モンテカルロフィルタと時系列データマイニング

高知大学数理情報科学科

本田研究室

小松　学

2007 年　2 月　8　日

要旨

現実社会には多様な時系列データが存在する。時系列データとは時間の経過とともに変化する現象の記録である。気温や気圧などの気象データ、株価データなどの経済データも時系列データの一例である。近年、情報技術の発展に伴い大量のデータが蓄積可能となってきている。そこで、大量のデータからパターン発見をするデータマイニングという分野が研究されている。データマイニングの手法を時系列データに適用することによりパターン発見や未来の観測値の予測への応用が期待できる。

そこで、時系列データマイニングの方法としてモンテカルロフィルタの有効性を検証する。モンテカルロフィルタとは北川(1996)高次元の非線形･非ガウス型の一般状態空間モデルのフィルタリングおよび平滑化の方法であり、確率分布を多数の粒子で近似するというものである。この方法によって非ガウス型の分布にも適用できるという利点がある。

本研究では簡単な線形･ガウス型の一般状態空間モデルのモデル式から観測データを生成し、観測値からその時の状態を推定し、推定した状態から観測値の予測を行い理論値である観測データと比較しその予測精度を検証した。さらにモンテカルロフィルタにおいて大量の時系列データから状態推定を行う効果を見るために平滑化区間についての実験を行った。平滑化区間を変更した場合に状態推定、観測値予測にどのような影響を与えるかについて検証を行った。

検証の結果、モンテカルロフィルタを時系列データマイニングの手法に用いた場合、状態推定に関しては平滑化区間を大きくすると粒子の分布が狭くなりこの影響によって予測精度は落ちることが分かった。一方観測値の予測精度については平滑化区間による影響は見られなかった。

一方、粒子による状態推定、観測値予測ともに長い時系列データに対して精度を落とす事なく追跡可能だった。このことから時系列データマイニングの手法としては状態や観測の自動予測という用途では有望である。また予測が単なる値ではなく確率密度分布で与えられるのも大きな利点である。

さらに非線形･非ガウス型のモデルに対する検証を行い実社会のデータに適用していくことが課題である。

目次

1　はじめに

近年、情報技術の進歩に伴い大量のデータを保存することが可能となった。大量のデータの保存が可能になったことで時系列データに関しても長い時刻幅で記録することが可能となった。ここでいう時系列データとは時間と共に変化する現象の記録のことである。現実社会には多くの時系列データが存在する。具体的には気温や、気圧、雨量などの気象データ、株価データなどの経済データも時系列データである。このような時系列データはグラフで表示することによってその特徴をとらえることができる。大量のデータからパターン発見を行う方法としてデータマイニングという分野が研究されているが、そのデータマイニングを時系列データに適用することで、そのデータからのパターンの発見や、未来の観測値の予測への応用が期待される。

そこで、時系列データマイニングの方法として今回、モンテカルロフィルタという手法を用いる。一般状態空間モデルにおいてシステムモデル、観測モデルの 2 式から生成されるデータに対してモンテカルロフィルタによる状態推定と観測値の予測を行いその有効性を検証する。

また、モンテカルロフィルでは一般に 30 程度以下にすべきであるといわれている平滑化区間について平滑化区間を変化させたときの状態推定、観測値の予測への影響を調べる。さらにその結果から時系列データマイニングへの適用性を検証する。

２　モンテカルロフィルタの概要

２．１　モンテカルロフィルタについて

モンテカルロフィルタとは高次元の非線形･非ガウス型の一般状態空間モデルにおける、フィルタリングおよび平滑化の方法として 1996 年、北川源四郎氏によって提案されたものである[1]。その特徴として、アルゴリズムが比較的簡単で実装が容易で様々な問題に適用できるということがあげられる。

モンテカルロフィルタの基本的な考え方は非常にシンプルで条件付き分布を多数の粒子で近似表現するというものである。この方法であれは非ガウス型の分布も多数の粒子で近似できる。そして、その粒子を追跡することで複雑な時間発展システムを解析することができる。

モンテカルロフィルタという言い方以外にも、ブートストラップフィルタ、粒子フィルタ[2]とも呼ばれている。

モンテカルロフィルタを使用する利点をあげてみると、まず一般状態空間モデルを使用することで一見複雑な動きをしているようなデータでもシステムモデル、観測モデルを用いることで一様に扱うことが可能である。さらに非ガウス型の確率分布に対しても適用が可能であることや、予測値を確率分布の形で表現することができるなどがあげられる。

以下ではモンテカルロフィルタの概要について北川(2005)[3]に基づいて述べていく。

２．２　一般状態空間について

２．２．１　一般状態空間モデルとモデル式

一般状態空間とは時刻tにおける観測値をy_tとしこの観測値を表現するのに必要なベクトルをx_tとする。このx_tを状態ベクトルという。この状態ベクトルx_tと観測値y_tは以下の式で表すことができる。

$$x_t = F_t x_{t-1} + G_t v_t \qquad (2.1)$$

$$y_t = H_t x_t + w_t \qquad (2.2)$$

式（2.1）はシステムモデルといわれる。時刻tの状態x_tは１期前の状態の関数として記述される。式（2.2）は観測モデルといわれる。観測値y_tが状態x_tの関数として記述される。ただし、システムモデルにはシステムノイズv_t、観測モデルには観測ノイズw_tと呼ばれるノイズがある。

２．２．２　鎖状グラフィカルモデル

上記のシステムモデル、観測モデルとx_t、y_tは下の図の様に表現できる。

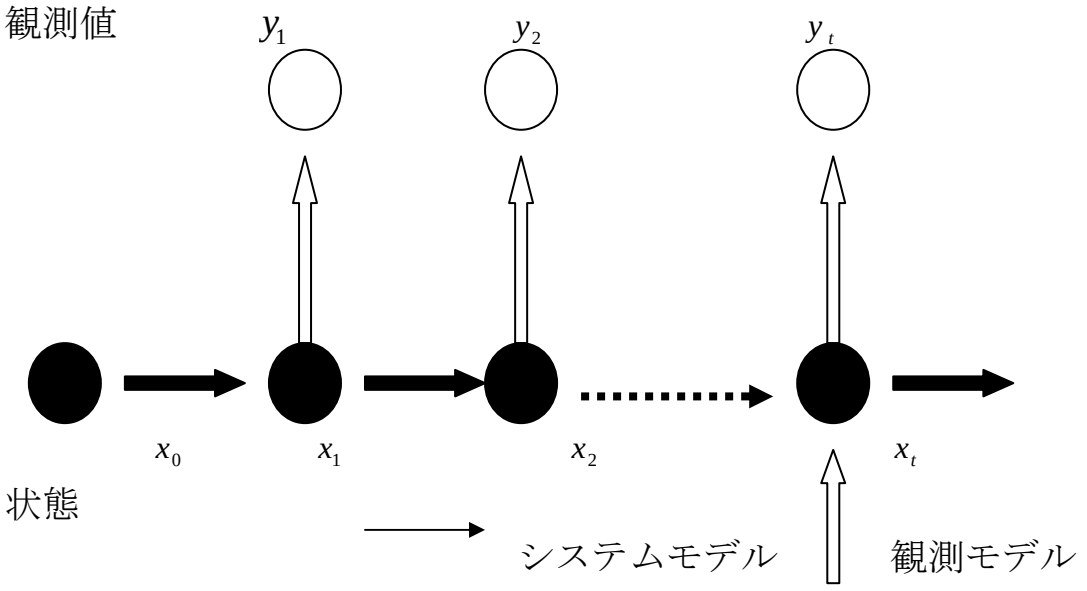


図１　鎖状グラフィカルモデル。

図１において、左下の状態は初期値x_0からスタートしシステムモデルによって変化していく。そして、それぞれの状態x_tから観測モデルを用いてy_tを求める。通常、観測者には観測値y_tだけが観測され、状態x_tは観測できない。そこで、モンテカルロフィルタでは観測値y_tから観測できない状態x_tを推定する。

２．２．３ 確率密度分布の粒子による表現ついて

モンテカルロフィルタでは状態x_t、y_t観測値の値が図 2 の赤で表されるようにデルタ関数で表される。そしてこれを確率密度分布（図 2 の曲線）で表す事ができる。

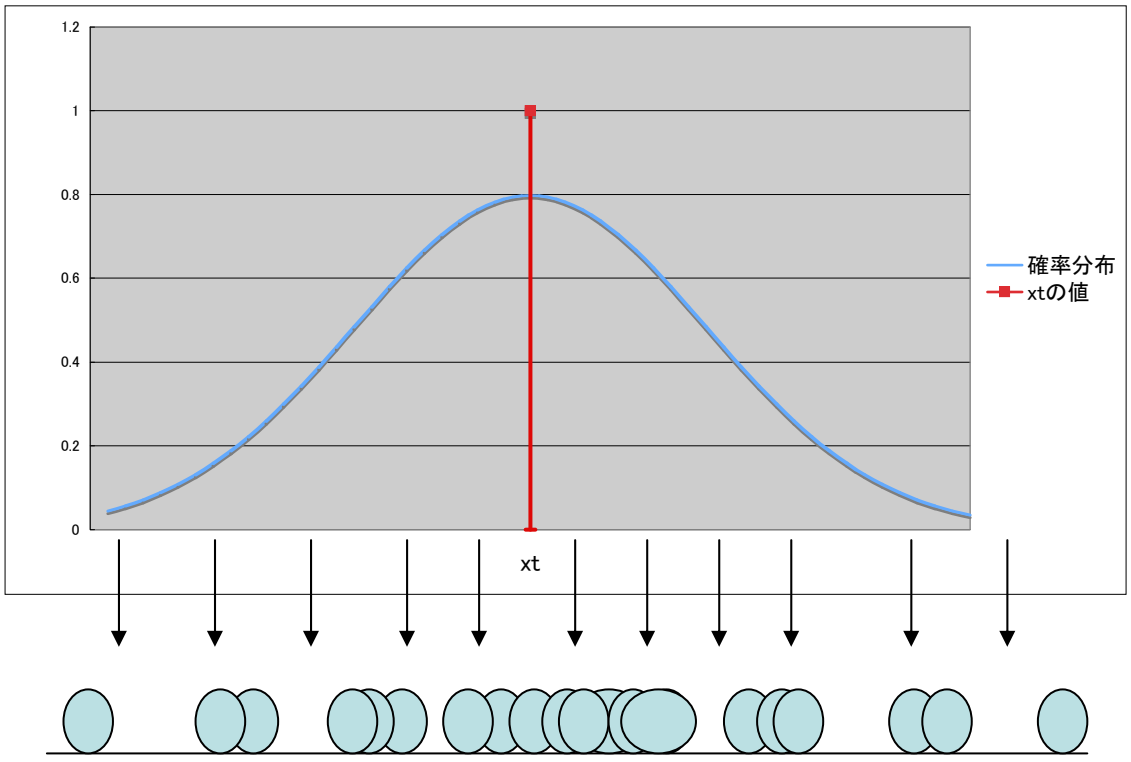


図２　粒子による確率分布の表現と粒子による表現。

この確率分布をモンテカルロフィルタでは粒子の頻度で表す。粒子による近似を実現でき、粒子はそれぞれx_tの値を持ち、確率の高いところは粒子の頻度が高く確率の低いところは粒子の頻度も低い。

