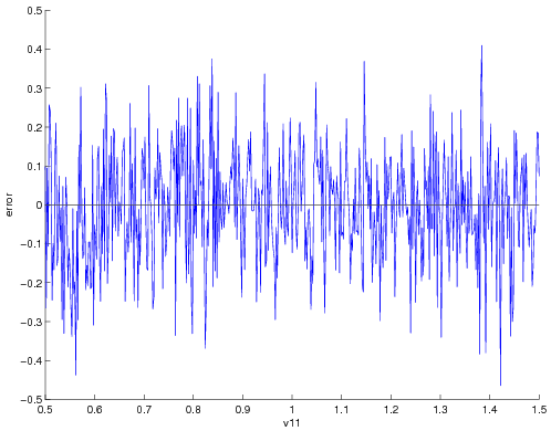
(a) weighted binary tree



(b) unweighted binary tree



(c) CRN vanilla



(d) vanilla

Figure: $N = 1024$ for tree-based filters, $N = 1536$ for vanilla filters

Factor Stochastic Volatility Model

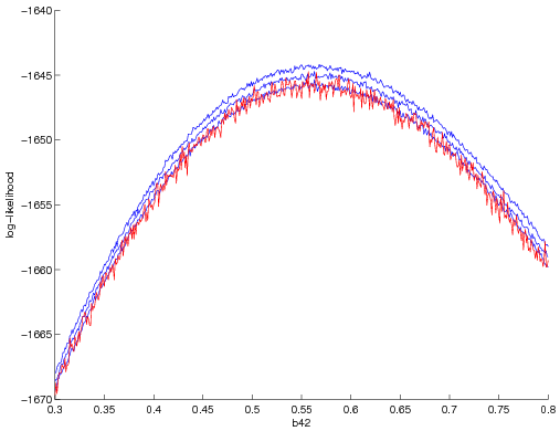
- Used in quantitative finance.
- Models volatility of asset values as a stochastic process.
- Factor loading matrix allows us to model dependent item valuations.
- Calibration (parameter estimation) is very important in practice.
- Model is as in [Liu & West, '00]

$$\begin{aligned}\mathbf{y}_t &\sim N(\mathbf{B}\mathbf{f}_t, \boldsymbol{\Psi}) \\ \mathbf{f}_t &\sim N(\mathbf{0}, \mathbf{H}_t) \\ \boldsymbol{\alpha}_t &\sim N(\boldsymbol{\Phi}\boldsymbol{\alpha}_{t-1}, \mathbf{U})\end{aligned}$$

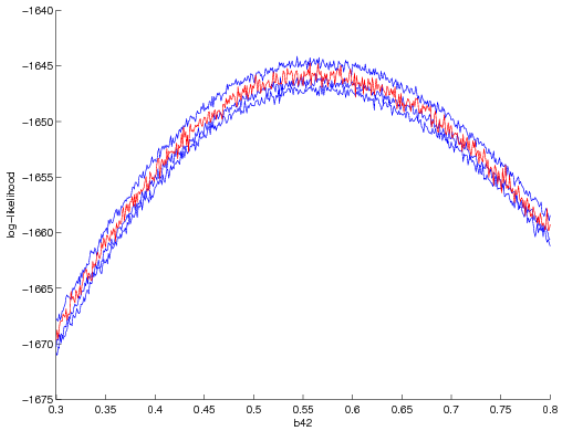
where $\boldsymbol{\Psi} \stackrel{\text{def}}{=} \text{diag}(\psi_1, \ldots, \psi_M)$, $\mathbf{H}_t \stackrel{\text{def}}{=} \text{diag}(\exp(\boldsymbol{\alpha}_t))$, $\boldsymbol{\Phi} \stackrel{\text{def}}{=} \text{diag}(\phi_1, \ldots, \phi_K)$

- This gives $\mathbf{y}_t | \boldsymbol{\alpha}_t \sim N(\mathbf{0}, \mathbf{B}\mathbf{H}_t\mathbf{B}^T + \boldsymbol{\Psi})$

