RESIDUAL AND STRATIFIED BRANCHING PARTICLE FILTERS

BY MICHAEL A. KOURITZIN

University of Alberta

ABSTRACT. A class of discrete-time branching particle filters is introduced with individual resampling: If there are $\mathbb{N}_n$ particles alive at time n, $\mathbb{N}_0 = N$, $a_n \leq 1 \leq b_n$, $\widehat{\mathbb{L}}_{n+1}^i$ is the current unnormalized importance weight for particle i and $\mathbb{A}_{n+1} = \frac{1}{N}\sum_{i=1}^{\mathbb{N}_n} \widehat{\mathbb{L}}_{n+1}^i$, then weight is preserved when $\widehat{\mathbb{L}}_{n+1}^i \in (a_n\mathbb{A}_{n+1}, b_n\mathbb{A}_{n+1})$. Otherwise, $\left\lfloor \frac{\widehat{\mathbb{L}}_{n+1}^i}{\mathbb{A}_{n+1}} \right\rfloor + \rho_n^i$ offspring are produced and assigned weight $\mathbb{A}_{n+1}$, where ρ_n^i is a Bernoulli of parameter $\frac{\widehat{\mathbb{L}}_{n+1}^i}{\mathbb{A}_{n+1}} - \left\lfloor \frac{\widehat{\mathbb{L}}_{n+1}^i}{\mathbb{A}_{n+1}} \right\rfloor$. The algorithms are shown to be stable with respect to the number of particles and perform better than the bootstrap algorithm as well as other popular resampled particle filters on both tracking problems considered here. Moreover, the new branching filters run significantly faster than these other particle filters on tracking and Bayesian model selection problems.

1. INTRODUCTION

Nonlinear filtering deals with determining the distribution of the current state of a non-observable, random, dynamic signal X given the history of a distorted, corrupted partial observation process Y living on the same probability space $(\Omega, \mathcal{F}, P)$ as X. Bayesian model selection, sometimes done while filtering, deals with determining which of a class of signal models $\{X^{(i)}\}_{i\in I}$ best fits the observed values of Y by pairwise Bayes' factor comparison. For many practical problems each potential signal is a time-homogeneous discrete-time Markov process $\{X_n,\ n = 0, 1, 2, ...\}$, living on some complete, separable metric space (E, ρ), with initial distribution π_0 and transition probability kernel K. The observation process takes the form ($Y_0 = 0$ and) $Y_n = h(X_{n-1}) + V_n$ for $n \in \mathbb{N}$, where $\{V_n\}_{n=1}^{\infty}$ are independent random vectors with common *strictly positive, bounded* density g that are independent of X, and the sensor function h is a mapping from E to $\mathbb{R}^d$. Then, the objective of filtering is to compute the conditional probabilities $\pi_n(A) = P\left(X_n \in A \,\middle|\, \mathcal{F}_n^Y\right)$, $n = 1, 2, ...$, for all Borel sets A or, equivalently, the conditional expectations $\pi_n(f) = E^P\left(f(X_n) \,\middle|\, \mathcal{F}_n^Y\right)$

2010 *Mathematics Subject Classification.* Primary 60G35; Secondary 62M20, 60G09, 60J80.

Key words and phrases. Particle Filter, Branching Process, Bayesian Model Selection.

Correspondence to: Department of Mathematical and Statistical Sciences, University of Alberta, Edmonton (Alberta), Canada T6G 2G1. Email address: michaelk@ualberta.ca.

for bounded functions $f : E \to \mathbb{R}$, where $\mathcal{F}_n^Y \doteq \mathcal{B}\{Y_l, l = 1, ..., n\}$ is the information obtained (meaning the σ-algebra generated) from the back observations $\{Y_l, l = 1, ..., n\}$. The objective of Bayes factor model selection is to compare the ratio B_n^{12} of marginal likelihoods between potential signal models $X^{(1)}$ and $X^{(2)}$ with respect to some reference probability measure Q.

To do both filtering and model selection, a reference probability measure Q is introduced under which the signal, observation process $\{(X_n, Y_{n+1}),\ n = 0, 1, ...\}$ has the same distribution as the signal, noise process $\{(X_n, V_{n+1}),\ n = 0, 1, ...\}$ does under P. Hence, the observations are i.i.d. random vectors with strictly positive bounded density g and are independent of X under measure Q. All the observation information is absorbed into the likelihood process $\{L_n,\ n = 1, 2, ...\}$ transforming Q back to P, which in our case has the form

$$\frac{dP}{dQ}\Big|_{\mathcal{F}_\infty^X \vee \mathcal{F}_n^Y} = L_n = \prod_{j=1}^n \alpha_j(X_{j-1}), \text{ with } \alpha_j(x) = \frac{g\left(Y_j - h\left(x\right)\right)}{g\left(Y_j\right)}, \tag{1.1}$$

so $L_n = \alpha_n(X_{n-1})L_{n-1}$ and $L_0 = 1$. (Here and in the sequel, $\mathcal{F}_n^X = \mathcal{B}(X_j,\ j \le n)$ and $\mathcal{F}_\infty^X = \mathcal{B}(X_j,\ j \ge 0)$ are the σ-algebras generated by $\{X_j,\ 0 \le j \le n\}$ and $X_j,\ j \ge 0$ respectively.) The following (well-known) discrete Girsanov's theorem constructs the real probability P from the reference Q.

