Sequential MCMC Methods: Particle and Practical Filtering

Nick Polson
University of Chicago

Overview

- Particle and Practical Filtering
- Filtering with Fixed Parameters
- Filtering with Parameter Learning
- Examples
  - Random walk model
  - Stochastic Volatility models/SP500
  - Stochastic Volatility Jump models
- Discussion

Particle and Practical Filtering

- Particle Filtering

  Works extremely well in standard cases

  Model correct, parameters known, low-dimensions

- However, degeneracy occurs, especially when model is mis-specified or parameters are unknown.

- Example: Stock Market Crash of '87, for log-SV model

- Goal: accurate filtering algorithm that allows for parameter learning and avoids degeneracy - practical filtering

State-Space Framework

- General state-space model:

$$\begin{aligned} \text{Measurement:} \quad & S_t \sim p(S_t|V_t, \Theta) \\ \text{System:} \quad & V_t \sim p(V_t|V_{t-1}, \Theta) \\ \text{Parameters:} \quad & \Theta \sim p(\Theta) \end{aligned}$$

- Notation: $V_{s:t} = (V_s, \ldots, V_t)'$, $S_{s:t} = (S_s, \ldots, S_t)'$ denote the block of states and data, respectively.

- Goal: Estimate joint filtering and parameter distribution, $p(V_t, \theta|S_{1:t}), t = 1, \ldots, T$, sequentially as new data arrive.

General Idea

1. View filter as marginal from the smoothing distribution:

$$p(V_t|S_{1:t}) = \int p(V_{0:t}|S_{1:t})dV_{0:t-1}$$

2. Solve filtering problem using efficient smoothing techniques (e.g., FFBS algorithm, simulation smoother).

Particle Filtering

Here's a simple simulation approach:

$$p\left(V_t|S_{1:t}\right): \text{ filtering density}$$
$$p\left(V_{t+1}|S_{1:t}\right): \text{ predictive density}$$
$$p\left(S_t|V_t\right): \text{ likelihood}$$
$$p\left(V_{t+1}|V_t\right): \text{ state transition}$$

Bayes rule links the predictive and filtering densities through the identity

$$p\left(V_{t+1}|S_{1:t+1}\right) = \frac{p\left(S_{t+1}|V_{t+1}\right) p\left(V_{t+1}|S_{1:t}\right)}{p\left(S_{t+1}|S_{1:t}\right)}$$

Estimate

$$p^N\left(V_t|S_{1:t}\right) = \sum_{i=1}^{N} \delta_{V_t^{(i)}} \pi_t^i$$

$$p^N\left(V_{t+1}|S_{1:t}\right) = \sum_{i=1}^{N} p\left(V_{t+1}|V_t^{(i)}\right) \pi_t^i,$$

Leads to next filtering distribution

$$p^N\left(V_{t+1}|S_{1:t+1}\right) \propto p\left(S_{t+1}|V_{t+1}\right) \sum_{i=1}^{N} p\left(V_{t+1}|V_t^{(i)}\right) \pi_t^i.$$

and

$$p^N\left(V_{t+1}|S_{1:t+1}\right) \propto \underbrace{p\left(S_{t+1}|V_{t+1}\right)}_{Likelihood} \underbrace{\sum_{i=1}^{N} p\left(V_{t+1}|V_t^{(i)}\right) \pi_t^i}_{Prior}.$$

Algorithm:

Start with a large set of "particles" $\left\{V_0^{(i)}\right\}_{i=1}^{N}$.

- The algorithm pushes through draws, $\left\{V_t^{(i)}\right\}_{i=1}^{N}$ using a re-sampling procedure based on filtering equations above.
- Auxiliary particle filter (APF) uses an initial re-sampling step and propagates these higher likelihood samples forward.

Practical Filtering: Key Idea

- For state-space models, $\exists$ lag k such that V_{t-k} and S_t are approximately independent given $S_{t-k+1:t-1}$.

- Write the filter as a mixture of lag-k smoothers:

$$\begin{aligned} p(V_t|S_{1:t}) &= \int p(V_{0:t}|S_{1:t})\, dV_{0:t-1} \\ &= \int p(V_{t-k+1:t}|V_{t-k}, S_{1:t})\, p(V_{t-k}|S_{1:t})\, dV_{t-k+1:t-1} \end{aligned}$$

1. Assume we have samples from $p(V_{t-k}|S_{1:t}) \approx p(V_{t-k}|S_{1:t-1})$.
2. Now, given V_{t-k}, we need a fast smoothing algorithm to generate from $p(V_{t-k+1:t}|V_{t-k}, S_{t-k+1:t})$.

Algorithm 1: Filtering with Known Parameters

Burn-In: For $t = 1, \ldots, k$:

For $g = 1, \ldots, G$:

For $i = 1, \ldots, I$:

Generate $V_{0:t} \sim p(V_{0:t}|S_{1:t})$

Set $\tilde{V}_0^{(g)} := V_0$

Sequential updating: For $t = k+1, \ldots, T$:

For $g = 1, \ldots, G$:

For $i = 1, \ldots, I$:

Generate $V_{t-k+1:t} \sim p(V_{t-k+1:t}|\tilde{V}_{0:t-k}^{(g)}, S_{1:t})$

Set $\tilde{V}_{t-k+1}^{(g)} := V_{t-k+1}$

Filtering with Parameter Learning

- Assume that $\exists$ lag k such that $V_{0:t-k}$ and S_t are approximately independent given $S_{1:t-1}$, marginalizing w.r.t. θ.