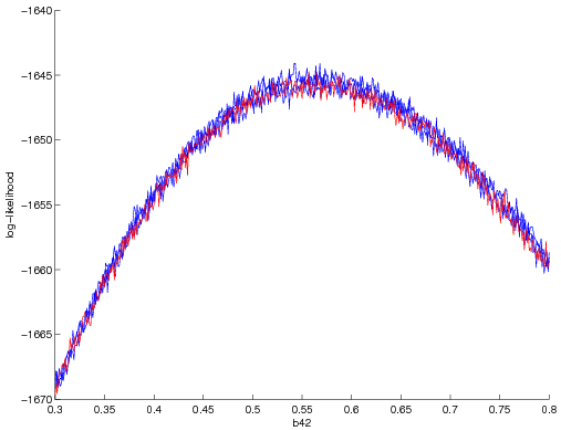
Factor Stochastic Volatility Results



(a) weighted binary tree



(b) unweighted binary tree



(c) CRN vanilla

Figure: $N = 1024$ for tree-based filters, $N = 1536$ for vanilla filters

Partially Observed Lotka-Volterra Model

- Simple case of stochastic kinetic models used in systems biology.
- Describes evolution of predator and prey levels or concentrations of chemical reactants.
- Can use SMC to simulate a diffusion approximation of the model.
- Particularly interesting when predator population is unobserved.

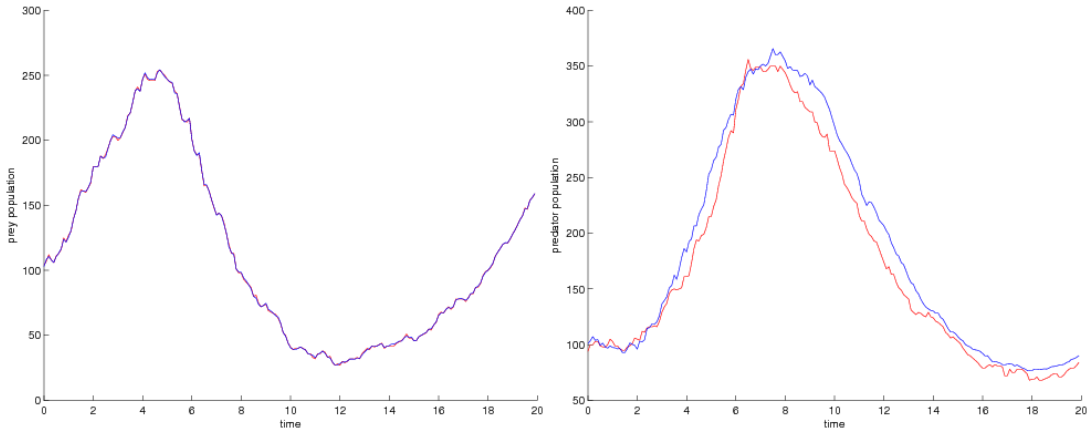
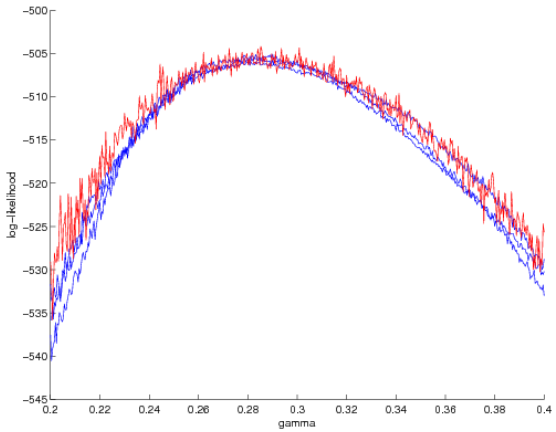
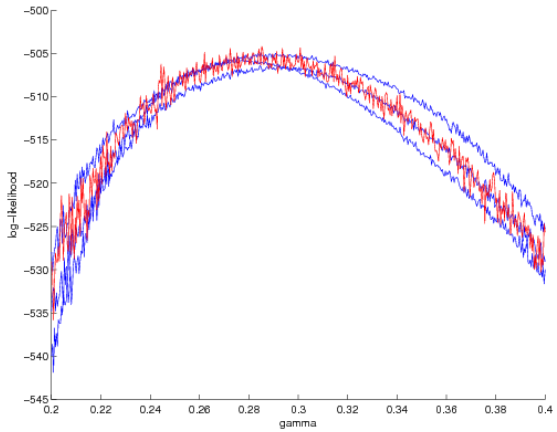