２．３　状態推定

２．３．１　状態推定について

状態空間モデルにおいて重要なことは観測値y_tからその状態ベクトルx_tを推定することである。状態推定ができれば、未来の予測など時系列データの解析に関する問題が解決される。観測値$y_{1:t}$が与えられた時、状態x_tを推定するにはx_tの条件付き確率$p(x_t \mid y_{1:t})$を求める。状態推定はどの時刻の観測データからどの時刻の状態を推定するかによって以下の 3 つに分類できる。

２．３．２　３つの条件付き分布

状態推定はどの時刻の観測値からどの時刻状態を推定するかで 3 つに分けられる。予測分布$p(x_t \mid y_{1:t-1})$、フィルタ分布$p(x_t \mid y_{1:t})$、平滑化分布$p(x_{t-L} \mid y_{t-L:t})$である。予測分布は時刻$t-1$までの観測値に基づく時刻$t$の状態の確率分布をあらわす。フィルタ分布は時刻$t$までの観測値に基づく時刻$t$の状態の確率分布。平滑化分布は時刻$t-L$から時刻$t$までの観測値に基づく時刻$t-L$の状態の確率分布を表す。

平滑化区間Lは可変だが、一般にあまり大きくせず、30 程度以下にすべきであるといわれている。その妥当性についても実験にて確認していく。

２．３．２　条件付き分布と漸化式

モンテカルロフィルタとアルゴリズムの別の呼び方の粒子フィルタ[2]では条件付き確率に関して以下のような関係性を述べている。

上記の３つの条件付き分布は以下のような漸化式の関係が成り立つ。

一期先予測

$$p(x_t \mid y_{t-1}) = \int p(x_t \mid y_{1:t}) p(x_{t-1} \mid y_{1:t-1}) dx \qquad (2.3)$$

フィルタ

$$p(x_t \mid y_t) = \frac{p(y_t \mid x_t)p(x_{t-1} \mid y_{1:t-1})}{p(y_t \mid y_{1:t})} \tag{2.4}$$

固定区間平滑化

$$p(x_{t-L} \mid y_{t-L:t}) = p(x_t \mid y_{1:t}) \int \frac{p(x_{t+1} \mid y_{1:T})p(x_{t+1} \mid x_t)}{p(x_{t+1} \mid y_{1:t})} dx \tag{2.5}$$

上記の漸化式を模式図で表すと以下のようになる。

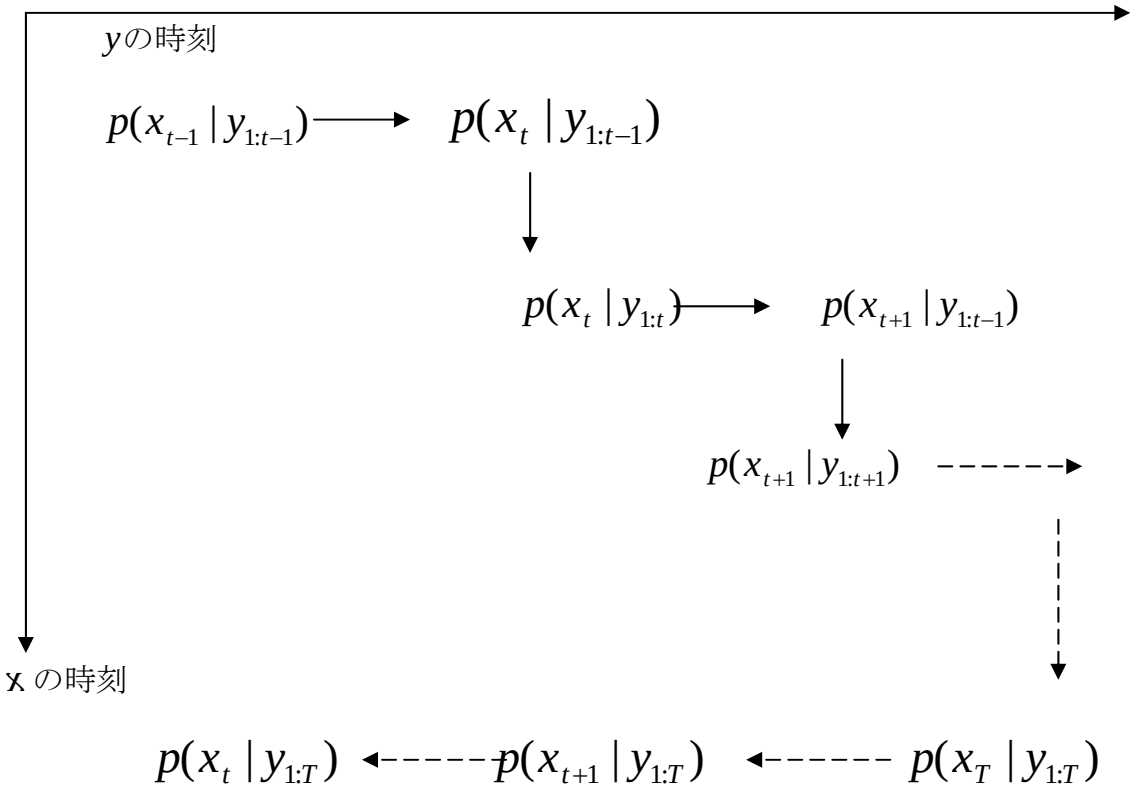


図 3　漸化式の模式図。

図 3 において、右の矢印が予測、下向きの矢印がフィルタリングの操作で左向きの矢印が平滑化の操作を表す。予測とフィルタリングとを繰り返し粒子を予測する。そして、十分先の時刻までのデータを得て過去の状態を平滑化によって改めて推定する。

２．４　モンテカルロフィルタのアルゴリスムについて

２．４．１　モンテカルロフィルタの基本的なアイディア

モンテカルロフィルタ条件付き確率を粒子によって近似する。モンテカルロフィルタによる近似は図 4 の様な非ガウス型の確率分布にも適用ができる。このとき粒子の数はモデルの複雑さや必要な精度によって異なる。モンテカルロフィルタは現在の粒子から一期先の予測粒子を生成し、それをリサンプリングしフィルタの実現値を生成する。

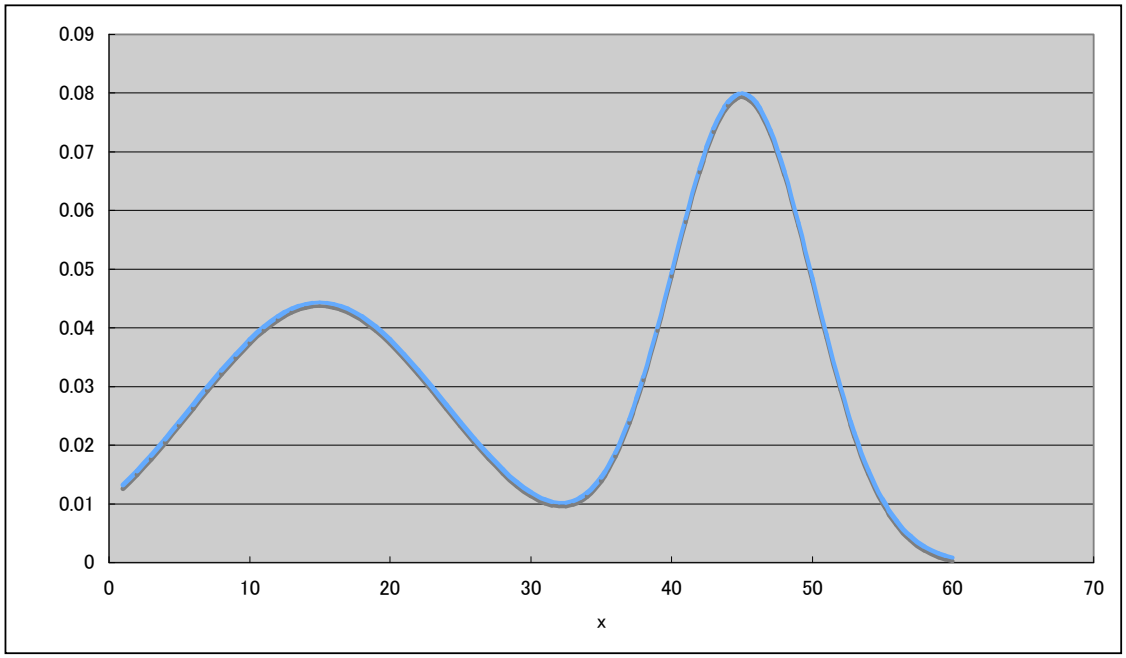


図 4　非ガウス分布の例。

２．４．２　予測

予測ステップでは現在の状況x_{n-1}の条件付き分布$p(x_{n-1}\mid y_{1:n-1})$の実現値と見なせる粒子$\{f_{n-1}^{(1)},\ldots,f_{n-1}^{(m)}\}$とシステムノイズ$v_n$の実現値と見なせる粒子$\{v_n^{(1)},\ldots,v_n^{(m)}\}$を用いて式（2.6）のように求める。

$$p_n^{(j)} = Ff_{n-1}^{(j)} + v_{n-1}^{(j)} \tag{2.6}$$

これによって求められた$p_n^{(j)}$が予測粒子となる。

２．４．３　フィルタリング

フィルタリングでは、まず観測値y_nに基づく予測した粒子pの尤度を計算する。

$$\alpha_n^{(j)} = p(y_n \mid p_n^{(j)}) = r(G(y_n, p_n^{(j)}))\left|\frac{\partial G}{\partial y_n}\right| \tag{2.7}$$