- Write the joint filtering distribution as:

$$\begin{aligned} p(V_t, \theta | S_{1:t}) &= \int p(V_{0:t}, \theta | S_{1:t}) \, dV_{0:t-1} \\ &= \int p(V_{t-k+1:t}, \theta | V_{0:t-k}, S_{1:t}) \, p(V_{0:t-k} | S_{1:t}) \, dV_{0:t-1} \end{aligned}$$

1. Assume we have samples from $p(V_{0:t-k}|S_{1:t}) \approx p(V_{0:t-k}|S_{1:t-1})$.
2. Now, given $V_{0:t-k}$, we need a fast algorithm to generate from $p(\theta, V_{t-k+1:t}|V_{0:t-k}, S_{1:t})$.

Algorithm 2: Filtering with Unknown Parameters

Initialization: Set $\Theta^{(g)} = \Theta_0$, $g = 1, \ldots, G$.

Burn-In: For $t = 1, \ldots, k$:

For $g = 1, \ldots, G$: (initialize $\Theta = \Theta^{(g)}$)

For $i = 1, \ldots, I$:

Generate $V_{0:t} \sim p(V_{0:t}|\Theta, S_{1:t})$

Generate $\Theta \sim p(\Theta|V_{0:t}, S_{1:t})$

Set $(\tilde{V}_0^{(g)}, \Theta^{(g)}) := (V_0, \Theta)$.

Sequential updating: For $t = k+1, \ldots, T$:

For $g = 1, \ldots, G$: (initialize $\Theta = \Theta^{(g)}$)

For $i = 1, \ldots, I$:

Generate $V_{t-k+1:t} \sim p(V_{t-k+1:t}|\tilde{V}_{0:t-k}^{(g)}, \Theta, S_{1:t})$

Generate $\Theta \sim p(\Theta|\tilde{V}_{0:t-k}^{(g)}, V_{t-k+1:t}, S_{1:t})$

Set $(\tilde{V}_{t-k+1}^{(g)}, \Theta^{(g)}) := (V_{t-k+1}, \Theta)$

Example 1: Random-walk Model with Unknown Variance

- Simple linear model:

$$\begin{aligned} S_t &= V_t + \epsilon_t, & \epsilon_t &\sim \mathcal{N}(0, \sigma^2) \\ V_t &= \phi V_{t-1} + \eta_t, & \eta_t &\sim \mathcal{N}(0, \sigma^2) \end{aligned}$$

- Fix $V_0 = 0$ and $\phi = .9$.
- Conjugate prior: $\sigma^2 \sim \mathcal{IG}(n_0, S_0)$.
- Marginal filtering distributions available analytically:

$$\begin{aligned} \sigma^2 | S_{1:t} &\sim \mathcal{IG}(n_t, S_t) \\ V_t | S_{1:t} &\sim T_{n_t - 1}(m_t, S_t\, C_t) \end{aligned}$$

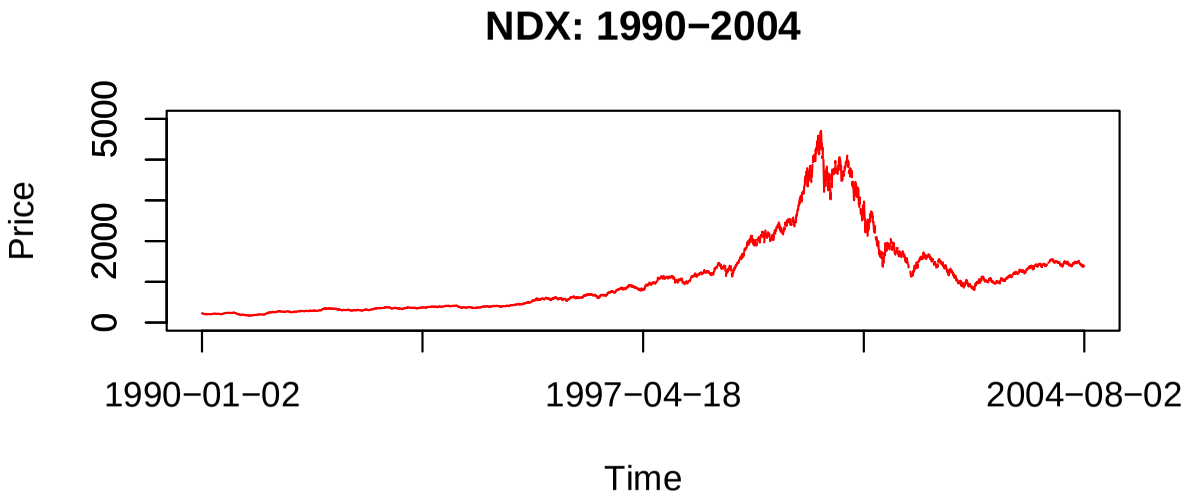
- Practical filter
  - Two-block Gibbs to generate (V, σ^2)
    - $(V|\sigma^2)$: FFBS
    - $(\sigma^2|V)$: Inverse Gamma
  - Lag $k = 25$, $G = 1000$ chains, $I = 5$ iterations per chain
- SIR Particle Filter (GSS '93)
  - Choose $G = 100,000$ particles