Figure: Estimated expected predator-prey population levels for the partially observed Lotka-Volterra model

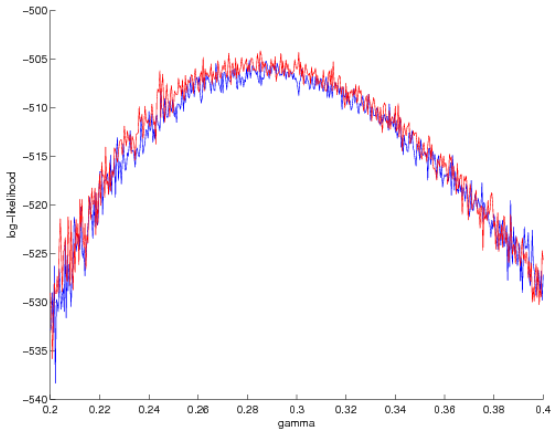
Partially Observed Lotka-Volterra Results



(a) weighted binary tree



(b) unweighted binary tree



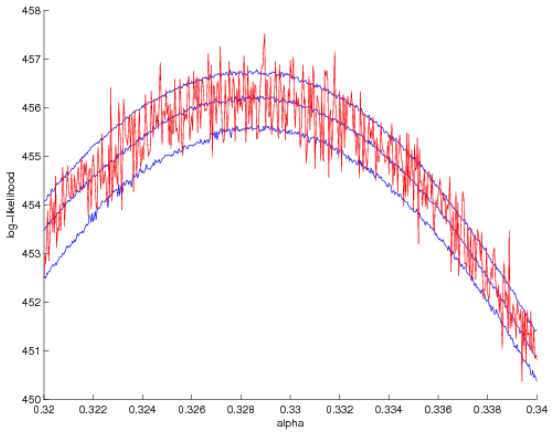
(c) CRN vanilla

Figure: $N = 1024$ for tree-based filters, $N = 1536$ for vanilla filters

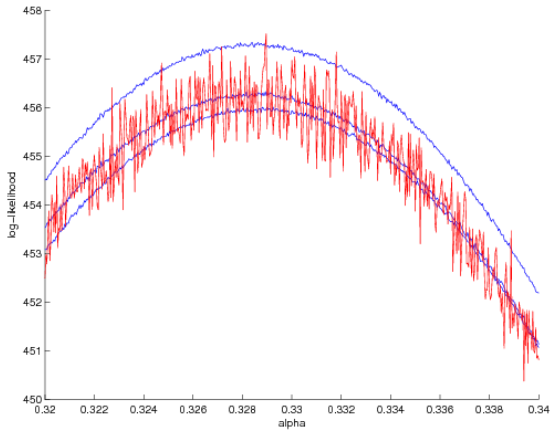
Dynamic Stochastic General Equilibrium Model

- Used to explain macroeconomic phenomena using microeconomic principles.
- We observe representative rational agents in the market.
- System is subject to random shocks.

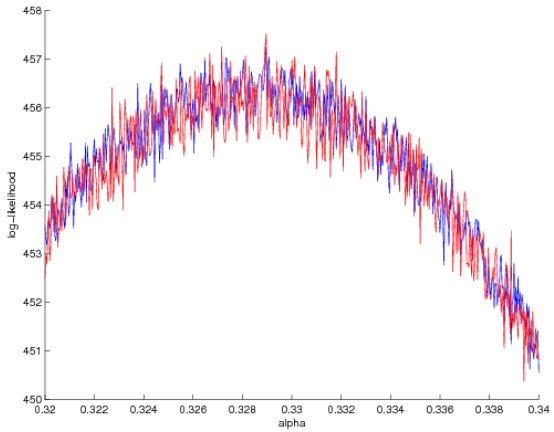
Dynamic Stochastic General Equilibrium Results