Theorem 1. *Suppose that $\Omega = (E \times \mathbb{R}^d)^\infty$, $\mathcal{F} = \mathcal{B}((E \times \mathbb{R}^d)^\infty)$, $\{X_n,\ n = 0, 1, ...\}$ and $\{Y_n,\ n = 1, 2, ...\}$ are independent processes on $(\Omega, \mathcal{F}, Q)$, the $\{Y_n\}$ are i.i.d. with strictly-positive, bounded density g on $\mathbb{R}^d$ and $V_n \doteq Y_n - h(X_{n-1})$ for all $n = 1, 2, ...$ Then, there exists a probability measure P such that (1.1) holds, $\{V_n,\ n = 1, 2, ...\}$ are i.i.d. on $(\Omega, \mathcal{F}, P)$ with density g and $\{X_n\}$ is independent of $\{V_n\}$ with the same law as on $(\Omega, \mathcal{F}, Q)$.*

The *unnormalized filters* are then

$$\sigma_n\left(f\right) = E^Q\left(L_n f\left(X_n\right) \middle| \mathcal{F}_n^Y\right), \tag{1.2}$$

so $\sigma_0 = \pi_0$, as $L_0 = 1$ and $\mathcal{F}_0^Y = \{\emptyset, \Omega\}$ and the filter satisfies $\pi_n\left(f\right) = \frac{\sigma_n(f)}{\sigma_n(1)}$ by Bayes rule. Moreover, the Bayes factor satisfies $B_n^{12} = \frac{\sigma_n^{(1)}(1)}{\sigma_n^{(2)}(1)}$, where $\sigma_n^{(i)}\left(f\right) = E^Q\left(L_n^{(i)} f\left(X_n^{(i)}\right) \middle| \mathcal{F}_n^Y\right)$, with $L_n^{(i)} = \prod_{j=1}^n \alpha_j(X_{j-1}^{(i)})$, is the unnormalized filter for signal model $X^{(i)}$. Therefore, we can combine Bayesian model selection *and* filtering (for each potential signal) by constructing approximations (denoted $\mathbb{S}_n^N$ below) to the unnormalized filter for each candidate signal model. As words of caution, our setting is certainly not the most general possible for our unnormalized and branching particle filter approach as we do not want to over complicate the setting and conditions while introducing new methods. Indeed, it is anticipated that with some work other observation models can be used and some form of Lookahead Sequential Monte Carlo strategy related to those considered in Lin et. al. (2013) could be developed based upon our branching particle algorithms. However, there are also interesting situations like rare event importance sampling (see Le Gland and Oudjane (2006)) that appear ill-suited for our approach, even with a more general setting.

1.1. **Background.**

Particle filters are utilized widely and the original (resampled) interacting particle filters have been intensely studied (see e.g. Del Moral and Miclo (2000) and Cappe et. al. (2007) for an overview and historical account). However, particle filters performance depends heavily upon at least two factors:

- The importance density proposals used for sampling, and
- The resampling method used,

with both being active areas of investigation and the later claim being justified in e.g. Del Moral et. al. (2000), Douc et. al. (2005) and Hol et. al. (2006). Moreover, resampling is the most difficult and critical step to parallelization as is pointed out in Murray et. al. (2016) so effective replacement by branching may be even more valuable in parallel implementations. Furthermore, resampled particle filters approximate the actual filter π_n so prior filter estimates must be stored to perform Bayes factor model selection. On the other hand, the weighted particle filter (credited to Handschin (1970), Handschin and Mayne (1969)) approximates the unnormalized particle filter σ_n, is the most basic particle filter and is embarassingly computer parallelizable. More generally, branching particle filters, like those introduced by Crisan and Lyons (1997), can have model selection capabilities, effective resampling and be highly parallelizable. However, branching particle filters suffer from dramatic particle swings and difficult analysis - or do they? Herein, we introduce and analyze branching particle filters that avoid the weighted-particle-filter particle spread problems yet still have immediate model selection capabilities. They include the weighted particle filter as the extreme zero-resampling case and a branching variation of the better algorithm in Del Moral et. al. (2000) as the fully-resampled case. They are stable with respect to particle number swings and can be analyzed using exchangeability (in lieu of independence). In order to focus just on our branching scheme, we ignore possible (large, problem-dependent) gains attainable by using alternative importance sampling density proposals and stick to sampling from the signal dynamics.

There are many approaches to reducing resampling noise in the basic bootstrap filter. For example, researchers brought in importance sampling and delayed bulk resampling methods (see e.g. Del Moral et. al. (2012)). Others have introduced less noisy types of resampling, which we discuss below. However, there are few studies like Ballantyne et. al. (2000) of the practical partially-resampled algorithms where decisions are made on a particle-by-particle basis with the aim of only removing the poor particles and splitting the best particles (in an unbiased manner). Kouritzin and Sun (2005) do obtain L_2-rates of convergence for a partially-resampled algorithm in a specific setting. Our present work introduces new classes of branching particle filters, motivates their use and sets up a framework for studying them. We refer the reader to standard texts Bain and Crisan (2008) and Del Moral (2004) for motivation and theoretical background of particle filters as well as filtering in general.

The first, and still popular, model selection procedures in particle filters used prediction error methods (see e.g. Djurić (1999)). Our methods are Bayes factor based, motivated by Kass and Raftery (1995), and used in Kouritzin and Zeng (2005a), Kouritzin and Zeng (2005b) and Kouritzin (2015). As is shown in these later papers, one merely needs the unnormalized filters or filter ratio processes to employ these methods. Therefore, our branching particle approximations of the unnormalized filter are most appropriate. Although not considered here, particle filters and, more generally, particle integration techniques as discussed in Del Moral et. al. (2013) are also important in applied probability, Bayesian statistics, numerical physics, probabilistic machine learning, and engineering sciences (see e.g. Andrieu et. al. (2014), Andrieu et. al. (2010), Del Moral et. al. (2013)).

1.2. **Motivation.** Our goal now is to motivate the use of our novel branching particle filters so we summarize the results that will be presented differently in Section 4. First we compare existing methods to one of the methods introduced herein.

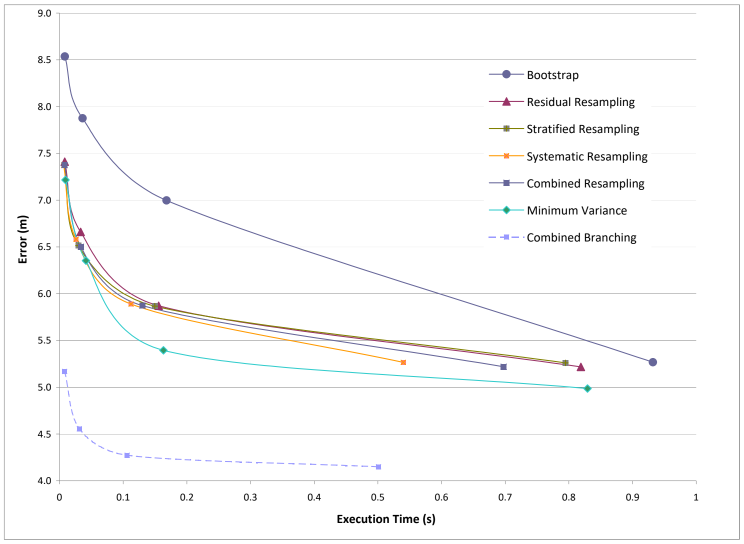

FIGURE 1. Average Error of Test Model

Figure 1 displays error versus execution time in meters per second on one of the problems considered herein. The bootstrap, residual, stratified, systematic, combined and minimum variance algorithms displayed here are all standard resampled particle filters used in practice and re-explained within. The Combined Branching method is one of the simpler algorithms introduced herein. There are still two algorithms introduced below that significantly outperform this new Combined Branching one but are modestly more complicated. One can see that original bootstrap algorithm behaves poorly compared to the other standard resampled algorithms.