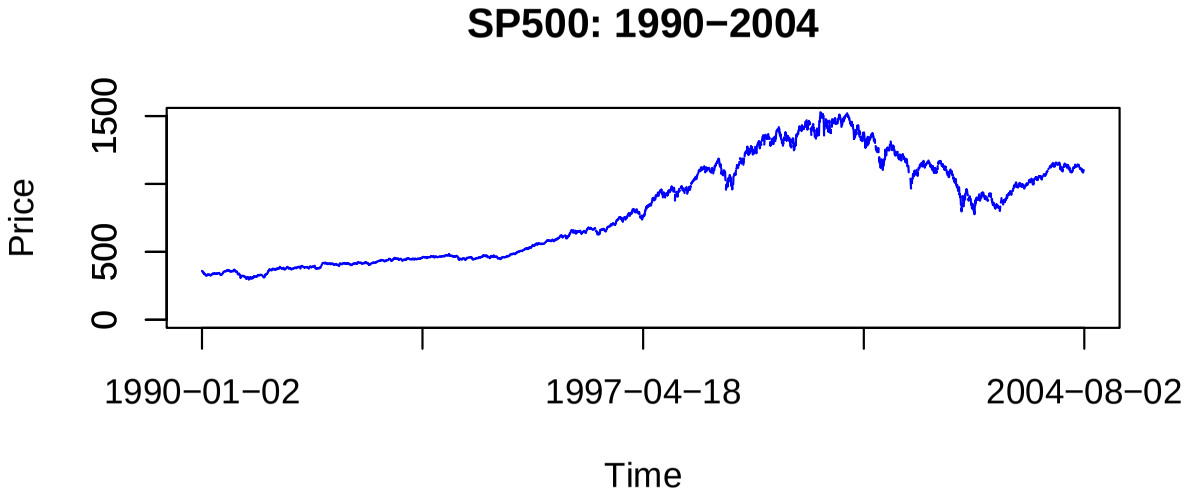
Example: Stochastic Volatility

- Standard Log-AR(1) model (JPR, 94):

$$\begin{aligned} y_t &= \exp(x_t/2)\epsilon_t, & \epsilon_t &\sim \mathcal{N}(0,1) \\ x_t &= \alpha + \beta x_{t-1} + \eta_t, & \eta_t &\sim \mathcal{N}(0,\sigma^2) \end{aligned}$$

- y_t asset return, x_t log volatility
- IID Errors (ie, no leverage).
- Non-Gaussian likelihood
- Shephard & Kim (1994) mixture-of-normals approximation


NDX: 1990−2004
Price
0
2000
5000
1990−01−02
1997−04−18
2004−08−02
Time


SP500: 1990−2004
Price
0
500
1500
1990−01−02
1997−04−18
2004−08−02
Time

Implementation

- Conjugate prior for $\theta = (\alpha, \beta, \sigma^2)$.
- Practical filter
  - Indicator variable $\mathbf{z}$ to break mixture
  - Three-block Gibbs
    - $(\mathbf{z}|V, \theta)$: Multinomial
    - $(V|\mathbf{z}, \theta)$: FFBS
    - $(\theta|V)$: Normal-inverse gamma
  - Lag $k = 50$, $G = 2000$ chains, $I = 10$ iterations per chain
- SIR Particle filter
  - Add small evolution noise for θ.
  - Use $G = 100,000$ particles

Bayesian/Monte Carlo Filtering and Smoothing References

- Gordon, Salmond & Smith (1993)
- Kitagawa (1996)
- Hürzeler & Künsch (1998)
- Pitt & Shephard (1999)
- Doucet, Godsill & Andrieu ('00)
- Liu & West (2001)
- Doucet, deFreitas & Gordon ('01)
- Many others...

- Carlin, Polson & Stoffer (1992)
- Carter & Kohn (1994)
- Frühwirth-Schnatter (1994)
- de Jong & Shephard (1995)
- Kitagawa (1996)
- Shephard & Pitt (1997)
- Durbin & Koopman (2000,1)
- Many others...

Application: Filtering with Jumps

Main Problem:

Identification of volatility V_t, jump times, sizes (J_t, ξ_t) and parameters Θ

Bayes Rule

$$p(V_t, J_t, \xi_t, \Theta | S_{1:t}) \propto p(S_{1:t} | V_t, J_t, \xi_t, \Theta) p(V_t, J_t, \xi_t, \Theta)$$

MCMC simulates from this distribution and provides estimates of quantities of interest – sequentially.

- Model: $p(V_t, J_t, \xi_t, \Theta)$ needs specification. Decomposes as

$$p(V_t, J_t, \xi_t | \Theta) p(\Theta)$$

  Prior $p(\Theta)$ helps in identification.

  "Things in nature rarely jump" (Marshall)

So far we've looked at

1. Particle Filter

   Can degenerate; works very well for just states including nonlinearities.

2. Practical Filter

   Doesn't degenerate and can be used for sequential parameter learning

Now we'll look at a more efficient way of "pushing" particle through using the

- Auxiliary Particle Filter: (Pitt and Shephard, 1999).

  Can also be used for sequential parameter learning (Storvik, 2002)

APF Algorithm

Update set of N particles $(V_t^{(g)}, \Theta^{(g)}, s_t^{(g)})$

1. Begin with particles $\tilde{V}_{t-1}^{(g)} = (V_{t-1}^{(g)}, \Theta^{(g)}, s_{t-1}^{(g)})$ and associated weights, $w_{t-1}^{(g)}$.

2. Compute first-stage weights, $\lambda_t^{(g)} \propto p^*(S_t|V_{t-1}^{(g)}, \Theta^{(g)})$, and re-sample particles $\tilde{V}_{t-1}^{(g)}$ with probabilities proportional to $\lambda_t^{(g)}$.