ただしGはHの逆関数、rは観測ノイズwの密度関数とする。

次に求めた尤度で予測粒子pをリサンプリングする。これによって求められたf_nはフィルタ分布からの実現値と見なせる。

２．４．４　平滑化について

平滑化とはフィルタが時刻nまでの観測値を用いてx_nを推定しているのに対し平滑化は過去の観測値を用いてx_nの推定を行っている。したがって、一般には平滑化を行った方がよりよい精度の状態推定が行えることになる。平滑化を行う際に平滑化区間をどの程度にするのかという問題がある。一般に 30 程度以下にとどめておくべきであるといわれている。つまり、あまりにも時刻の離れた観測値を考慮した状態推定は有効的でないという事である。

２．４．５　アルゴリズムの概略図

ここまでの流れを図にすると図 5 のようになる。

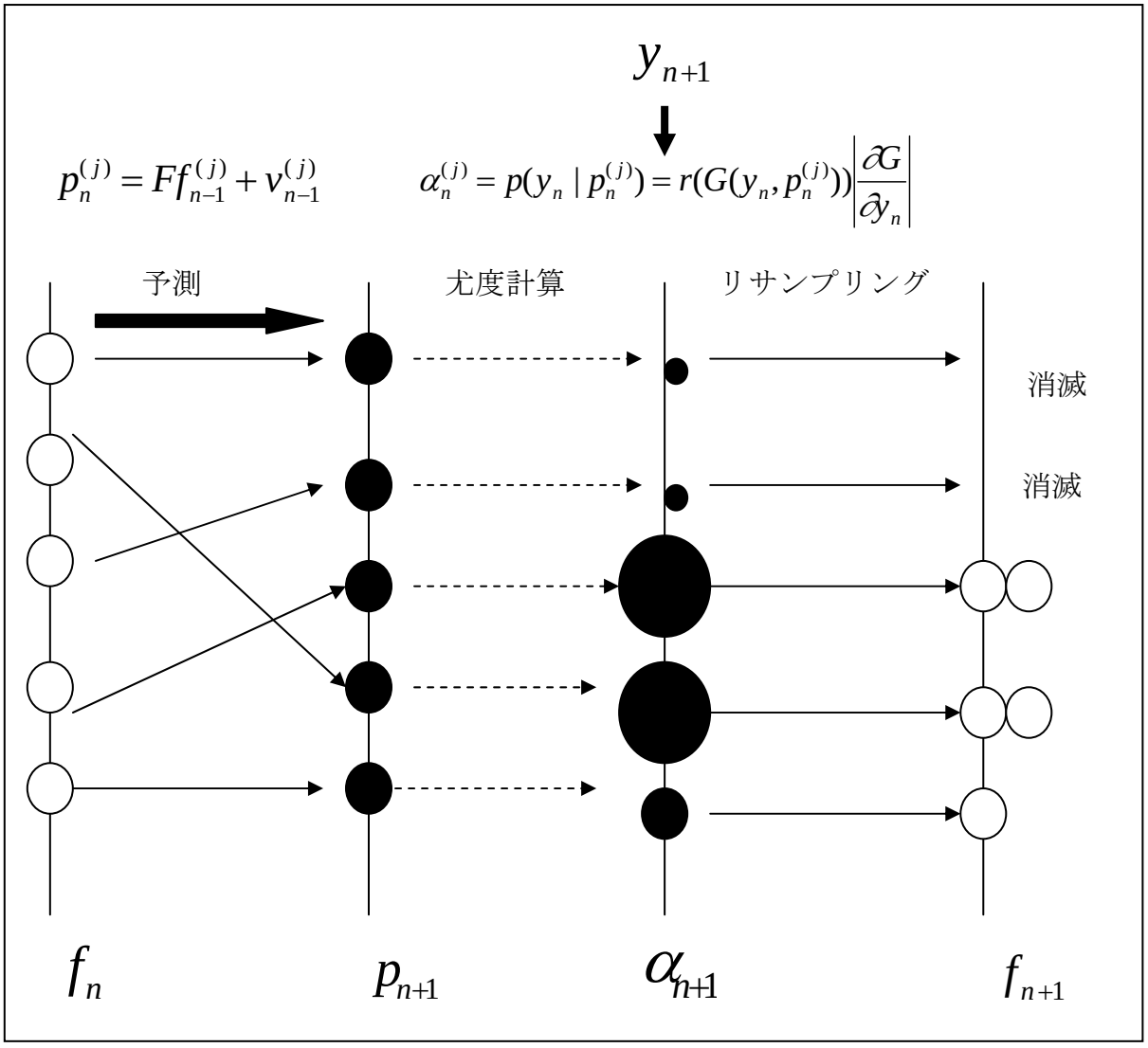


図 5　アルゴリズムの概略図。

図 5 において、f_nから予測粒子p_{n+1}を予測する。そして、それぞれの予測粒子に関して観測値y_{n+1}を用いて尤度α_{n+1}の計算を行う。求めた尤度に従って予測粒子p_{n+1}をリサンプリングする。リサンプリングによって時刻$n+1$の状態の粒子f_{n+1}を確定する。

３　実験

３．１　実験内容

前節までで一般状態空間モデルや、モンテカルロフィルタのアルゴリズムを説明してきたが、実際にモンテカルロフィルタによる時系列解析において、その精度を調べる。そして平滑化に関して平滑化区間は一般に 30 程度以下にすべきである、とされているが実際に平滑化区間を変化させた場合、状態推定、観測値予測の精度にどのような変化がおこるか検証を行う。

３．２　実験に用いたデータについて

今回、実験に際し使用したデータは以下のようなモデル式から与えられる。

$$x_n = x_{n-1} + v_n \tag{3.1}$$

$$y_n = x_n + w_n \tag{3.2}$$

式（3.1）はシステムモデル、式（3.2）は観測モデルである。
ただし、システムノイズv_n、観測ノイズw_nはそれぞれ、$N(0,1)$の正規分布に従うものとする。