However, it may be surprising how much better one can do yet using the algorithms introduced within, which merely change the amount and type of resampling or branching used. We will step through our novel residual, combined, dynamic and effective branching particle algorithms that give a nice tradeoff between simplicity and performance.

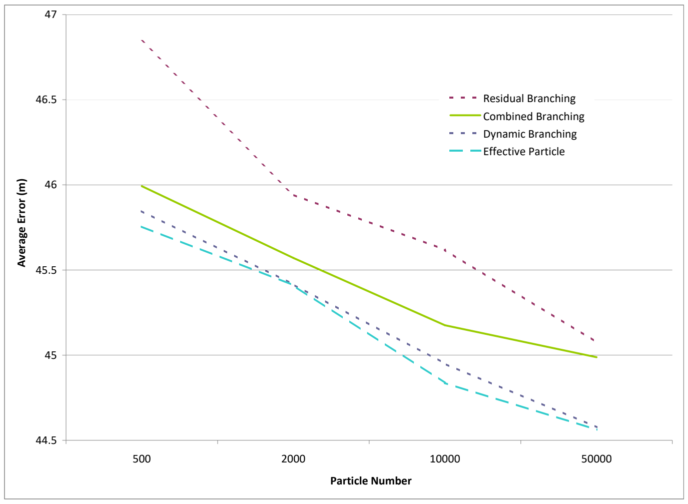

FIGURE 2. Average Error of Range Only Model

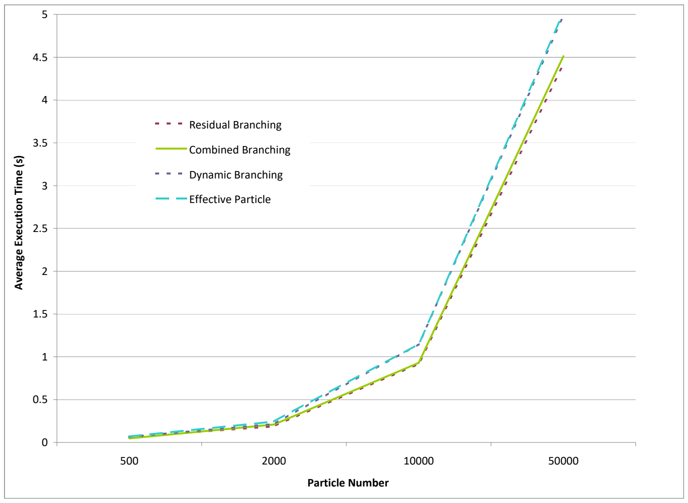

FIGURE 3. Average Execution Time of Range Only Model

Figures 2 and 3 display our new methods on a second model considered herein. The residual branching particle filter is the simplest, followed by the combined. It is not obvious which of the dynamic or effective particle methods one should choose as the execution time is slightly less for the dynamic. This is discussed in Section 4.

1.3. **Notation.** We use standard notation where possible:

$\lfloor z \rfloor$ is the largest integer less than $z \geq 0$ and $\{z\} = z - \lfloor z \rfloor$.

$y \vee z = \begin{cases} y & \text{if } y > z \\ z & \text{if } y \leq z \end{cases}$ is the larger value.

$\mathcal{G} \vee \mathcal{H}$ is the smallest σ-algebra containing both $\mathcal{G}, \mathcal{H}$ when they are σ-algebras.

$\forall$ stands for all (what follows).

$:=$ is used for pseudocode assignment.

$U \sim F$ is used for creation of random variable U with distribution F.

$\mathcal{B}(S)$ is the Borel σ-algebra on topological space S.

$\mathcal{B}(\{Z_a,\ a \in A\})$ is the σ-algebra generated by the random variables $\{Z_a,\ a \in A\}$.

P is the real world probability measure with E^P denoting expectation with P.

Q (with expectation E^Q) is the reference probability with pure noise observations.

σ is the unnormalized filter constructed with respect to reference measure Q.

$Q^Y(\cdot) = Q(\cdot | \mathcal{F}_\infty^Y)$ so $Q^Y(A) = Q(A | \mathcal{F}_n^Y)$ a.s. for $A \in \mathcal{F}_\infty^X \vee \mathcal{F}_n^Y$ since the observations are i.i.d. and independent of the signal and particles under Q.

1.4. **Computer Specifications.** All experiments are performed on the same computer system, consisting of a Lenovo Y410P Laptop with a 4th generation Intel Core i7-4700MQ processor, 8GB PC3-12800 DDR3L SDRAM 1600 MHz memory, 1TB 5400 RPM hard disk, Windows 8.1 64 bit operating system and the C++ compiler from Visual Studio 2015.

1.5. **Layout.** In the next section, we discuss resampled (Bootstrap-related) particle filters for tracking and model selection problems. In particular, multinomial, residual, stratified, systematic, combined residual-stratified and minimum variance schemes are considered. Our class of branching particle filters is introduced and studied in Section 3. The common complaints of unstable particle numbers as well as unpredictable results and speed of branching filters are largely overcome. We also give variants/improvements that use stratified and state-dependent branching. Indeed, it is shown in Section 4 that our most basic branching algorithm can be significantly faster and more accurate at tracking than all the bootstrap variant algorithms considered here. We then study improved branching algorithms that give yet better performance or require less computation time. The resampling and branching operations required for the various algorithms have been tabulated and are included in the online Supplemntary Materials (see Appendix A). Finally, the proof of our stability result, establishing boundedness of particle numbers and weights, has also been relegated to the online Supplemntary Materials (see Appendix B).