3. For each re-sampled particle:

   (a) Generate $\Theta^{(g)}, V_t$

   (b) Update sufficient statistics $s_t^{(g)} = S(s_{t-1}^{(g)}, V_t^{(g)})$, and set $\tilde{V}_t^{(g)} = (V_t^{(g)}, \Theta^{(g)}, s_t^{(g)})$.

4. Compute the second-stage weights $w_t^{(g)} \propto w_{t-1}^{(g)}\, p(S_t^*|V_t^{(g)}, \Theta^{(g)})/\lambda_t^{(g)}$.

Stochastic Volatility Jump Models

Jump Equity Models

DPS (2000) and EJP (2003): double jump model

$$d\log(S_t) = \mu dt + \sqrt{V_{t-}}dW_t^s + d\left(\sum_{j=1}^{N_t}\xi_{\tau_j}^s\right)$$

$$dV_t = \kappa_v(\theta_v - V_t)dt + \sigma_v\sqrt{V_{t-}}dW_t^v + d\left(\sum_{j=1}^{N_t}\xi_{\tau_j}^v\right)$$

where $\xi^s = \mu_s + \rho_J\xi^v + \sigma_s\varepsilon$, $\varepsilon \sim N(0,1)$, $\xi^v \sim \exp(\mu_v)$ and $N_t \sim Poi(\lambda t)$.

- Stochastic Volatility (SV) with $\lambda = 0$
- Jumps in returns (SVJ) with $\mu_v = 0$
- Double Jump model (SVCJ) with possibly correlated jumps between volatility and returns.

Applications

Three applications to illustrate the methodology

1. Stochastic Volatility (SV) estimates for the S&P500 and Nasdaq NDX100 equity indicies.

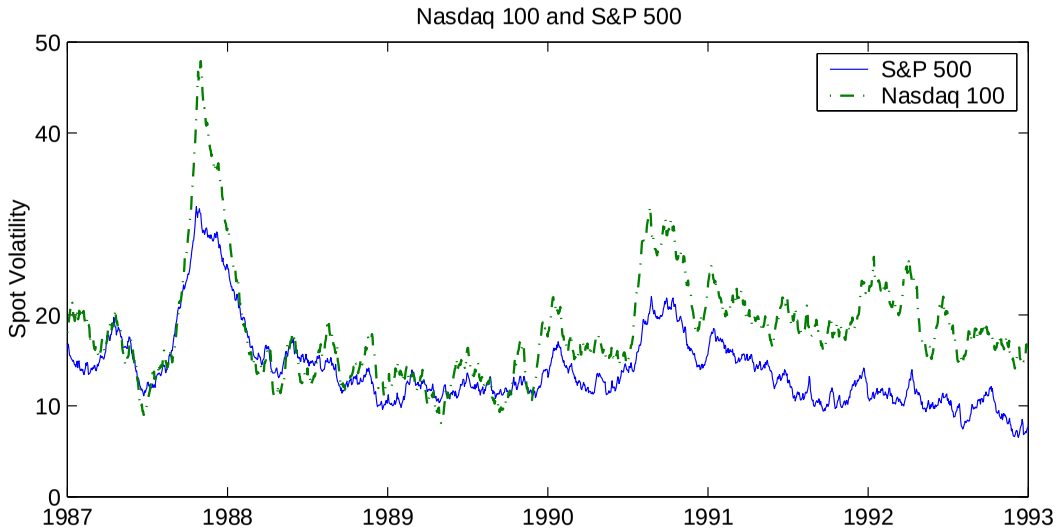
   Smoothed estimates: looking back and estimating historical vols

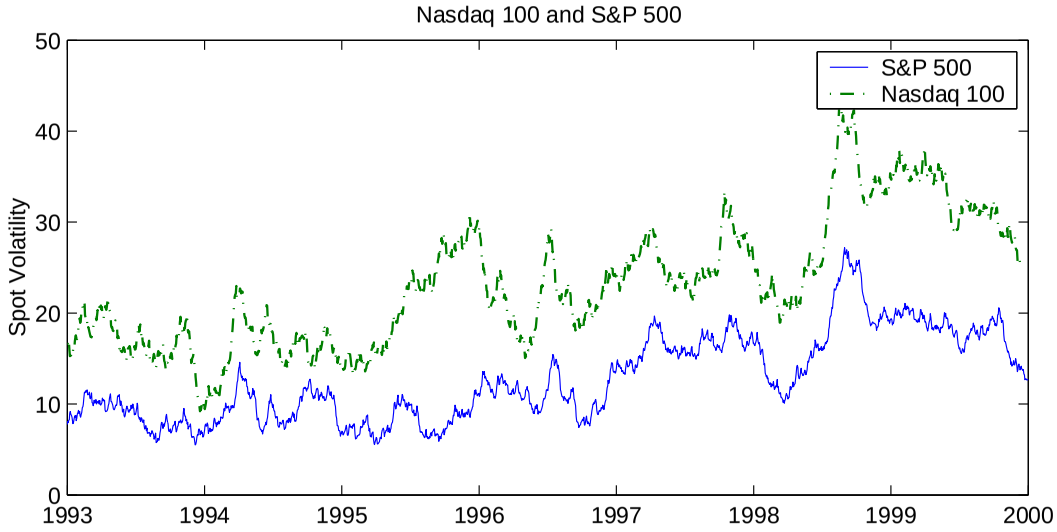
   Filtered estimates: providing on-line vols