この２式のモデル式から生成された観測データから、未知の状態の状態推定を行う。

観測データの１例をグラフで表すと図 6 の様なグラフになる。

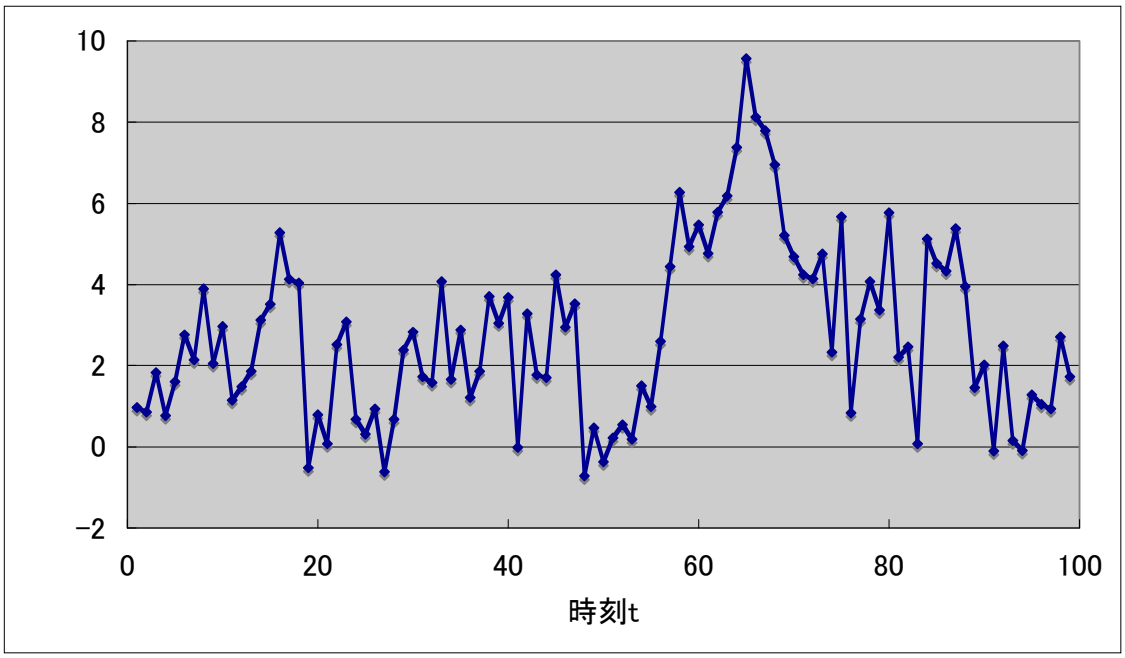


図 6　観測データ。

このデータの背景には観測データを生成した未知の状態 x_n のグラフが存在する。

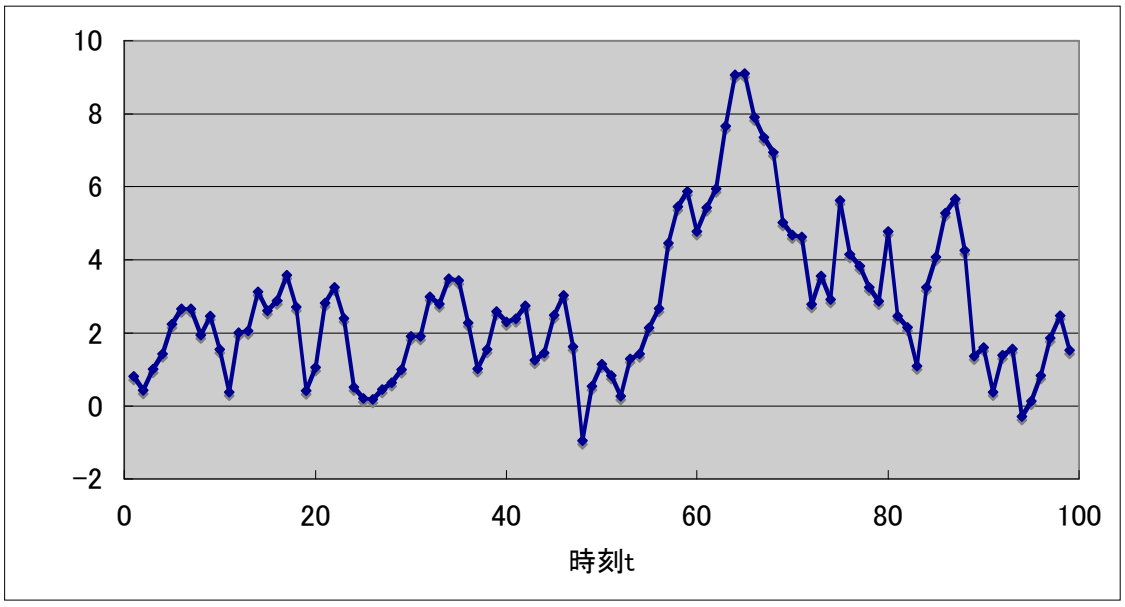


図 7　観測データを生成する状態のグラフ。

観測値 y_n と状態 x_n のグラフが一見同じように見えるがこれはモデル式が単純であるからであり、2 つのグラフは異なったグラフである。

３．３　実験条件について

表１　　実験条件

項目	実験条件
ステップ数	100
粒子数	1000
平滑化区間	0,15,30,50

表２　評価基準

評価基準	予測の平均値±標準偏差の範囲内に理論値があるかどうか
評価区間	1ー49 ステップ
その他	20 試行の平均で評価

３．４　状態推定の結果

３．４．１　各平滑化区間における状態推定の結果

平滑化区間 0 の結果例は以下のようになる。
（各平滑化区間における 20 試行の結果は付録 1 を参照。）

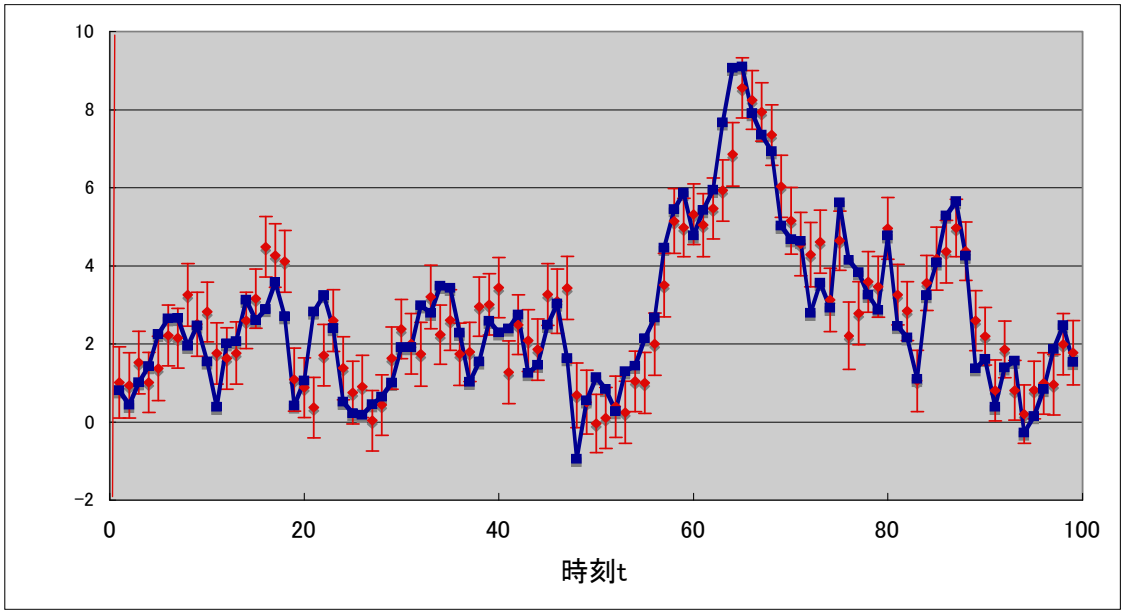


図 8　平滑化区間=0 の状態推定の結果例。
ここで青いグラフは理論値、赤いエラーバー付きシンボルが予測分布を表す

図 8 は平滑化区間 0 の状態推定の結果例を表しており、赤いエラーバー付きのシンボルが粒子の分布、青いシンボルが理論値を表しています。赤い点が粒子の分布の平均値、エラーバーがその標準偏差を表しています。この結果を定量的に評価するために評価基準を導入する。粒子の分布を平均値±標準偏差の領域に理論値があれば成功、なければ失敗として評価区間内での成功率を調べる。さらに 20 試行の平均値を評価する。

グラフを見ると粒子の分布は初期値では考えられる範囲に広く分布させている。次の時刻からは粒子の分布も狭くなり、その後は理論値を大きく外す事なく一定の精度で追跡しているのが分かる。