2. Resampled Particle Filters

In this section, we review the *resampled particle filters*, starting with the original *bootstrap algorithm* of Gordon et. al. (1993), but also including: Residual resampling of Liu and Chen (1998), Stratified resampling of Kitagawa (1996), Combined Residual-Stratified resampling discussed in Douc et. al. (2005), Minimum Variance resampling of Crisan (2001) and Systematic resampling of Carpenter et. al. (1999). [1] These algorithms overcome the increasing variance weight problem of the weighted filter pointed out in Doucet et. al. (2000), only require $O(N)$ in operations for N particles due to a clever idea of Carpenter et. al. (1999) and were shown to be unbiased in Douc et. al. (2005). We also explain how to use these algorithms in model selection problems.

2.1. **Basic Bootstrap Algorithm.**

The *bootstrap particle filter* was a breakthrough in sequential data estimation and its convergence properties have been studied in e.g. Del Moral and Miclo (2000). However, it has limitations in terms of model selection, parallelizability, performance and speed. For clarity, we summarize the algorithm:

```
**Initialize:** $\left\{\mathbb{X}_0^k\right\}_{k=1}^N$ are independent initial particle samples of $\pi_0$, $V_{N+1} = 1$
**Repeat:** for $n := 0, 1, 2, ...$ do
  (1) Weight by Observation: $\widehat{\mathbb{L}}_{n+1}^k := \alpha_{n+1}\left(\mathbb{X}_n^k\right)$ for $k := 1, 2, ..., N$
  (2) Normalize Weight: $w_{n+1}^k := \frac{\widehat{\mathbb{L}}_{n+1}^k}{\widehat{\mathbb{L}}_{n+1}}$ for $k := 1, 2, ..., N$, where $\widehat{\mathbb{L}}_{n+1} := \sum_{i=1}^N \widehat{\mathbb{L}}_{n+1}^i$
  (3) Evolve Independently:
      $$P(\widehat{\mathbb{X}}_{n+1}^k \in \Gamma_k\ \forall\ k | \mathcal{F}_n^{\mathbb{X}} \vee \mathcal{F}_n^Y) := \prod_{k=1}^N K(\mathbb{X}_n^k, \Gamma_k)\ \forall \Gamma_k$$
  (4) Estimate $\pi_{n+1}$ by: $\mathbb{P}_{n+1}^N := \sum_{k=1}^N w_{n+1}^k \delta_{\widehat{\mathbb{X}}_{n+1}^k}$.
  (5) Resample: $p_i := \sum_{k=1}^i w_{n+1}^k$ for $i := 1, ..., N$, $j := N-1$
      **Repeat:** for $k := N, N-1, ..., 2, 1$ do
        • Draw $[0,1]$-uniform $U_k$ and set $V_k := U_k^{\frac{1}{k}} V_{k+1}$
        • While $V_k \le p_j$ set $j := j - 1$
        • Set $\mathbb{X}_{n+1}^k := \widehat{\mathbb{X}}_{n+1}^{j+1}$
```

Remark 1. *There are a few things to note about our version of the bootstrap algorithm:*

1. *Our use of the Carpenter et. al. (1999) improvement dramatically improves the efficiency of this algorithm as well as the Residual to follow.*
2. *We extract our estimate before resampling to avoid excess noise.*

[1]Some real-time applications are not conducive to sudden switches to slow times so we limit ourselves to procedures where the resampling is essentially evenly spread out over time.

(3) *We have called our algorithm* Basic *Bootstrap here because many authors see (e.g. Murray et. al. (2016)) use the Bootstrap algorithm to refer to any algorithm that samples from the signal dynamics and weights by the likelihood. We will not follow this nomenclature because it would mean all our resampled methods would be labeled the bootstrap algorithm. Instead, we will follow the alternative terminology, drop the* Basic *from the Bootstrap Algorithm henceforth and label the other algorithms by their resampling method, which is what is important to us.*

(4) *The conditioning in Step (3) is with respect to the back observations as well as all particles after the previous resampling step.*

The multinomial resampling introduces excess noise that degrades performance.

2.2. **Residual Resampling.** Liu and Chen (1998) introduced residual resampling to reduce resampling noise in the bootstrap filter. The idea is to keep particles at the higher-weight sites (those with weight above the average) so fewer particles are redistributed and less resampling noise is introduced, reducing the number of uniform random variables used from N to some random R defined below.

The bootstrap algorithm is modified by replacing step (5) by the following:

```
(5) Preserve: $S := 0$

    Repeat: for $j := 1, 2, ..., N$ do
      – $k := 0$
      – While $k < \lfloor N w_{n+1}^j \rfloor$ set $k := k+1$, $\mathbb{X}_{n+1}^{S+k} := \widehat{\mathbb{X}}_{n+1}^j$
      – $S := S + k$
(6) Resample: $R := N - S$; $\overline{p}_i := \sum_{k=1}^{i} \frac{\{N w_{n+1}^k\}}{R}$ for $i := 1, ..., N$; $j := N - 1$

    Repeat: for $k := N, N-1, ..., 2, S+1$ do
      – Draw $[0,1]$-uniform $U_k$ and set $V_k := U_k^{\frac{1}{k}} V_{k+1}$
      – While $V_k \le \overline{p}_j$ set $j := j - 1$
      – Set $\mathbb{X}_{n+1}^k := \widehat{\mathbb{X}}_{n+1}^{j+1}$
```

The reduced resampling noise results in improved performance and speed.

2.3. **Stratified Resampling.** Kitagawa (1996) reduced the randomness in the (bootstrap) uniform random variables to an interval of length $\frac{1}{N}$ and avoided the need for ordering the uniforms. The bootstrap algorithm, given above, is easily modified for this improvement by replacing Step (5) with:

```
(5) Resample: $p_i := \sum_{k=1}^{i} w_{n+1}^k$ for $i := 1, ..., N$, $j := 1$

    Repeat: for $k := 1, 2, ..., N$ do
      – Draw $\left[\frac{k-1}{N}, \frac{k}{N}\right]$-uniform $U_k$
      – While $U_k \ge p_j$ set $j := j + 1$
      – Set $\mathbb{X}_{n+1}^k := \widehat{\mathbb{X}}_{n+1}^j$
```

Each uniform above has variance $\frac{1}{12N^2}$ versus $\frac{1}{12}$ for the bootstrap. This smaller uniform variance translates into smaller particle system variance in the resampling.