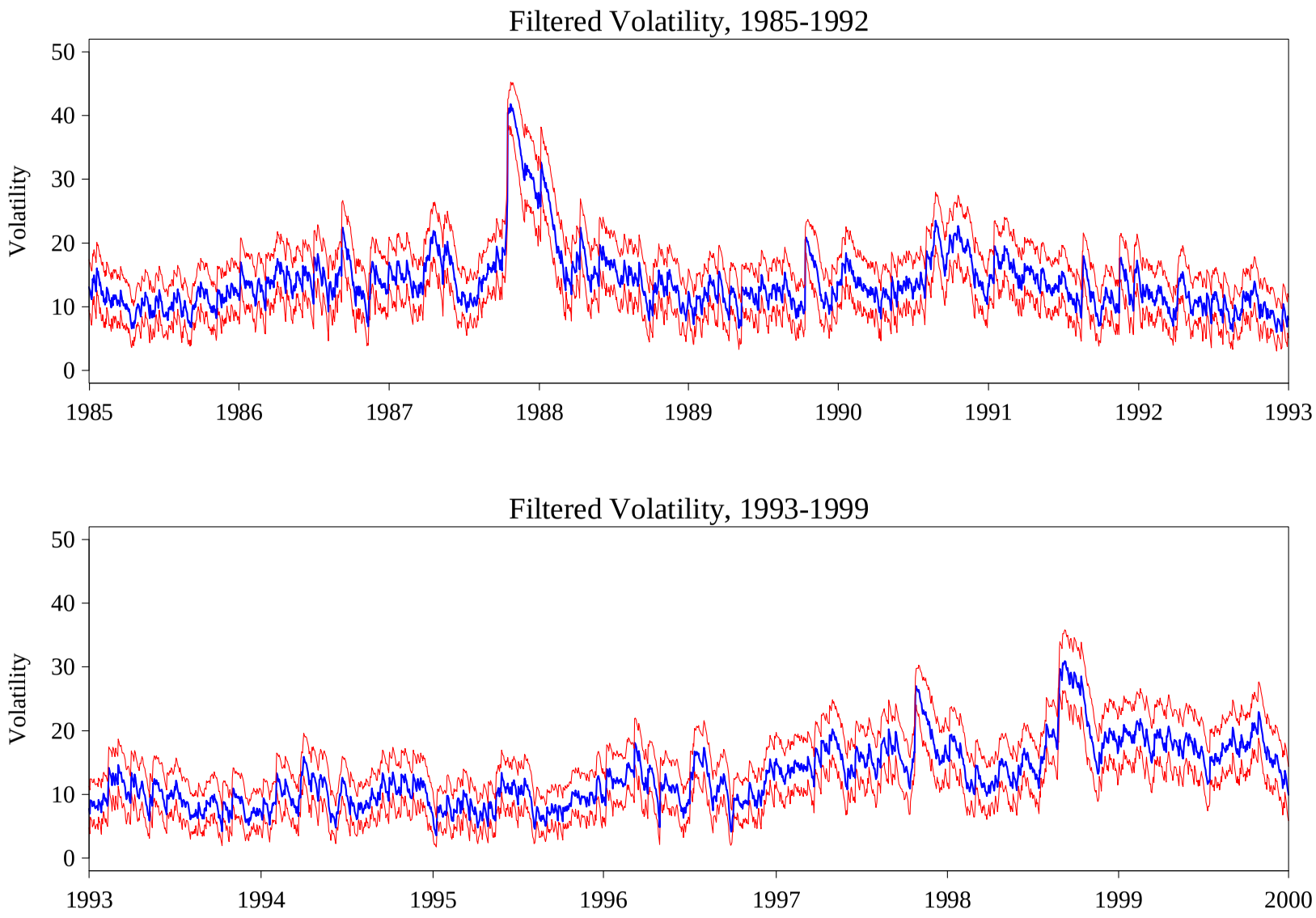
2. Simulated example to illustrate the time frequency of observation (Daily, Weekly and Monthly)

3. Extracting Jumps and Volatility: SVJ model with jumps in returns and SVCJ model with double jumps in return or volatility