以降はそれぞれの平滑化区間 0,15,30,50 についての結果例を同じ形式のグラフを用いて評価していく。

推定、予測の成功　　推定、予測の失敗

図 9　状態推定、予測の成功、失敗例。

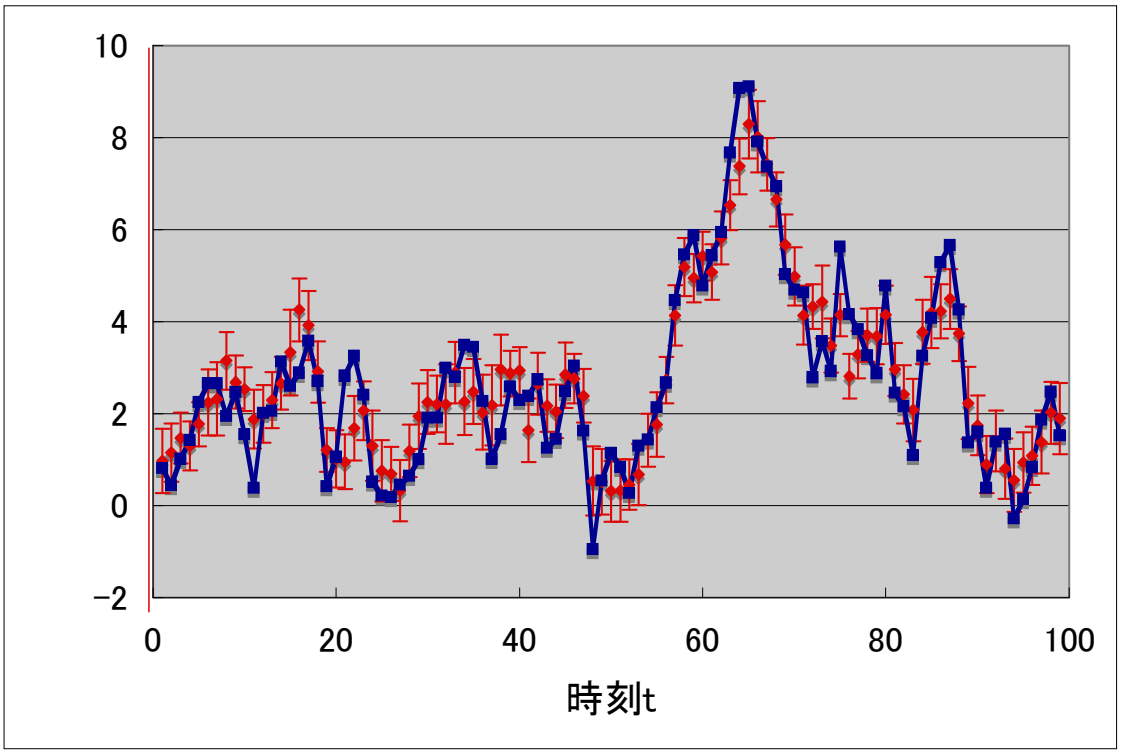


図 10　平滑化区間 15 の状態推定の結果例。各シンボルは図 8 参照。

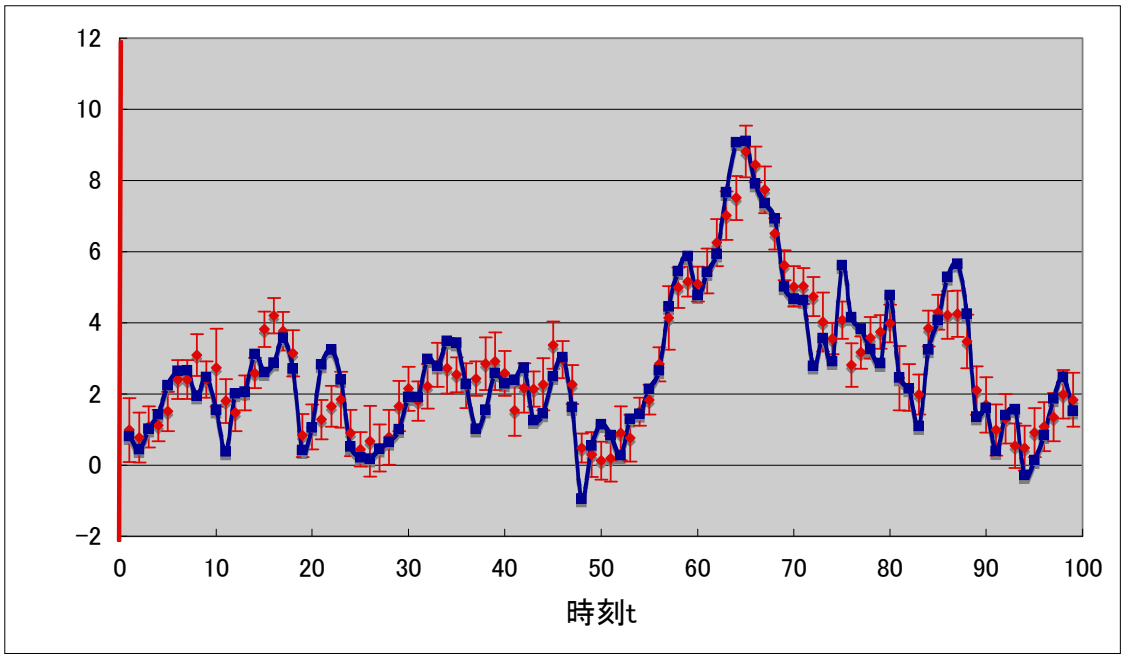


図 11　平滑化区間 30 の状態推定の結果例。各シンボルは図 8 参照。

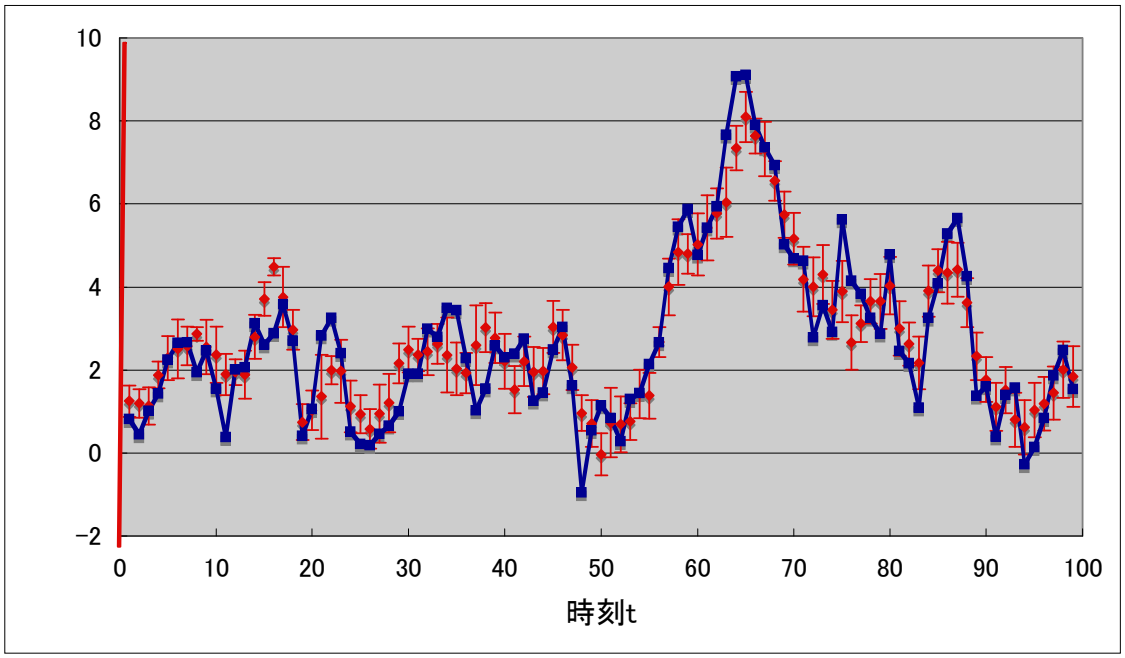


図 12 平滑化区間 50 の状態推定の結果例。各シンボルは図 8 参照。

３．４．２　状態推定の精度

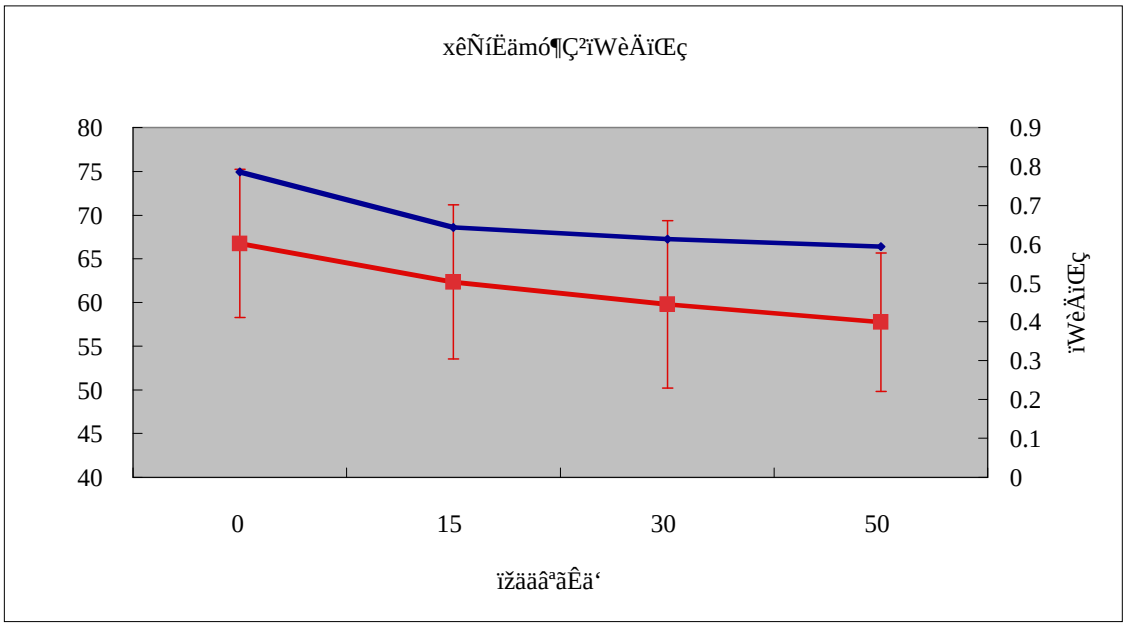


図 13　推定確率と標準偏差の平滑化区間に対する推移。青い線は粒子の分布の標準偏差、赤線は予測確率、エラーバーは予測確率の標準偏差を表す。

グラフを見ると赤いグラフ、青いグラフともに平滑化区間を大きくするにつれて下がっていっているのが分かる。つまり平滑化区間を大きくすると状態推定の確率が下がっていく事がグラフから分かる。そしてそのときの粒子の分布の標準偏差も小さくなっている。

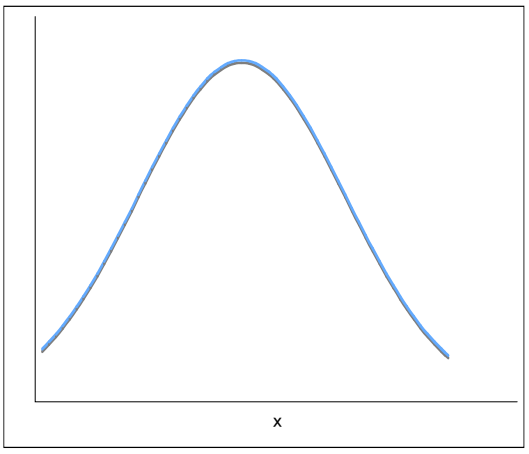
さらに状態推定の確率、確率の標準編差、粒子の分布の標準偏差を表にする。

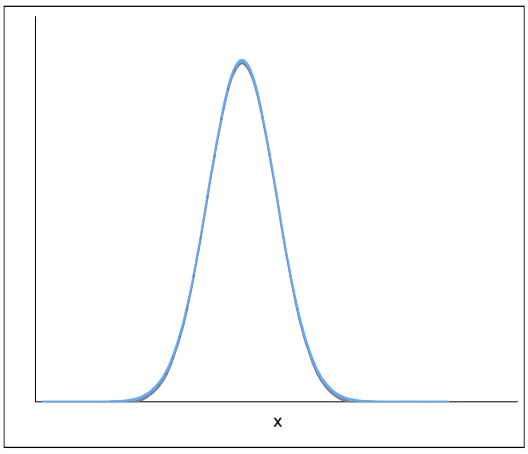
表 3　状態推定の結果。

平滑化区間 L	x の予測確率(%)	粒子分布の標準偏差の平均
0	66.76±8.47	0.786
15	62.35±8.83	0.644
30	59.79±9.60	0.613
50	57.75±7.92	0.590

表 3 は各平滑化区間とそのときの予測確率、確率の標準偏差、粒子の分布の標準偏差を表す。モンテカルロフィルタによる状態推定は平滑化区間 0 においては 66.76%を平均に±8%の幅をとり、推移しているのがわかる。平滑化区間を大きくするに伴って状態推定の確率も 62.35%、59.79%、57.75%と落ちていっている。どの区間も平均から±8ー9%の範囲で変動している事も分かる。

次に粒子の分布の標準偏差について考察してみる。粒子の分布の標準偏差が小さくなっているという事は粒子の分布が狭い範囲に偏っているという事である。





粒子の分布が大きい場合の分布　　粒子の分布が小さい場合の分布

図 14　標準偏差と粒子の分布。

平滑化区間が大きくなると推定確率は低下しているが、粒子の分布の標準偏差を比べると 0.78 から 0.59 まで小さくなっているが分かる。つまり状態推定において、確率が落ちているのはその粒子の分布の標準偏差が小さくなり粒子に偏りが見られるためであるといえる。