(a) weighted binary tree



(b) unweighted binary tree



(c) CRN vanilla

Figure: $N = 1024$ for tree-based filters, $N = 1536$ for vanilla filters

Dynamic Stochastic General Equilibrium Results II

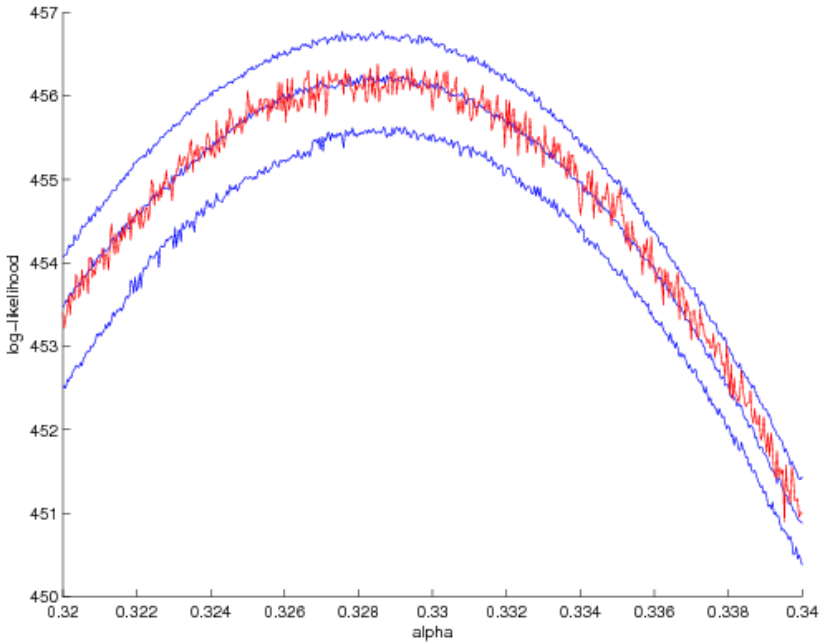
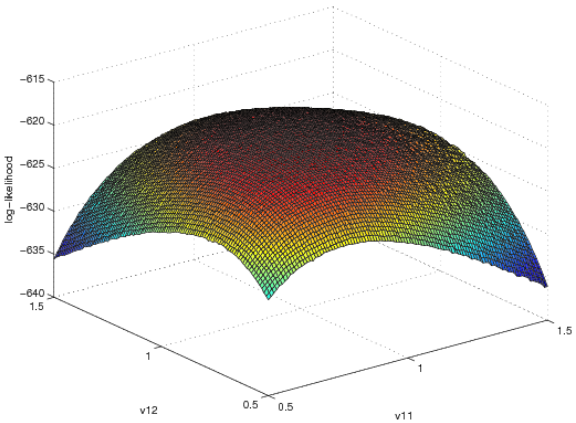
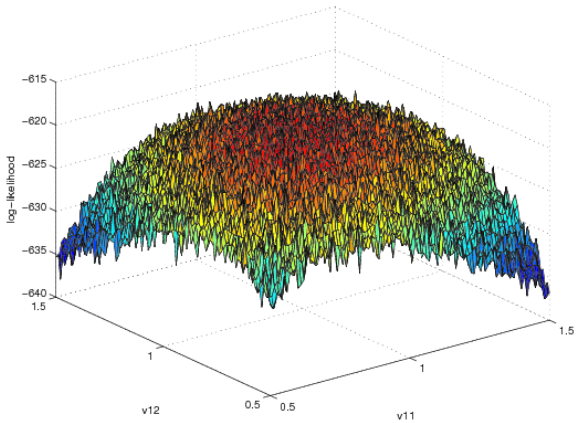


Figure: DSGE log-likelihood plots for the vanilla particle filter with $N = 20000$ (red) and the weighted binary tree particle filter with 1024 particles (blue)

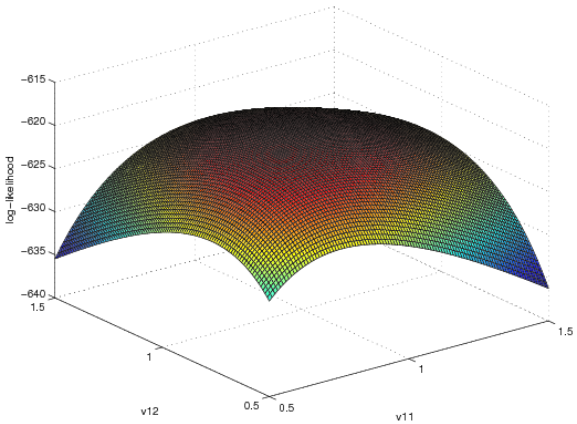
2D Gaussian State-Space Model Log-Likelihood



