

# The Application of Particle Filters on Non-linear Dynamic Problem Spaces

**Huan Yan**  
Florida State University  
yan@cs.fsu.edu

## Abstract

Estimation techniques for Non-Linear Dynamic spaces do not typically yield optimal solutions. Traditional methods such as linear regression are ineffective for conducting analysis because of the rigidity of the model produced and massive datasets that are required. The scale of the problem space will also require us to code recursively.

## 1 Introduction

Particle filters are useful derivations of Monte Carlo Methods with many applications including computer vision, econometrics, robotics, and forecasting. Many applications which involve time series modeling become more accurate when particle filtering is applied. MCM has recently enjoyed a rise in popularity thanks to cheaper and more efficient computational power.

### 1.1 Terminology

MCM- Monte Carlo Method  
SMC- Sequential Monte Carlo  
MCMC- Markov Chain Monte Carlo  
CLT- Central Limit Theorem  
ECN- Electronic Communications Network  
ATS- Automated Trading System  
IS- Importance Sampling

## 2. Background and Prior Work

This paper is based on prior work in a Software Engineering project at FSU. It utilizes several components, the first component is a Mathematica file that uses Wolfram servers to perform data harvesting. The second component is a C++ program that utilizes the data we have provided. The third component is a trading software platform provided by a third party. For demonstration purposes a proof of concept version is included. These components work in tandem to provide the information and functionality required to execute an automated trading system.

As a result of lower barriers of entry in terms of computing power and accessibility of technology, automated trading has overtaken the financial world. Traditional stock

exchanges such as the NYSE account for only 10-20% of all trades executed. The rest are done entirely by automated computer systems. Automated trading is so widespread that for every traditional stock exchange such as the NYSE, there are ten “ECN” centers such as ARCA and Bloomberg Tradebook.

Successful implementation of ATS delegates much of the decision making tasks which would typically require constant attention from a person.

Particle filters are particularly useful for this problem domain because of the need for econometrics that are self adapting based on a theoretically infinite data-set.

This is contrast to methods such as a linear regression, which are great at modeling performance on historical data, but perform poorly when applied to forecasting domains.

Particle filters address this issue by applying probabilistic measurements to the domain space.

### 2.1 Technique Overview

Particle methods assume  $x_k$  and the observations  $y_k$  can be modeled in this form:

- $x_0, x_1, \dots$  is a first order Markov process such that

$$x_k | x_{k-1} \sim p_{x_k | x_{k-1}}(x | x_{k-1})$$

With an initial distribution  $p(x_0)$ .

MCM techniques can be generalized to any domain where the problem is inherently stochastic.

$$p(x_k | y_0, \dots, y_k)$$

## 2.2 The Importance of Importance Sampling

Importance sampling (and resampling) is the meat of the Monte Carlo Method and where the bulk of computation takes place. Importance Sampling (IS) is represented by

$$\int f(x_k) p(x_k | y_0, \dots, y_k) dx_k \approx \frac{1}{P} \sum_{L=1}^P f(x_k^{(L)})$$

Where the set of particles we use is represented by

$$\pi(x_k | x_{0:k-1}, y_{0:k})$$

The target distribution is represented by

$$\pi(x_k | x_{0:k-1}, y_{0:k}) = p(x_k | x_{k-1}).$$

Importance sampling utilizes prior probability distributions to acquire the next set of normalized importance weights.

Importance Sampling and Resampling is modeled as:

1. For  $L = 1, \dots, P$  draw samples from the initial set of particles

$$x_k^{(L)} \sim \pi(x_k | x_{0:k-1}^{(L)}, y_{0:k})$$

2. For  $L = 1, \dots, P$  refresh the importance weights up to a normalizing constant:

$$\hat{w}_k^{(L)} = w_{k-1}^{(L)} \frac{p(y_k | x_k^{(L)}) p(x_k^{(L)} | x_{k-1}^{(L)})}{\pi(x_k^{(L)} | x_{0:k-1}^{(L)}, y_{0:k})}.$$

This simplifies to the equation

$$\hat{w}_k^{(L)} = w_{k-1}^{(L)} p(y_k | x_k^{(L)}),$$

3. For  $L = 1, \dots, P$  update the normalized importance weights:

$$w_k^{(L)} = \frac{\hat{w}_k^{(L)}}{\sum_{J=1}^P \hat{w}_k^{(J)}}$$

4. Compute an estimate of the effective number of particles as

$$\hat{N}_{eff} = \frac{1}{\sum_{L=1}^P \left(w_k^{(L)}\right)^2}$$

- 5) If the effective number of particles is less than a given

threshold  $\hat{N}_{eff} < N_{thr}$ , then perform resampling:

- a) Draw  $P$  particles from the current particle set with probabilities proportional to their weights. Replace the current particle set with this.
- b) For  $L = 1, \dots, P$  set  $w_k^{(L)} = 1/P$ .