平滑化区間は 30 以下にすべきといわれているが今回のモデルにおいては平滑化区間が 10 以上になると、状態推定の予測精度はあまり良くないといえる。

つまり平滑化区間を大きくしすぎない方がよいのは粒子の分布に偏りが生じ精度が落ちるからであるといえる。

３．５　観測値予測の結果

３．５．１　各平滑化区間における観測値予測の結果

状態推定と同じように平滑化区間 0,15,30,50 の結果例の結果を示す。

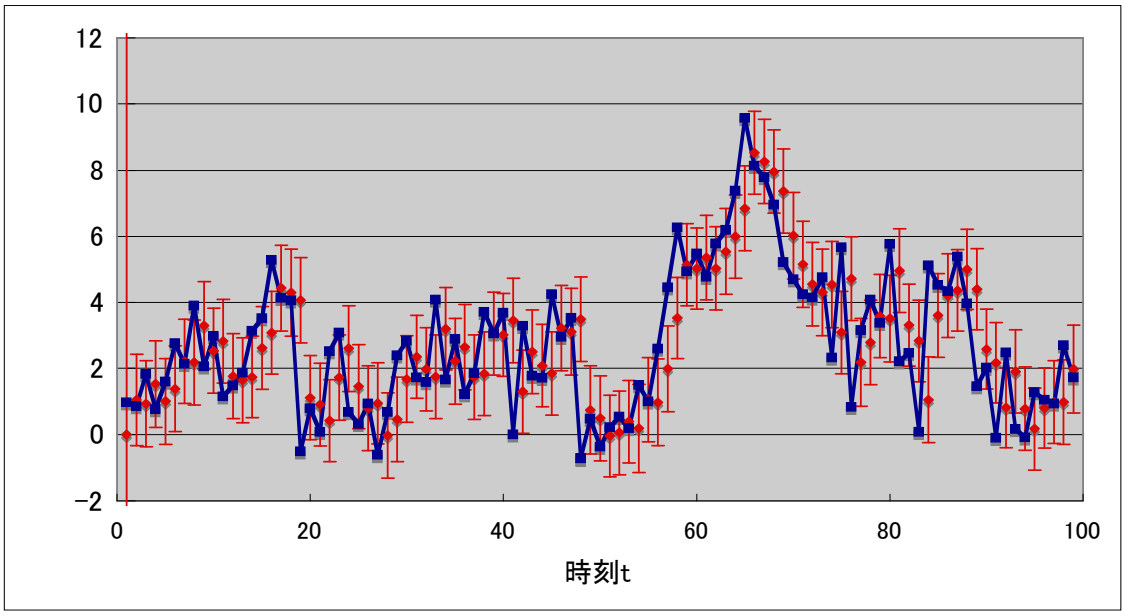


図 15　平滑化区間 0 の観測値予測の結果例。青い線は理論値、赤いシンボルは

粒子の平均値とエラーバーは粒子の分布の標準偏差を表す。

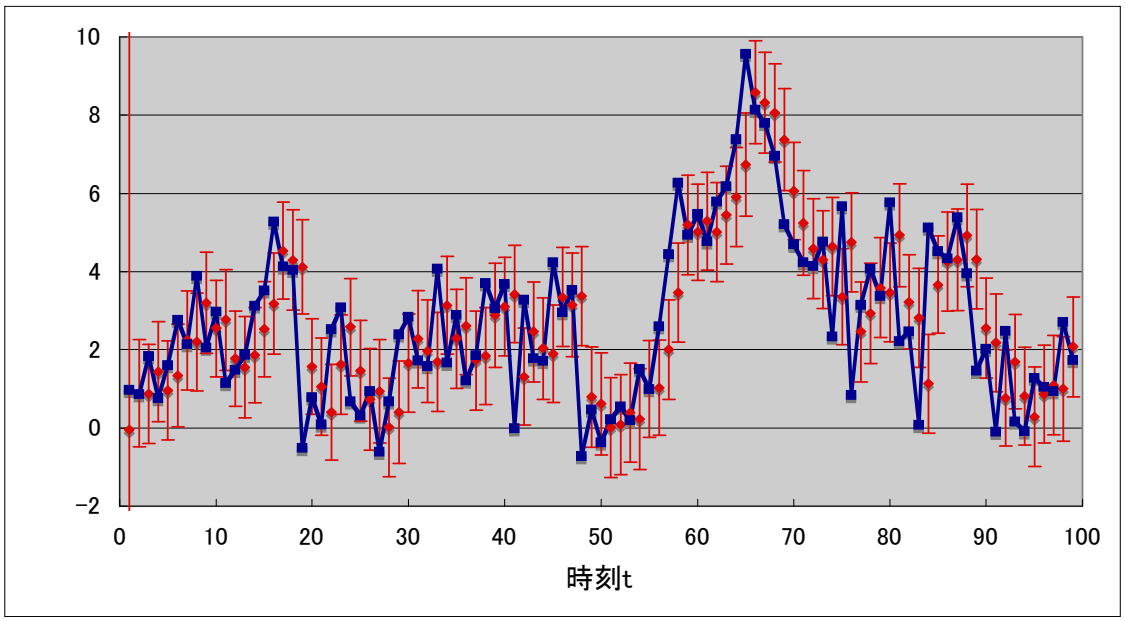


図 16　平滑化区間 15 の観測値予測の結果例。各シンボルは図 15 参照。

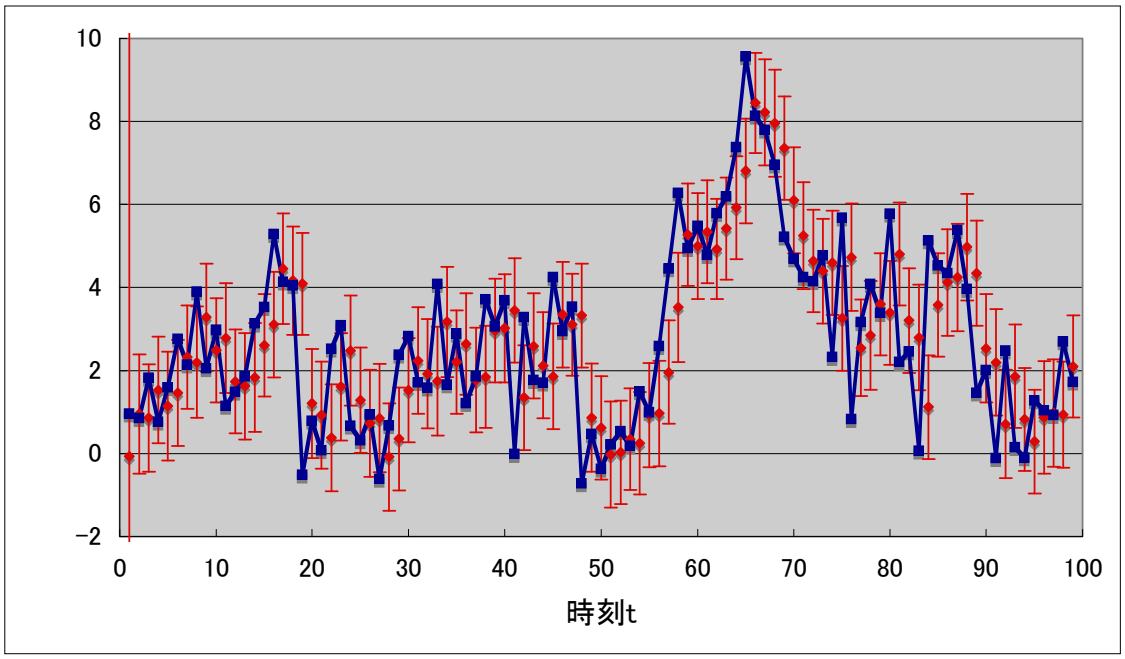


図 17　平滑化区間 30 の観測値予測の結果例。各シンボルは図 15 参照。

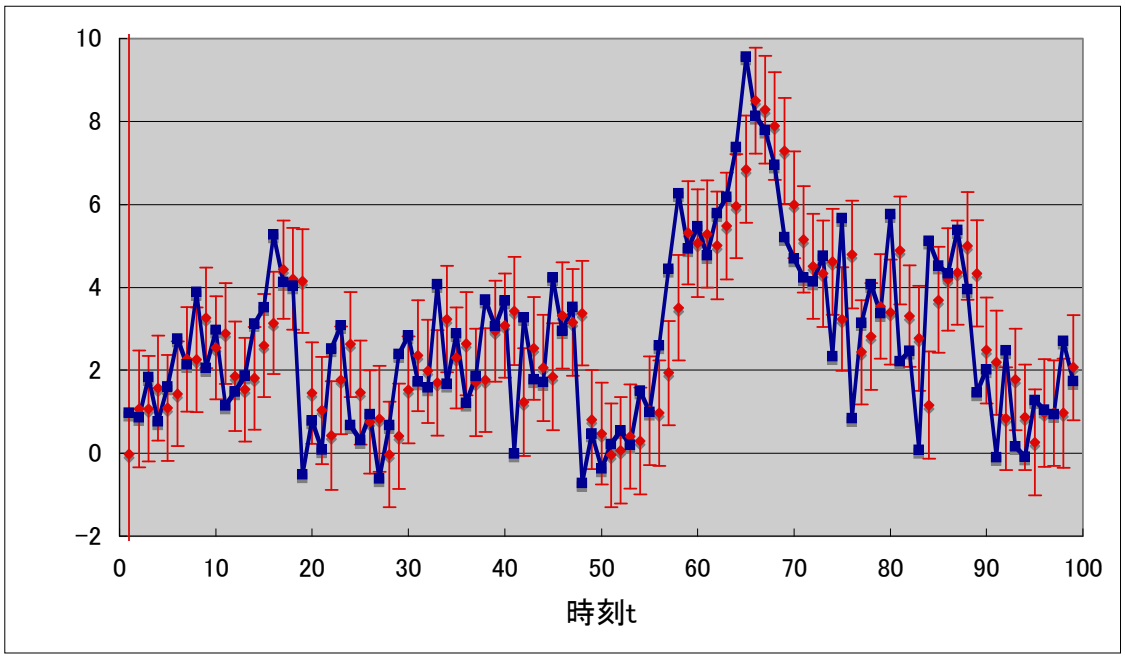


図 18　平滑化区間 50 の観測値の予測の結果例。各シンボルは図 15 参照。

３．５．２　観測値予測の精度

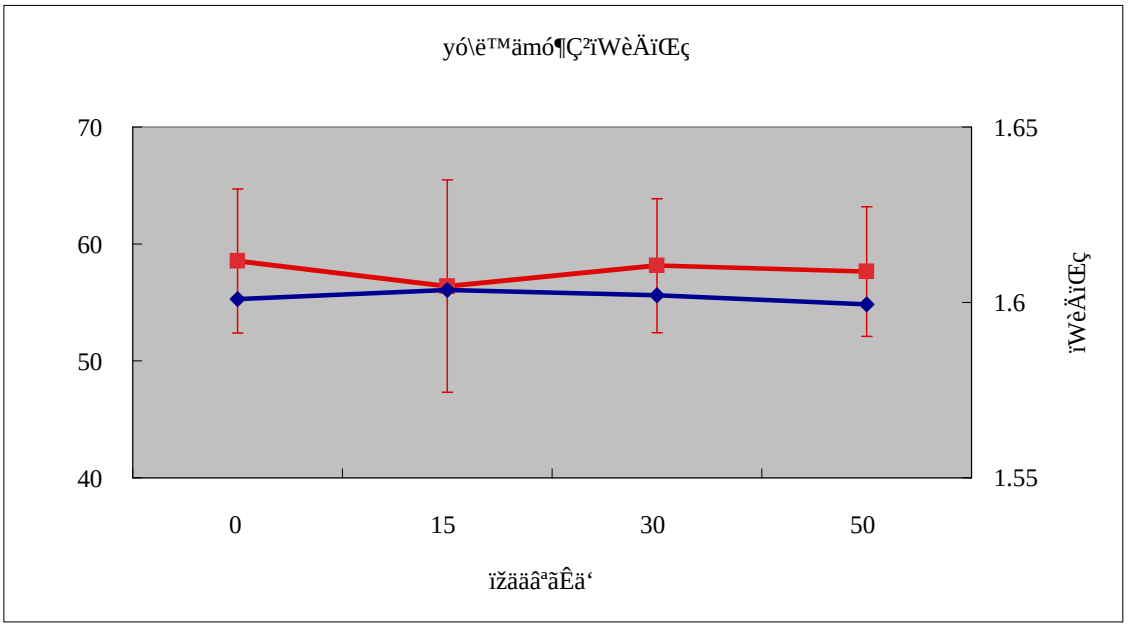


図 19　予測確率推と標準偏差の平滑化区間に対する推移。

図の形式は状態推定を同じで、赤いグラフが観測値の予測確率、青いグラフ

が粒子の分布の標準偏差を表す。観測値の予測確率は確率の平均±標準偏差だけ変化します。状態推定のグラフと違い予測確率、粒子の標準編差はほぼ横ばいに推移しているのが分かる。