2.4. **Systematic Resampling.** Carpenter et. al. (1999) modified the stratified resampling to be more computationally efficient at the cost of giving up conditional independence. Only one uniform random variable is used. Step (5) simply becomes:

```
(5) Resample: $p_i := \sum_{k=1}^{i} w_{n+1}^k$ for $i := 1, ..., N$, $j = 1$

    Draw $\left[-\frac{1}{N}, 0\right]$-uniform $U$

    Repeat: for $k := 1, 2, ..., N$ do
      – Set $U_k := U + \frac{k}{N}$
      – While $U_k \geq p_j$ set $j := j + 1$
      – Set $\mathbb{X}_{n+1}^k := \widehat{\mathbb{X}}_{n+1}^j$
```

Systematic resampling can in theory behave poorly (see Douc et. al. (2005)). It can also behave very well (see Hol et. al. (2006)).

2.5. **Combined Resampling.** Since the Residual and Stratified resampling methods reduce the randomness in different ways, the combination might produce a yet better method (see Douc et. al. (2005)). Step (5) in the bootstrap is replaced with:

```
(5) Preserve: $S := 0$

    Repeat: for $j := 1, 2, ..., N$ do
      – $k := 0$
      – While $k < \lfloor N w_{n+1}^j \rfloor$ set $k := k + 1$, $\mathbb{X}_{n+1}^{S+k} := \widehat{\mathbb{X}}_{n+1}^j$
      – $S := S + k$
(6) Resample: $R := N - S$; $\overline{p}_i := \sum_{k=1}^{i} \frac{\{N w_{n+1}^k\}}{R}$ for $i := 1, ..., N$; $j := 1$

    Repeat: for $k := 1, 2, ..., R$ do

      – Draw $\left[\frac{k-1}{R}, \frac{k}{R}\right]$-uniform $U_k$
      – While $U_k \geq \overline{p}_j$ set $j := j + 1$
      – Set $\mathbb{X}_{n+1}^{S+k} := \widehat{\mathbb{X}}_{n+1}^j$
```

Combined Resampling clearly improves Residual Resampling but it is unclear if it improves Stratified. Stratified uses N random variables with variance $\frac{1}{12N^2}$ while Combined resampling uses R random variables with variance $\frac{1}{12\,R^2}$. Which produces less resampling noise is not obvious without further analysis.

2.6. **Minimum Variance Scheme.** Crisan (2001) introduced the Minimum Variance branching particle system. Since this scheme actually uses a fixed number of particles, we will consider it a resampling scheme. It is similar to the Combined Resampling method in the sense it uses residuals as well as negative correlations but has a very different implementation. Modifying the description of this algorithm in Bain and Crisan (2008) slightly to be more efficient, we replace Step (5) in the bootstrap with:

```
(5) Particle Numbers: $\mathbb{N}_{n+1}^{N} := N$, $W := N$

    Repeat: for $j := 1, 2, ..., N-1$ do

      – $\mathbb{N}_{n+1}^{j} := \lfloor N w_{n+1}^{j} \rfloor$
      – Draw $[0,1]$-uniform $U_j$
      – If $\{N w_{n+1}^{j}\} + \{W - N w_{n+1}^{j}\} < 1$ and $U_j \leq \frac{\{N w_{n+1}^{j}\}}{\{W\}}$ then
          $\mathbb{N}_{n+1}^{j} := \mathbb{N}_{n+1}^{j} + \mathbb{N}_{n+1}^{N} - \lfloor W \rfloor$
      – else if $\{N w_{n+1}^{j}\} + \{W - N w_{n+1}^{j}\} \geq 1$ and $U_j \geq \frac{\{N w_{n+1}^{j}\} - \{W\}}{1 - \{W\}}$ then
          $\mathbb{N}_{n+1}^{j} := \mathbb{N}_{n+1}^{j} + \mathbb{N}_{n+1}^{N} - \lfloor W \rfloor$
      – else if $\{N w_{n+1}^{j}\} + \{W - N w_{n+1}^{j}\} \geq 1$ and $U_j < \frac{\{N w_{n+1}^{j}\} - \{W\}}{1 - \{W\}}$ then
          $\mathbb{N}_{n+1}^{j} := \mathbb{N}_{n+1}^{j} + 1$
      – for $k := 0, ..., \mathbb{N}_{n+1}^{j} - 1$ do $\mathbb{X}_n^{\mathbb{N}_{n+1}^{N} - k} := \widehat{\mathbb{X}}_{n+1}^{j}$
      – $\mathbb{N}_{n+1}^{N} := \mathbb{N}_{n+1}^{N} - \mathbb{N}_{n+1}^{j}$, $W := W - N w_{n+1}^{j}$
```

The real difference from the Combined Resampling method is instead of using Stratified Uniforms to create the negative correlations, we code the negative correlations into the pseudocode. The main difference between the algorithm above and that in Bain and Crisan (2008) is that we note $1 - U_j$ is $[0,1]$-uniform if U_j is and use this to make the algorithm slightly more efficient. Notice, W and $\mathbb{N}_{n+1}^{N}$ keep track of the remaining weight and particles to place. We reorder the particles, i.e. the first becomes the last and vice versa, in order to avoid extra operations related to tracking the number of particles placed as well as those left to place.

As the name suggests, the advantage of this method is that the weight-equalizing resampling is done in the minimum variance manner. This implies that the j^{th} particle with (normalized) weight w^j will be resampled into either $\lfloor N w^j \rfloor$ particles with probability $1 - \{N w^j\}$ or $\lceil N w^j \rceil$ with probability $\{N w^j\}$, whereas the previous methods had a (small) chance of producing more than $\lceil N w^j \rceil$ particles. In this way, the Minimum Variance method looks good. However, the larger number of uniform random variables used compared to the Combined Method (N versus R) may require more execution time. It will be interesting to see which algorithm, the Combined Resampling or the Minimum Variance, is more effective on our problems.