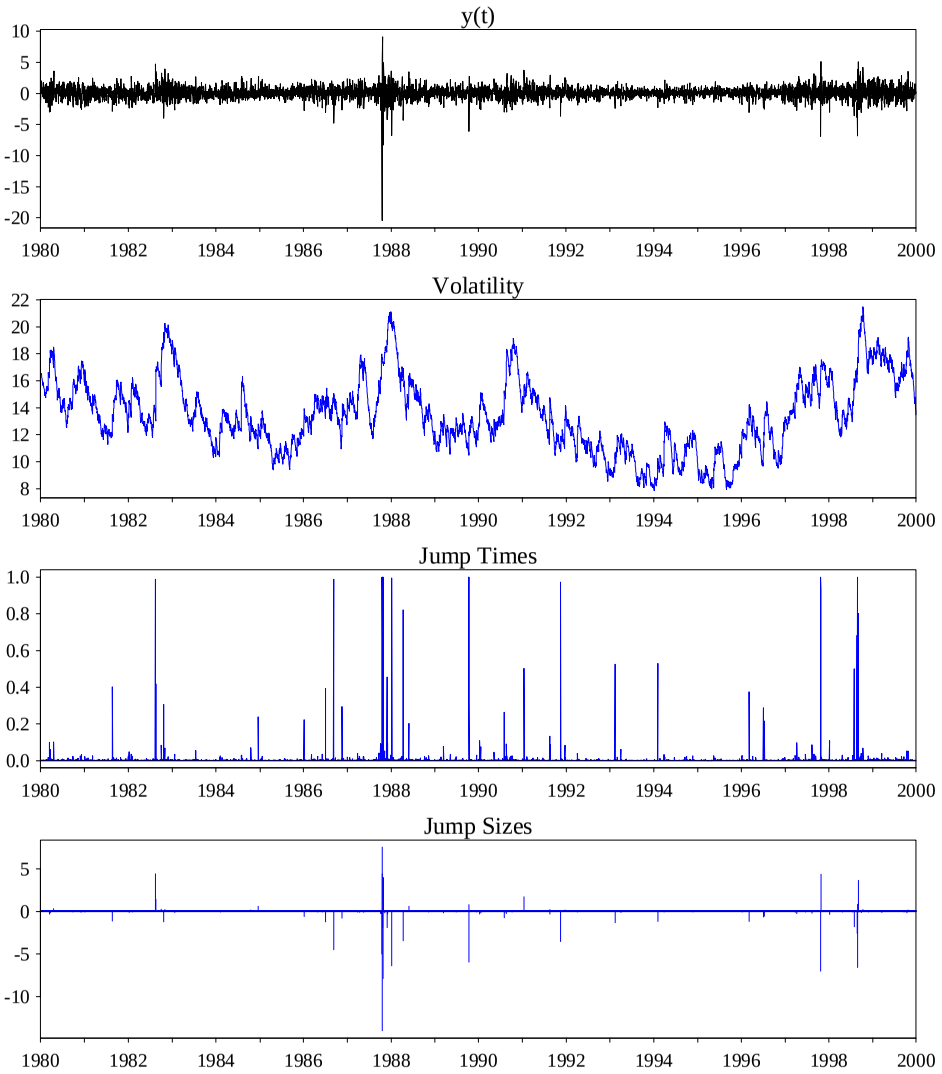
Ex 1: Smoothed Volatility

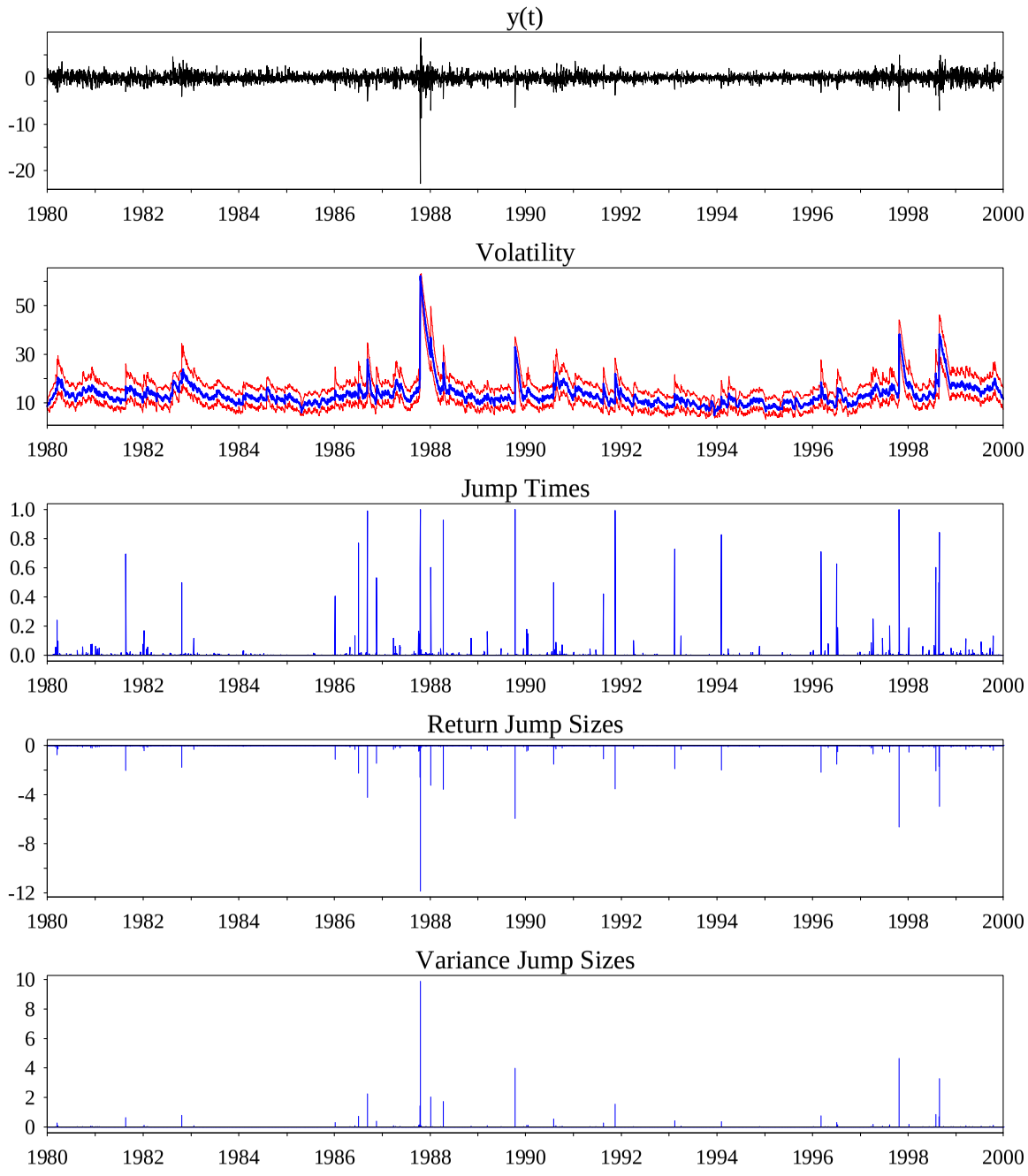





Filtered Volatility, 1985-1992
Volatility
50
40
30
20
10
0
1985
1986
1987
1988
1989
1990
1991
1992
1993
Filtered Volatility, 1993-1999
Volatility
50
40
30
20
10
0
1993
1994
1995
1996
1997
1998
1999
2000