## 3. Research

This covers the implementation of a simple SMC particle filter

The program is split up into several header files.

If you want to change the name of output or input files, or adjust the number of particles used then modify main.cpp.

We take an input file and load it into a vector for observed data, For every new piece of data we take in we use c++ function Transform and perform resampling every iteration.

Each new piece of data we receive makes the value that we predicted data more accurate.

The input file represents the change over time of a security over an arbitrary time interval (N). The output file represents a prediction of our price at interval (N+1)

### 3.1 Benefits

SMC is simple to implement and scales relatively well as long as the data is not high-dimensional.

Recursion allows us to generalize the problem for theoretically infinite datasets

Higher accuracy can be achieved using more particles

### 3.2 Pitfalls

SMC is a very computationally expensive algorithm when scaled for practical use, a compromise must be struck between efficiency and accuracy in terms of how many particles we use to resample our filter. Additional particles are at a cost of  $O(N^2)$  as demonstrated by efficiency graphs below

### 3.3 Method of Evaluation

We compared SMC to various forms of linear regression and found that SMC significantly outperforms linear regression. This is largely due to the fact that linear regression is just an average of current data. On the other hand SMC is a

stochastic evaluation method, while SMC does not necessarily approximate magnitude of change it does accurately predict the direction. Significant datasets must be used to ensure the accuracy of SMC.

#### 4. Results

The graphs with the first 100 points of observed and predicted data are included with the file with Number of Particles = 1000.

Additionally I have also attached graphs to demonstrate that SMC Implementation is  $O(N^2)$

WARNING: DO NOT RUN WITH (PARTICLES > 12500) you will time out from linprog before you finish.

Data = 500	
Number of Particles	Time to Completion
500	.44521
1000	1.4379
2000	6.5066
5000	35.7586
10000	224.96

Data = 1000	
Number of Particles	Time to Completion
500	.088869
1000	2.8244
2000	12.3924
5000	71.6907
10000	450.635

#### 5. Conclusion

SMC significantly outperforms linear regression in modeling tasks especially when data sets ( $N > 5000$ ), however it is still very computationally intensive. Great care must be taken to ensure that the model is not oversaturated with particles or else significant slow down will occur.

Possible improvements for SMC include using a better filtering technique, for instance to reduce the impact of outliers especially over time which can significantly skew the results of SMC predictions. Most improvements will impact the way Importance Sampling is handled in SMC simulations.

We can also introduce markov chains at the cost of a little more overhead to our algorithm. Markov chains may help us smooth out the outliers in much larger datasets.

For visual representations of these conclusions please consult the graphs included.

I will continue to investigate improvements for SMC for upgrading my ATS system. The broad applicability of SMC and particle filters means they can be used in any field that requires numbers to be forecasted.

#### 6. Acknowledgements

This class for introducing me to important concepts in AI which I would not have known about otherwise

David Gaitros for inspiring me to Software Engineering.

The Internet for being a giant fountain of knowledge that we can all drink from.

Particle Filter Prototype based on Zhiyuan Weng's Particle++ class

### Works Cited

Blevins, Jason. "Introduction to Sequential Monte Carlo Methods." Introduction to Sequential Monte Carlo Methods. 29 Nov. 2008. 29 July 2013  
<<http://jblevins.org/notes/smc-intro>>.

Doucet, Arnaud, and Adam M. Johansen. "A Tutorial on Particle Filtering and Smoothing." International Joint Conferences on Artificial Intelligence (2009). Mar. 2012.  
<<http://www2.warwick.ac.uk/fac/sci/statistics/staff/academic-research/johansen/publications/DJ11.pdf>>.

Gu\*, D., and H. Hu\*. "Target Tracking By Using Particle Filter In Sensor Networks." International Journal of Robotics and Automation 24 (2009).

Novak, Peter, and Wojciech Jamroga. "Agents, Actions and Goals in Dynamic Environments." International Joint Conference on Artificial Intelligence (2011).

Thrun, Sebastian. "Particle Filters in Robotics." In Proceedings of Uncertainty in AI (2002).