(a) weighted binary tree



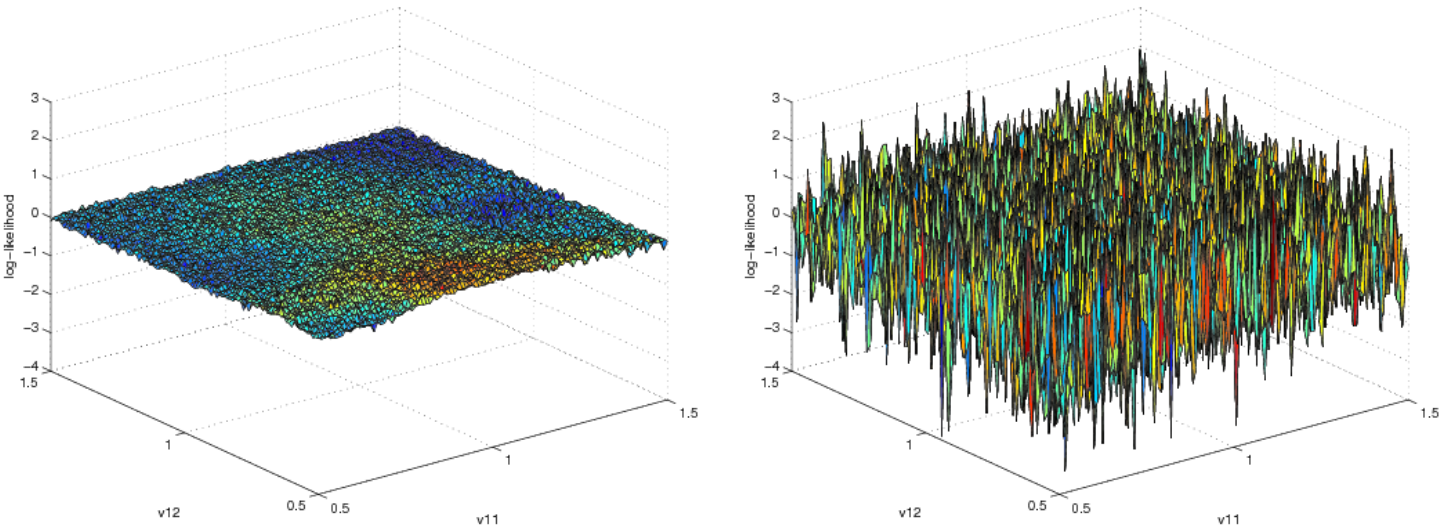
(b) vanilla



(c) true

Figure: 2D Gaussian State-Space Model Log-Likelihood

2D Gaussian State-Space Model Log-Likelihood Errors



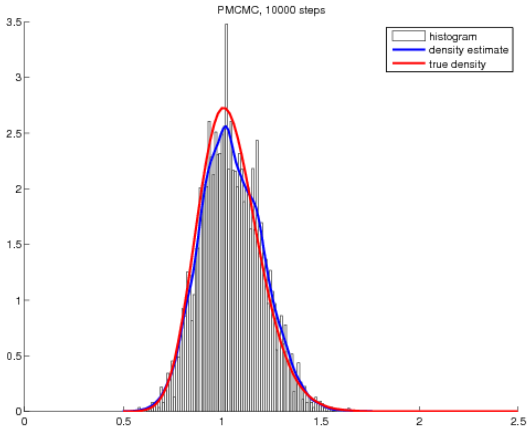
(a) weighted binary tree

(b) vanilla

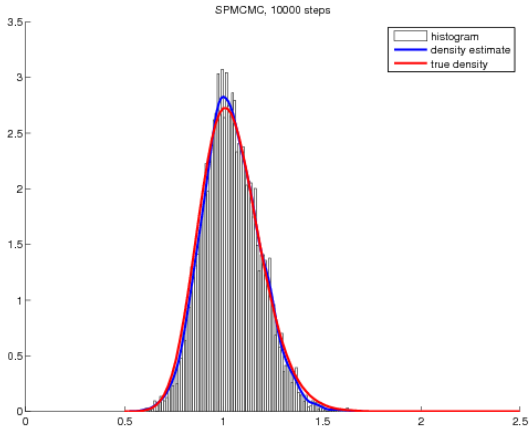
Figure: 2D Gaussian State-Space Model Log-Likelihood Errors