Ex: SVJ Jumps


y(t)
10
5
0
-5
-10
-15
-20
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
Volatility
22
20
18
16
14
12
10
8
Jump Times
1.0
0.8
0.6
0.4
0.2
0.0
Jump Sizes
5
0
-5
-10


y(t)
0
-10
-20
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
Volatility
50
30
10
Jump Times
1.0
0.8
0.6
0.4
0.2
0.0
Return Jump Sizes
0
-4
-8
-12
Variance Jump Sizes
10
8
6
4
2
0

Sequential Parameter Learning

- Simulated Data Example

  1. Storvik's algorithm with $N = 100,000$ particles.
  2. Practical filter with $G = 250, I = 50$ and $k = 50$.

  Similar performance. No particle degeneracy.

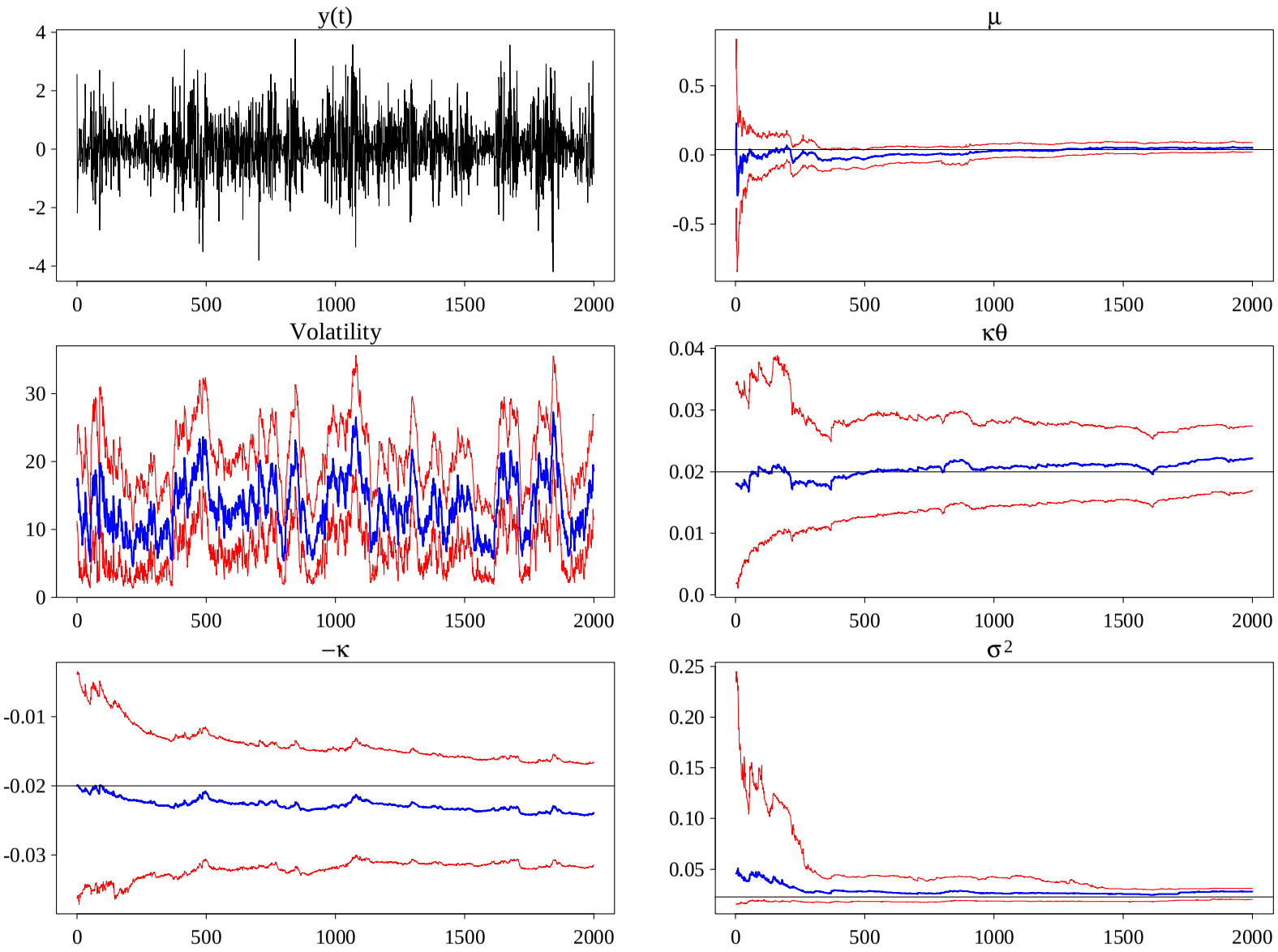
- S&P500 dataset 1980-2000

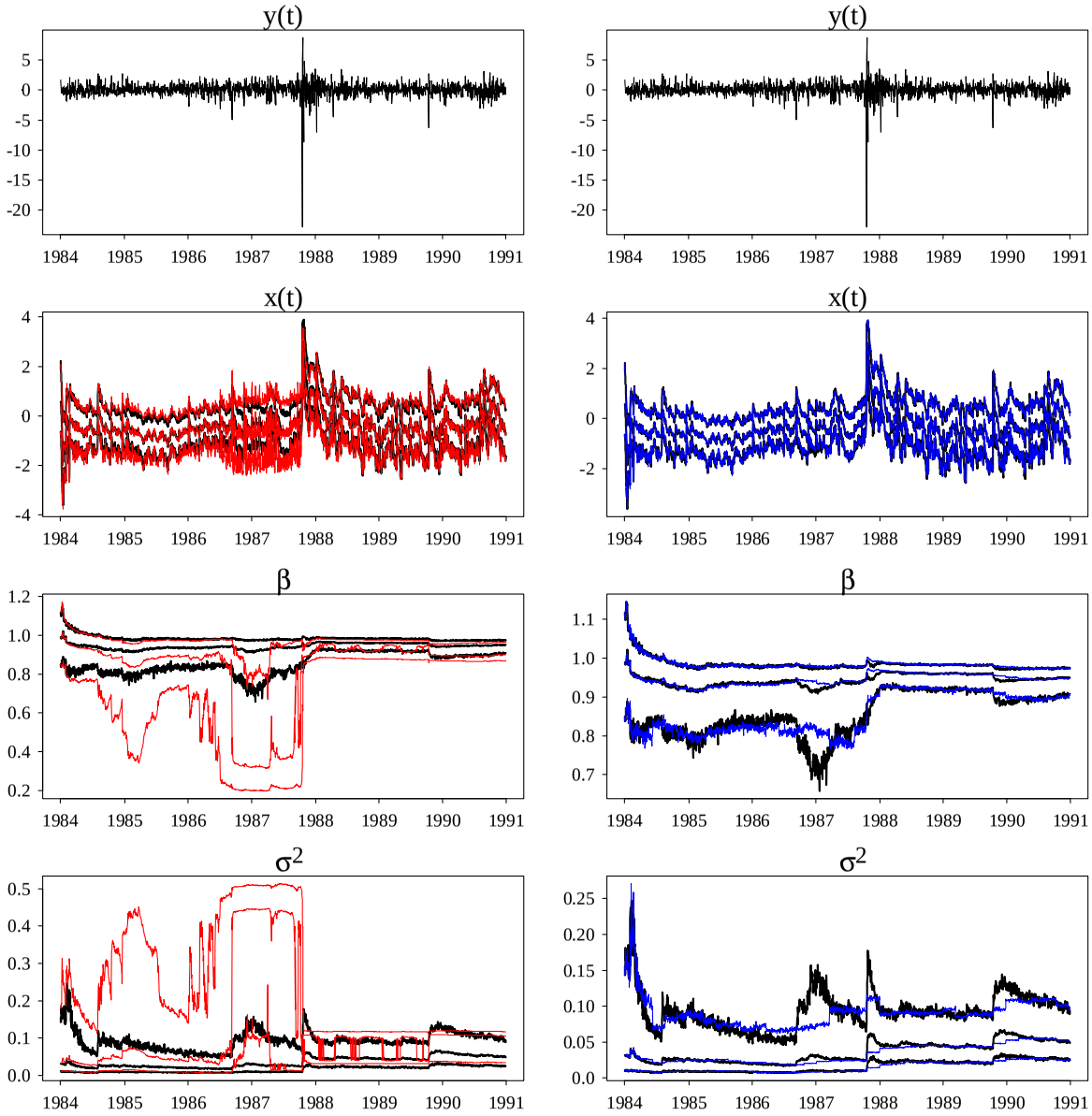
  Includes the 1987 crash (-20%).

  APF: degenerates

  Pract: reliable inference

  However, has difficulty in adapting to the "change" in the posterior on σ_v after the crash.


y(t)
μ
Volatility
κθ
−κ
σ²
0
500
1000