2.7. **Chopthin Algorithm.** While this article was under review, Gandy and Lau (2016) introduced a novel noise reduction algorithm, called the Chopthin algorithm, that works with e.g. systematic and stratified resampling. Basically, this algorithm follows these resampling methods until final production of the new particles. Then, instead of doing complete resampling and resetting all (normalized) weights to one, it does partial resampling by thinning the low weight particles and splitting some of the high weight particles in such a way that the expected number of particles remains the same. In this way both particles and weights must be propogated and the number of particles is random. However, the amount of resampling noise can be dramatically reduced. This algorithm shares some of the advantages of our algorithms introduced herein. However, the Combined Resampling algorithm also reduces the resampling noise in the Stratified algorithm. Indeed, it is not clear that the Stratified/Chopthin will be better than Combined Resampling on most problems and the Combined Resampling method has the advantages of not needing to propogate weights and of having a constant number of particles. We do not compare our algorithms to the Chopthin algorithm due to its essentially simultaneous introduction. However, it should be noted that our algorithms win mostly on speed and on the ease of model selection. Adding the Chopthin algorithm will not help on either of these and our algorithms compare most favorably to the Combined Resampling method.

2.8. **Model Selection.** In most real tracking problems, one never knows the best model of reality and should let the data decide, leading to the combined problem of model selection and tracking. The unnormalized filter total mass $\sigma_n(1)$ gives Bayes factor of the observations containing a given signal model (i.e. $Y_j = h(X_{j-1}) + V_j$ for $j \le n$) to pure noise (i.e. $Y_j = V_j$ for $j \le n$). The ratio of unnormalized filters $B_n^{12} = \frac{\sigma_n^{(1)}(1)}{\sigma_n^{(2)}(1)}$ for two different models ($Y_j = h(X_{j-1}^1) + V_j$ for $j \le n$ and $Y_j = h(X_{j-1}^2) + V_j$ for $j \le n$) gives the Bayes factor for signal 1 versus signal 2.

Del Moral and Miclo (2000, p. 16) recover the unnormalized filter allowing model selection with bootstrap-type algorithms. By Bayes rule and (1.2), one finds that

$$\pi_{n-1}(\alpha_n) = \frac{\sigma_{n-1}(\alpha_n)}{\sigma_{n-1}(1)} = \frac{\sigma_n(1)}{\sigma_{n-1}(1)} \Rightarrow \sigma_n(1) = \prod_{m=1}^{n} \pi_{m-1}(\alpha_m), \tag{2.1}$$

where α_m is defined in (1.1). Hence, to do model selection in the bootstrap-type algorithms, one can just add the following after step (1):

(1a) Model Selection: $\sigma_{n+1}(1) = \sigma_n(1)\frac{\widehat{\mathbb{L}}_n}{N}$, where $\widehat{\mathbb{L}}_n = \sum_{i=1}^{N} \widehat{\mathbb{L}}_n^i$.

We then need not calculate $\widehat{\mathbb{L}}_n$ in step (2) but must add $\sigma_0(1) = 1$ to the initialize.

3. Branching Particle Filters

In this section, we introduce a class of branching particle filters. We no longer necessarily have full resampling but rather allow partial resampling and weight propagation. In one extreme case, we have the weighted particle filter where no resampling takes place and weights are always propagated. In the other extreme case, we have a fully resampled particle filter, which can be thought of as a branching alternative to the residual resampling or combined resampling particle filter. In between, we have a whole class of branching particle filters with a flexible amount of resampling that affords an effective tradeoff between weight variance increase (the weighted particle issue) and resampling noise (the bootstrap particle issue). We first describe the branching particle filters in terms of uniform random variables $\{\mathbb{U}_n^k\}$ used to create the branching variables $\{\rho_n^k\}$ in two different ways.

The following branching Markov process $\{\mathbb{S}_n^N, n = 0, 1, ...\}$ approximates the *unnormalized* filter $\{\sigma_n, n = 0, 1, ...\}$ in terms of the observations as follows:

Initialize: $\left\{\mathbb{X}_0^k\right\}_{k=1}^N$ are independent samples of π_0, $\mathbb{N}_0 := N$, $\mathbb{N}_n := 0$ for all $n \in \mathbb{N}$ and $\mathbb{L}_0^k := 1$ for $k := 1, ..., N$.

Repeat: for $n := 0, 1, 2, ...$ do

(1) Weight by Observation: $\widehat{\mathbb{L}}_{n+1}^k := \alpha_{n+1}\left(\mathbb{X}_n^k\right)\mathbb{L}_n^k$ for $k := 1, 2, ..., \mathbb{N}_n$

(2) Evolve Independently:

$$Q^Y(\widehat{\mathbb{X}}_{n+1}^k \in \Gamma_k\ \forall\ k | \mathcal{F}_n^{\mathbb{X}} \vee \mathcal{F}_{n+1}^{\mathbb{U}}) := \prod_{k=1}^{\mathbb{N}_n} K(\mathbb{X}_n^k, \Gamma_k) \quad \forall \Gamma_k$$

(3) Estimate σ_{n+1} by: $\mathbb{S}_{n+1}^N := \dfrac{1}{N}\displaystyle\sum_{k=1}^{\mathbb{N}_n} \widehat{\mathbb{L}}_{n+1}^k \delta_{\widehat{\mathbb{X}}_{n+1}^k}$ and $\pi_{n+1}(f)$ by $\frac{\mathbb{S}_{n+1}^N(f)}{\mathbb{S}_{n+1}^N(1)}$.