Application to MCMC

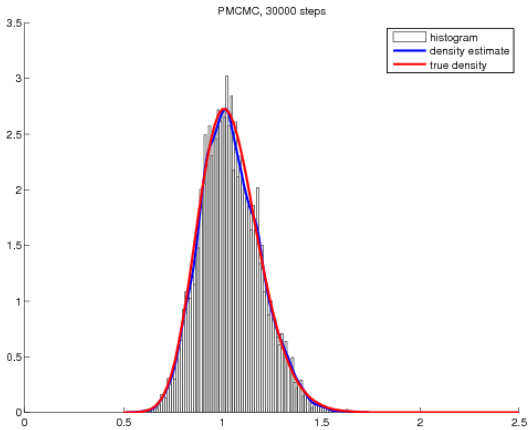
- Note that the estimate of the likelihood is unbiased (without interpolation)! [Del Moral '04]
- We can perform MCMC on $(\theta, \mathbf{u})$ treating $\mathbf{u} \in \mathbb{R}^{Nd}$ as an auxiliary variable.
- For some steps, propose a new $\mathbf{u}$, for others propose a new θ.
- We obtain a reduction in the 'variance' of the acceptance ratio, ie. $\min\{1, \frac{p(\mathbf{y}_{0:T}|\theta',\mathbf{u}')p(\theta')p(\mathbf{u}')}{p(\mathbf{y}_{0:T}|\theta,\mathbf{u})p(\theta)p(\mathbf{u})}\}$ is closer to $\min\{1, \frac{p(\mathbf{y}_{0:T}|\theta')p(\theta')}{p(\mathbf{y}_{0:T}|\theta)p(\theta)}\}$
- This can constitute an improvement over the standard PMMH algorithm of [Andrieu, Doucet & Holenstein (to appear)]
- For the 2D Gaussian state-space model, acceptance ratio differences/move discrepancy rate of 9% for SPMCMC and 36% for PMMH compared to the marginal acceptance ratios.



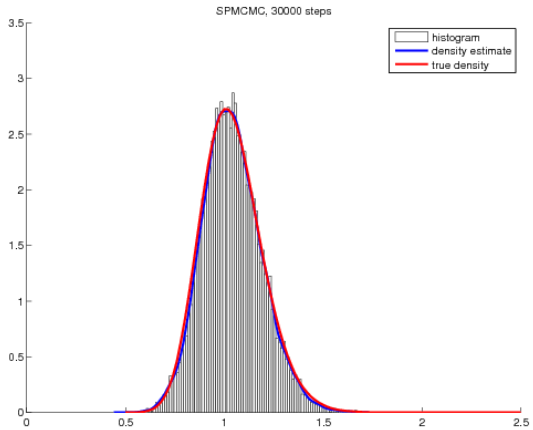
(a) PMMH, 10000 steps



(b) SPMCMC, 10000 steps



(c) PMMH, 30000 steps



(d) SPMCMC, 30000 steps

Figure: PMMH and SPMCMC results on 2D Gaussian state-space model

Remarks

- The tree-based resampling schemes lead to significantly smoother estimators.
- It is the particles that are smooth as a function of θ.
- T can be arbitrarily large: resampling 'resets' the particles.
- There are no regularity conditions or auxiliary distributions or extra parameters.
- $O(N \log N)$ time complexity is not that bad in practice.
- This type of variance reduction method can accelerate PMCMC convergence.

References

- Michael K. Pitt. Smooth Particle Filters for Likelihood Evaluation and Maximisation. *The Warwick Economics Research Paper Series* 651, University of Warwick, Department of Economics, 2002.
- Jane Liu and Mike West. Combined parameter and state estimation in simulation-based filtering. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2000.
- Pierre Del Moral. Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications. New York: Springer. 2004.
- Christophe Andrieu, Arnaud Doucet and Roman Holenstein. Particle Markov chain Monte Carlo (with discussion). JRSS B (to appear).