平滑化区間における観測値予測確率、確率の標準編差、粒子の分布の標準偏差を表にまとめる。

表 4　観測値予測結果のまとめ。

平滑化区間	y の予測成功確率	粒子分布の標準偏差
0	58.56±6.17	1.601
15	56.40±9.08	1.604
30	58.15±5.72	1.602
50	57.65±5.55	1.600

表 4 は各平滑化区間に対する予測確率、確率の標準偏差、粒子の分布の標準偏差を表す。観測値に対する予測は平滑化区間 0 において平均値は 58.56%で±6.17%の幅で変化する。そして平滑化区間が大きくなるに従って 56.4%、58.15%、57.65%と推移する。平滑化区間による変化は予測確率が若干上下するが、状態推定とは異なり、予測の分布の標準偏差の平均値にはほとんど変化が見られない。平滑化区間 0 の場合と 50 の場合では確率の標準偏差が小さくなっていて確率が推移する範囲が狭くなり安定的に動くようにも考えられるが、平滑化区間 15 においての確率の標準偏差が 9.08 と大きい事からもばらつきの範囲で一定であり平滑化区間に対する観測値予測に与える影響はあまりないといえる。

３．６　実験結果からの考察

状態推定、観測値予測の両結果から今回のモデルにおいての平滑化区間が与えるモンテカルロフィルタへの影響をまとめる。状態推定に関しては平滑化区間を大きくしすぎると粒子の分布に偏りが生じるため、推定確率は低下する。

一般に 30 程度以下にすべきであるといわれる平滑化区間は、その通りであまり大きくしない方がよく、今回のモデルでは平滑化区間が 10 以上の場合その精度は低くなった。観測値予測に関しては平滑化区間の違いによる影響は今回の結果からは見られなかった。しかし、状態推定と観測値予測を同時に行うことを考えると、今回のようなシンプルなモデルでは平滑化区間に関してはあまり大きくせずにモンテカルロフィルタを使うべきであるといえる。

３．７　モンテカルロフィルを使用しない場合との比較

最後に、モンテカルロフィルタによる状態推定がどの程度優れているか分かりやすい例を挙げてみる。モンテカルロフィルタを用いずに与えられたモデル式（システムモデル）のみで数期先まで予測しようとするとどうなるか以下のグラフで説明する。

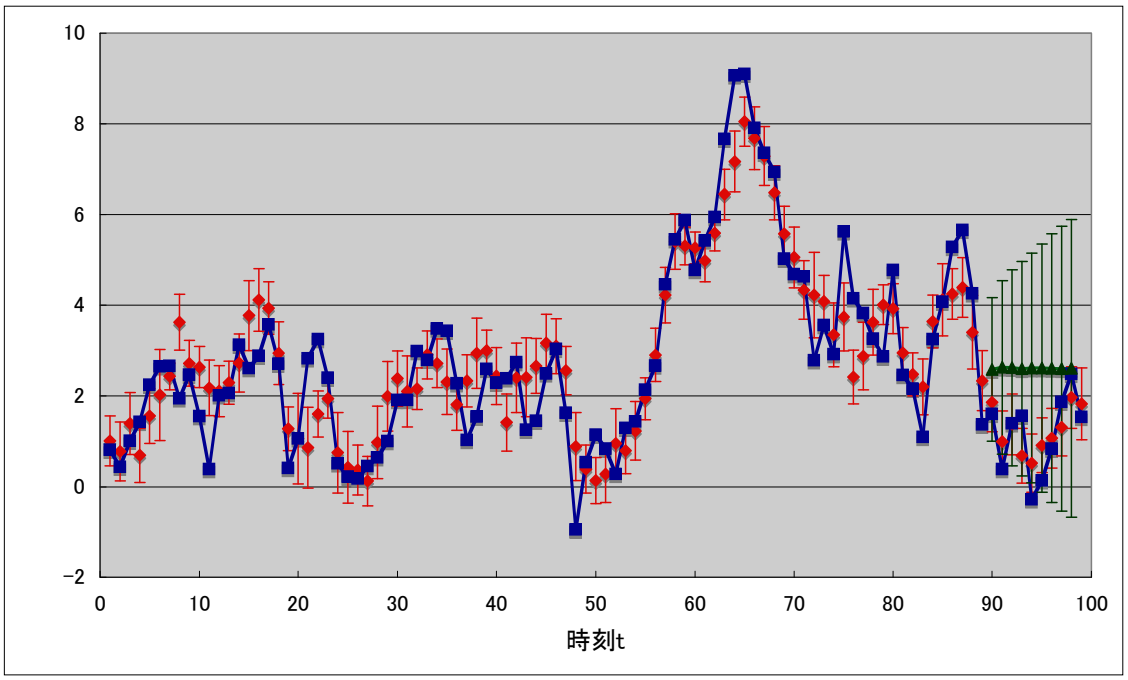


図 20　モンテカルロフィルタを使用しない場合の比較図。青い線は理論値、赤い線は粒子の分布、緑の点とエラーバーで表されているのがシステムモデルから予測した状態である。

上のグラフでは 90 ステップまではモンテカルロフィルタを使い状態推定を行い、90 から 99 まではモンテカルロフィルタを使用したものとシステムモデルの式を用いて単純に粒子を伝搬させた予測分布を表す。グラフのとおりこのモデル式では状態推定の範囲がノイズの正規分布によってだんだんと広がっていっているのが分かる。つまり予測の範囲が広くなっていくので先の状態がうまく推定できていない。この推定された状態から観測値を生成しようとすると、観測値も広い範囲でしか予測できず、モンテカルロフィルタが一定の精度で推定できていることの有効性がよくわかる。

３．８　時系列データマイニングへの応用

これまでの結果からモンテカルロフィルタによる状態推定、予測の結果が出た。平滑化区間によるモンテカルロフィルタに対する影響をふまえると平滑化

区間に関してはあまり大きくない方がよいと結論づける事ができる。

今回のモデル式においては大量の観測データを用いる状態推定においては平滑化はあまり効果が見られない結果となった。

しかし、今回の実験のグラフを見ると、状態推定、予測ともに一定の精度を保ったまま、長い時系列に関して追跡を続けていっているのが分かる。グラフでの予測の動きを見る限り動きを大きく見失うようなことなく追跡できている。この性質を生かした時系列データマイニングへの応用、例えば観測値や状態の粒子による自動追跡などが期待できる。

課題としては今回のモデルはシステムモデル、観測モデルともに正規分布に従うノイズを持つモデル式だったが、時刻によって複数のモデル式を切り替わるといったモデルに対してモンテカルロフィルタを用いる事により、その時刻のデータがどのモデル式から生成されたものか判断するといったモンテカルロフィルタの有効性の検証や現実の時系列データに対するデータマイニングが今後の課題となる。

4　終わりに

今回、モンテカルロフィルタと時系列データマイニングについて実験を行っ

た。モデル式から観測データを生成しモンテカルロフィルタによって状態推定、観測値予測を行った。

平滑化区間 L の変化による状態推定、観測値予測への影響は今回のモデルにおいて状態推定については平滑化区間を大きくすることで粒子の分布に偏りが生じその精度が落ちる事が分かった。そして、観測値予測については平滑化区間を大きくしても影響はあまり見られない事が分かった。平滑化区間を大きくする、つまり長期間蓄積したデータを活用しデータマイニングを行う事についてはあまりメリットがない場合もありえるといえる。

しかし、一方で 100 ステップ経過後も精度を落とす事なく粒子によって、状態、観測値の追跡が可能だという事が確認された。つまり、長期間の状態、観測値の自動予測が可能であるといえ、この性質を生かした時系列データマイニングへの応用が期待できる。

時系列データマイニングの方法にモンテカルロフィルタを用いた今回の研究によって、モンテカルロフィルタの時系列データへの有効性と平滑化区間に関して平滑化区間を長くし、大量の蓄積したデータを活用するという面ではあまり効果的でない事が分かった。

今回のようなシンプルなモデルではなくシステムモデル、観測モデルが時刻とともに複数の式を切り替えながら変化していくようなモデルの場合においては、尤度を用いることでそのモデルを特定するといったモンテカルロフィルタを用いた別の有効性も発見できる。

謝辞

本研究を行うにあたり、高知大学理学部数理情報科学科本田理恵助手には、多くのご指導、助言をいただきました事を深く感謝申し上げます。

参考文献

[1] Kitagawa,G.,"Monte Carlo filter and smoother for non-Gaussian nonlinear state space models,"Journal of Computational and Graphical Statistics,Vol5,No.1,1996,1-25

[2]　樋口知之　粒子フィルタ　電子情報学会誌　2005　Vol88,NO12.P.989

[3] 北川源四郎　時系列解析入門　岩波出版　2005

付録　目次

１．２０試行の実験結果

表１　平滑化区間０の実験結果

試行数	x 予測成功確率(%)	粒子の分布の標準偏差	y 予測成功確率(%)	粒子の分布の標準偏差

1	61.2	0.79	61.2	1.61
2	82.6	0.78	67.3	1.60
3	71.4	0.78	59.2	1.60
4	75.5	0.79	65.3	1.60
5	67.3	0.79	63.3	1.60
6	53.0	0.78	61.2	1.60
7	71.4	0.79	51.0	1.60
8	67.3	0.79	61.2	1.61
9	57.1	0.79	59.2	1.60
10	67.3	0.76	63.3	1.60
11	57.1	0.79	55.1	1.60
12	77.6	0.78	63.3	1.60
13	65.3	0.79	61.2	1.60
14	59.2	0.79	49.0	1.61
15	75.5	0.79	67.3	1.61
16	75.5	0.78	59.2	1.59
17	61.2	0.79	46.9	1.60
18	53.0	0.78	53.0	1.60
19	75.5	0.80	46.9	1.60
20	61.2	0.79	57.1	1.59