(4) Average Weight: $\mathbb{A}_{n+1} := \mathbb{S}_{n+1}^N(1)$

Repeat (5-6): for $k := 1, 2, ..., \mathbb{N}_n$ do

(5) Resampled Case: If $\widehat{\mathbb{L}}_{n+1}^k \notin (a_n\mathbb{A}_{n+1}, b_n\mathbb{A}_{n+1})$ then
   (a) Offspring Number: $\mathbb{N}_{n+1}^k := \left\lfloor \frac{\widehat{\mathbb{L}}_{n+1}^k}{\mathbb{A}_{n+1}} \right\rfloor + \rho_{n+1}^k$, with ρ_{n+1}^k a $\left(\frac{\widehat{\mathbb{L}}_{n+1}^k}{\mathbb{A}_{n+1}} - \left\lfloor \frac{\widehat{\mathbb{L}}_{n+1}^k}{\mathbb{A}_{n+1}} \right\rfloor\right)$-Bernoulli
   (b) Resample: $\mathbb{L}_{n+1}^{\mathbb{N}_{n+1}+j} := \mathbb{A}_{n+1}, \mathbb{X}_{n+1}^{\mathbb{N}_{n+1}+j} := \widehat{\mathbb{X}}_{n+1}^k$ for $j := 1, ..., \mathbb{N}_{n+1}^k$
   (c) Add Offspring Number: $\mathbb{N}_{n+1} := \mathbb{N}_{n+1} + \mathbb{N}_{n+1}^k$

(6) Non-resample Case: If $\widehat{\mathbb{L}}_{n+1}^k \in (a_n\mathbb{A}_{n+1}, b_n\mathbb{A}_{n+1})$ then
$\mathbb{N}_{n+1} := \mathbb{N}_{n+1} + 1$, $\mathbb{L}_{n+1}^{\mathbb{N}_{n+1}} := \widehat{\mathbb{L}}_{n+1}^k$, $\mathbb{X}_{n+1}^{\mathbb{N}_{n+1}} := \widehat{\mathbb{X}}_{n+1}^k$

Remark 2. *We extract our estimate before resampling to avoid excess noise. Key steps (5,6) determine the new number of particles $\mathbb{N}_{n+1}$ and weights $\mathbb{L}_{n+1}^k$ in an unbiased manner. When the prior weight $\widehat{\mathbb{L}}_{n+1}^k$ for particle k is extreme we do residual-style branching, splitting particles as deterministically as possible in (5). The result is zero or more particles all having the average weight at the same location as the parent. When the prior weight $\widehat{\mathbb{L}}_{n+1}^k$ is not extreme we run a weighted particle step in (6). The flexibility in this class of algorithms is in how we determine "extreme".*

Remark 3. $\mathbb{A}_{n+1}$*,* $\mathbb{L}_{n+1}^k$ *and* $\widehat{\mathbb{L}}_{n+1}^k$ *actually depend upon the initial number of particles N. Occasionally, we will stress this fact by relabeling $\mathbb{A}_{n+1}$ as $\mathbb{A}_{n+1}^N$.*

We are not the first to use branching particle filters for tracking. Indeed, we were inspired by Crisan and Lyons (1997), Crisan et. al. (1998) and Ballantyne et. al. (2000). However, our algorithms differ from the ones in those papers and our goals are also different.

After establishing the appropriate bounds on $\mathbb{N}_{n+1}$ in Theorem 2 to follow, we can easily see that this algorithm is also $O(N)$. Indeed, a careful comparison of this algorithm to the prior ones leads us to the believe that the constant implied in the $O(N)$ notation for the branching algorithm may be smaller than that for the bootstrap, especially when the Resampled Case does not occur too often. We will establish this fact experimentally below. Since σ_n is estimated both model selection and filtering can be done simultaneously without adding the additional step.

3.1. **Residual Branching Filter.** Like resampled filters, there are various ways to introduce the randomness, in this case the $\{\rho_{n+1}^k\}_{k=1}^{\mathbb{N}_n}$ for unbiased branching. The choice affects performance and implementation ease. A simple possibility is:

i) Let $\{\mathbb{U}_{n+1}^k\}_{k=1}^{\mathbb{N}_n}$ be independent $[0,1]$-Uniform RVs.

ii) Set $\rho_{n+1}^k := 1_{\mathbb{U}_{n+1}^k \le \left(\frac{\widehat{\mathbb{L}}_{n+1}^k}{\mathbb{A}_{n+1}} - \left\lfloor \frac{\widehat{\mathbb{L}}_{n+1}^k}{\mathbb{A}_{n+1}} \right\rfloor\right)}$.

In this way, the $\{\rho_{n+1}^k\}_{k=1}^{\mathbb{N}_n}$ are independent of each other and everything else. The reason for the *Residual* Branching labeling is, similar to Residual Resampling, we first create as many particles as we can deterministically and then we allocate the remaining offspring using independent uniform random variables. Note also we do not generate all $\mathbb{N}_n$ of the $\{\mathbb{U}_{n+1}^k\}$ and the $\{\rho_{n+1}^k\}$ since they are not used in Step (6) of the algorithm.

3.2. **Combined Branching Filter.** We can add stratified resampling to help control the number of particles and improve performance. When $a_n \approx b_n$, we:

i) Let $\{\mathbb{V}_{n+1}^k\}_{k=1}^{\mathbb{N}_n}$ be independent with $\mathbb{V}_{n+1}^k \sim \left[\frac{k-1}{\mathbb{N}_n}, \frac{k}{\mathbb{N}_n}\right]$-Uniform and $\mathbb{U}_{n+1}^k := \mathbb{V}_{n+1}^{p(k)}$, where p is a random permutation of $\{1, 2, ..., N_n\}$ uniformly distributed over the set of all permutations.

ii) Set $\rho_{n+1}^k := 1_{\mathbb{U}_{n+1}^k \le \left(\frac{\widehat{\mathbb{L}}_{n+1}^k}{\mathbb{A}_{n+1}} - \left\lfloor \frac{\widehat{\mathbb{L}}_{n+1}^k}{\mathbb{A}_{n+1}} \right\rfloor\right)}$.

Then, the $\{\rho_{n+1}^k\}_{k=1}^{\mathbb{N}_n}$ are exchangeable but not independent of each other. (They are actually negatively correlated, which is desireable for particle control.) The advantage of this approach is it is not possible to get mostly large or mostly small uniform random numbers so the number of particles will vary less.