1500
2000


y(t)
5
0
-5
-10
-15
-20
1984
1985
1986
1987
1988
1989
1990
1991
y(t)
5
0
-5
-10
-15
-20
1984
1985
1986
1987
1988
1989
1990
1991
x(t)
4
2
0
-2
-4
1984
1985
1986
1987
1988
1989
1990
1991
x(t)
4
2
0
-2
1984
1985
1986
1987
1988
1989
1990
1991
β
1.2
1.0
0.8
0.6
0.4
0.2
1984
1985
1986
1987
1988
1989
1990
1991
β
1.1
1.0
0.9
0.8
0.7
1984
1985
1986
1987
1988
1989
1990
1991
σ2
0.5
0.4
0.3
0.2
0.1
0.0
1984
1985
1986
1987
1988
1989
1990
1991
σ2
0.25
0.20
0.15
0.10
0.05
0.0
1984
1985
1986
1987
1988
1989
1990
1991

Discussion

- MCMC provides a way of performing Sequential Filtering methods. Standard MCMC methods such as Gibbs sampling and Metropolis-Hastings algorithm have been used for estimation and smoothing
- Particle Filtering and Practical Filtering together with extensions using the Auxiliary Particle Filter have been widely applied.
- Incorporating Sequential Parameter Learning is achievable but harder.