表 2　平滑化区間 15 の実験結果

試行数	x 予測成功確	粒子の分布の	y 予測成功確	粒子の分布の

	率(%)	標準偏差	率(%)	標準偏差
1	59.2	0.65	61.2	1.60
2	76.5	0.64	65.3	1.62
3	69.4	0.64	61.2	1.60
4	67.3	0.66	59.2	1.61
5	59.2	0.67	63.3	1.60
6	57.1	0.63	61.2	1.61
7	69.4	0.63	49.0	1.60
8	75.5	0.63	55.1	1.59
9	57.1	0.64	59.2	1.60
10	57.1	0.66	63.3	1.62
11	61.2	0.64	51.0	1.61
12	65.3	0.64	61.2	1.60
13	57.1	0.65	59.2	1.60
14	55.1	0.66	51.0	1.60
15	75.5	0.65	67.3	1.61
16	40.0	0.65	59.2	1.60
17	51.0	0.57	24.0	1.60
18	59.2	0.65	53.0	1.59
19	69.4	0.67	49.0	1.61
20	65.3	0.64	55.1	1.60

表 3　平滑化区間 30 の実験結果

試行数	x 予測成功確率(%)	粒子の分布の標準偏差	y 予測成功確率(%)	粒子の分布の標準偏差
1	57.1	0.62	61.2	1.60
2	69.4	0.59	67.3	1.60
3	61.2	0.62	59.2	1.61
4	69.4	0.64	63.3	1.60
5	61.2	0.64	65.3	1.60
6	55.1	0.60	65.3	1.60
7	73.5	0.67	51.0	1.61
8	73.5	0.57	57.1	1.60
9	51.0	0.62	55.1	1.60
10	59.2	0.63	61.2	1.60
11	51.0	0.63	55.1	1.61
12	67.3	0.69	61.2	1.59
13	51.0	0.62	59.2	1.61
14	44.9	0.63	51.0	1.61
15	65.3	0.60	65.3	1.60
16	69.4	0.63	59.2	1.61
17	38.8	0.41	49.0	1.60
18	51.0	0.57	53.0	1.59
19	69.4	0.65	46.9	1.60
20	57.1	0.63	57.1	1.60

表 4　平滑化区間 50 の実験結果

試行数	x 予測成功確率(%)	粒子の分布の標準偏差	y 予測成功確率(%)	粒子の分布の標準偏差
1	57.1	0.55	59.2	1.6
2	75.5	0.57	63.3	1.6
3	59.2	0.61	59.2	1.6
4	61.2	0.64	59.2	1.59
5	46.9	0.61	63.3	1.6
6	53.0	0.57	61.2	1.61
7	65.3	0.59	49	1.6
8	65.3	0.58	61.2	1.6
9	53.0	0.59	57.1	1.6
10	46.9	0.58	63.3	1.6
11	49.0	0.58	53	1.6
12	63.3	0.6	61.2	1.6
13	51.0	0.6	61.2	1.61
14	46.9	0.59	49	1.59
15	65.3	0.64	65.3	1.6
16	65.3	0.61	59.2	1.59
17	49.0	0.57	51	1.6
18	55.1	0.56	53	1.6
19	63.3	0.62	44.9	1.6
20	63.3	0.62	59.2	1.6

2　モンテカルロフィルタのプログラム

```
#include  <stdio.h>
#include  <stdlib.h>
#include  <math.h>
#define PI 3.1415926
#undef RAND_MAX
#define RAND_MAX 2147483647 /* 2^(31)-1 */
#define NN 1100/*粒子の最大数*/
#define MM 100/*時刻ステップの最大数*/
#define LL 50  /*平滑化の区間*/
double nextNormalRandom(double mu, double sigma2);
double gauss(double x, double mu, double sigma);
void showData(double d[], int num );
void showParticle(double f[], double v[], double w[], double p[], double
y[], double alpha[], int num) ;

main()
{
    FILE *fin;
    FILE *fou;
    int l,n,n1,nmax,nend;/*時刻の引数*/
    int j,j1,jmax;/*粒子数*/
    double v[NN],mu,sigma2,w[NN],mu2,sigma3;/*vの平均、分散*/
    double v1,w1,x[MM],y1[MM];/*観測値のノイズ、観測値の状態と観測値の配列。*/
    double p[MM][NN],y[MM][NN];/*状態pj、観測値yj*/
    double alfa[NN],a1,a2;
    double A,A1,a[NN],ran;//alfaの合計,尤度のメモリの配列
    double f[MM][NN],fh[MM][NN];//リサンプリングした配列
    int k,K[MM][NN];/*平滑化の時に使う配列と引数*/
    int ans1,ans2;
    char fname[256]; /*ファイル名前の配列*/
    double fwa,fa,bun,hensa,sa,Y,pbun,yhensa;
```

```
    srandom(time(NULL)%RAND_MAX);/*乱数発生系列の初期化*/
    nmax=100;/*ステップ数*/
    mu=0;
    mu2=0;
    sigma2=1;
    sigma3=1;
    jmax=1000;      /*粒子数*/
    a1=1/(sqrt(2*M_PI))/sigma3;
    a2=1;//Gの偏微分の値
    n=0;
    x[0]=0;

//  printf("新しいデータを作成しますか？作るなら0:  既存のファイルデータを使うなら
1:¥n");
    scanf("%d",&ans1);

    if(ans1==1){
//      printf("既存ファイル名の入力");
        scanf("%s",fname);
        if ((fin=fopen(fname,"r"))==NULL) {
            printf("ファイルがありません。¥n");
        }
        else{
            n1=0;
            while(fscanf(fin,"%lf %lf",&x[n1],&y1[n1])!=EOF){
                ++n1;
            }
          fclose(fin);
        }
    }
    else if (ans1==0){
        /*データ作成*/
        for(n=1;n<nmax;++n) {
```

```
            v1=nextNormalRandom(mu,sigma2);/*vの生成*/
            w1=nextNormalRandom(mu2,sigma3);/*wの生成*/
            x[n]=x[n-1]+v1;     /*状態*/
            y1[n]=x[n]+w1;      /*観測値の計算*/
            printf("%d %lf %lf¥n",n,x[n],y1[n]);
    }

    printf("作成データを保存しますか。1:yes 0:no¥n");
    scanf("%d",&ans2);
    if(ans2==1){
        printf("保存ファイル名入力");
        scanf("%s",fname);
        fou=fopen(fname,"w");
        for(n=0;n<nmax;++n){
            fprintf(fou,"%lf %lf¥n",x[n],y1[n]);
        }
        fclose(fou);
    }
    else if(ans2!=0){
        exit(0);
    }
}
else{exit(1);}
/*+-30で一様分布にとる*/
for (j=0;j<jmax;++j) {

    f[0][j]=(60.0/jmax)*j-30.0;
}

for (n=1;n<nmax;++n) {
   A=0;
    for (j=0;j<jmax;++j) {
        v[j]=nextNormalRandom(mu,sigma2);
```

```
    w[j]=nextNormalRandom(mu2,sigma3);
    p[n][j]=f[n-1][j]+v[j];
    y[n][j]=p[n][j]+w[j];
    alfa[j]=gauss((y1[n]-p[n][j]),mu2,sigma3)*a2;
    A+=alfa[j];
}

A1=1/A;
a[0]=0;
for(j=0;j<jmax;++j){
    a[j+1]=a[j]+alfa[j]*A1;
}

for(n1=0;n1<n;++n1){
    for(j=0;j<jmax;++j){
        fh[n1][j]=f[n1][j];
    }
}
/*リサンプリング*/
for(j=0;j<jmax;++j){
    ran=(double)random()/RAND_MAX;
    for(l=0;l<jmax;++l){
        if(ran<a[l+1] && a[l]<ran){
            k=l;
        }
    }
    f[n][j]=p[n][k];
    /*固定区間平滑化*/
    nend=n-1-LL;
    if(nend<0)nend=0;
    for(n1=n-1;n1>nend;--n1){
        f[n1][j]=fh[n1][k];
    }
```

```
        }
    }

    /*平均と標準偏差*/
    for(n=1;n<nmax;++n){
        fwa=0;bun=0;
        for(j=0;j<jmax;++j){
            fwa+=f[n][j];
        }
        fa=fwa/jmax;
        for(j=0;j<jmax;++j){
            bun+=pow(f[n][j]-fa,2);
        }
        hensa=sqrt(bun/jmax);
        printf("%d %lf %lf %lf¥n",n,fa,hensa,fa-x[n]);
    }

    for(n=1;n<nmax;++n){
        Y=0;pbun=0;sa=0;
        for(j=0;j<jmax;++j){
            Y+=p[n][j];
        }
        for(j=0;j<jmax;++j){
            pbun+=pow(p[n][j]-(Y/jmax),2);
        }
        sa=(Y/jmax)-y1[n];  /*予測観測値の平均と理論値の差*/
        printf("%d  %lf %lf %lf¥n",n,Y/jmax,sqrt(pbun/jmax),sa);
    }

}
```

```
/*表示関数*/
void showData(double d[], int num)
{
    int i;
    for (i=0; i<num;++i ) {
        printf("%lf ", d[i]);
    }
    printf("¥n");
}

/*結果確認*/
void showParticle(double f[], double v[], double w[], double p[], double
y[], double alpha[], int num)
{
    printf("f¥n");
    showData(f,num);
    printf("v¥n");
    showData(v,num);
    printf("w¥n");
    showData(w,num);

    printf("p¥n");
    showData(p,num);
    printf("y¥n");
    showData(y,num);
    printf("alpha¥n");
    showData(alpha,num);
}

/*正規分布を計算*/
double gauss(double x, double mu, double sigma) {
    return exp(-pow((x-mu),2)/(2*sigma*sigma))/(sqrt(2*M_PI)*sigma);
}
```

```
double nextNormalRandom(double mu, double sigma2)/*正規分布を生成*/
{
    double t,u,r1,r2;
    double sigma=sqrt(sigma2);
    t=sqrt(-2.0*log(1-(double)random()/RAND_MAX));
    u=2*PI*(double)random()/RAND_MAX;
    r1=t*cos(u);
    return r1*sigma+mu;
}
```