In the usual case, where $a_n \ll b_n$ so many particles are not resampled, there is a better stratified method. We replace (5-6) in the basic branching algorithm with:

```
5) Non-resample count: $l := 0$
6) For $k := 1, 2, ..., \mathbb{N}_n$ do
   If $\widehat{\mathbb{L}}_{n+1}^k \notin (a_n \mathbb{A}_{n+1}, b_n \mathbb{A}_{n+1})$ then: $\widehat{\mathbb{L}}_{n+1}^{k-l} := \widehat{\mathbb{L}}_{n+1}^k, \widehat{\mathbb{X}}_{n+1}^{k-l} := \widehat{\mathbb{X}}_{n+1}^k$
   Otherwise: $l := l + 1$, $\mathbb{L}_{n+1}^l := \widehat{\mathbb{L}}_{n+1}^k$, $\mathbb{X}_{n+1}^l := \widehat{\mathbb{X}}_{n+1}^k$
7) Let $\mathbb{N}_{n+1} := l$, $\{\mathbb{V}_{n+1}^k\}_{k=l+1}^{\mathbb{N}_n}$ be independent with $\mathbb{V}_{n+1}^k \sim \left[\frac{k-1}{\mathbb{N}_n - l}, \frac{k}{\mathbb{N}_n - l}\right]$-
   Uniform, $p$ be a random permutation of $\{l+1, l+2, ..., \mathbb{N}_n\}$, $\mathbb{U}_{n+1}^k := \mathbb{V}_{n+1}^{p(k)}$
8) For $k := l+1, l+2, ..., \mathbb{N}_n$ do
   $\mathbb{N}_{n+1}^k := \left\lfloor \frac{\widehat{\mathbb{L}}_{n+1}^{k-l}}{\mathbb{A}_{n+1}} \right\rfloor + 1_{\mathbb{U}_{n+1}^k \le \left(\frac{\widehat{\mathbb{L}}_{n+1}^{k-l}}{\mathbb{A}_{n+1}} - \left\lfloor \frac{\widehat{\mathbb{L}}_{n+1}^{k-l}}{\mathbb{A}_{n+1}} \right\rfloor\right)}$
```

$\mathbb{L}_{n+1}^{\mathbb{N}_{n+1}+j} := \mathbb{A}_{n+1}, \mathbb{X}_{n+1}^{\mathbb{N}_{n+1}+j} := \widehat{\mathbb{X}}_{n+1}^{k-l}$ for $j := 1, ..., \mathbb{N}_{n+1}^k$
$\mathbb{N}_{n+1} := \mathbb{N}_{n+1} + \mathbb{N}_{n+1}^k$

3.3. **Resampling Control.** To put our claim of improved particle control into perspective, we first consider the issue with popular earlier-generation branching particle algorithms. Of course, we do not include the constant-particle, minimum-variance branching algorithm as we have already re-labeled that as a resampled method. Instead, we look at the (complete) branching algorithms as presented in Crisan et. al. (1999). In the proof of Proposition 2.1 of this paper, it is shown that their branching particle filters satisfy

$$E[\mathbb{N}_{n+1}|\mathcal{F}_n] = \mathbb{N}_n, \tag{3.1}$$

where $\mathcal{F}_n$ is all the information about the particle system up to time n and $\mathbb{N}_n$ is the number of particles alive at time n. This equation implies that, if the number of particles starts to drift up or down, then there is nothing to pull it back. Invariably the number of particles become too small or too large to be workable at some time. Switching to our algorithms, we find the extreme complete branching case is when $a_n = b_n = 1$. Then, examining our branching algorithm, we find (by Step (5a)) that the total expected number of particles after resampling is:

$$\sum_{j=1}^{\mathbb{N}_n} E[\mathbb{N}_{n+1}^j | \sigma\{\mathbb{N}_n, \widehat{\mathbb{L}}_{n+1}^1, ..., \widehat{\mathbb{L}}_{n+1}^{\mathbb{N}_n}\}] = \sum_{j=1}^{\mathbb{N}_n} \frac{N \widehat{\mathbb{L}}_{n+1}^j}{\sum_{k=1}^{\mathbb{N}_n} \widehat{\mathbb{L}}_{n+1}^k} = N \tag{3.2}$$

regardless of what $\mathbb{N}_n$ is or whether residual or combined branching is used. This is very different than the prior algorithms since we always expect to return to the initial number of particles not the last number. This particle control becomes yet more pronounced when combined with the negatively correlated ρ's produced by the stratified scheme within the combined branching particle algorithm. Moreover, in the other extreme case, we choose $a_n = 0$ and $b_n = \infty$ and find that no resampling takes place. We then have the weighted particle filter, which has a constant number of particles.

In general, the amount of and conditions for branching is controlled by the parameters a_n, b_n. Suppose that $a_n = 0$ and $b_n = 1$. Then, we only branch the below-average weight particles and expect fewer particles after branching. Hence, there needs to be some condition to keep the number of particles constant. That condition can easily be verified to be:

$$\frac{\sum_{j=1}^{\mathbb{N}_n} 1_{\widehat{\mathbb{L}}_{n+1}^j \in (a_n \mathbb{A}_{n+1}, b_n \mathbb{A}_{n+1})} \widehat{\mathbb{L}}_{n+1}^j}{\sum_{j=1}^{\mathbb{N}_n} 1_{\widehat{\mathbb{L}}_{n+1}^j \in (a_n \mathbb{A}_{n+1}, b_n \mathbb{A}_{n+1})}} = S_{n+1}^N(1), \tag{3.3}$$

that is the average weight of the particles that do not branch is equal to the average weight that do. This condition is difficult to ensure in practice but we can clearly limit our interest to $0 \le a_n \le 1 \le b_n$ if we want some semblance of particle control.

In practice, we have found good results in keeping the number of particles fairly constant when a_n and b_n have geometric center $\sqrt{a_n b_n}$ of 1. This is related to the facts that the weights start at 1, multiply as the observations arrive, are non-negative and have mean 1. Even though it works well in practice, we are not ready to suggest that geometric center of 1 property should be a rule. Rather, it should be an area of future investigation. Still, we will stick to the geometric center 1 here and take $a_n = \frac{1}{r_n}$ and $b_n = r_n$ for some $r_n \in [1, \infty]$. More unbiased particle control could be added to retain even particle numbers. However, we are yet to see an example where it is warranted (see the results below).

The new multiplicative weight update has the form $\alpha_n(X_{n-1}) = \frac{g(Y_n - h(X_{n-1}))}{g(Y_n)}$, which adapts for noisy observations. However, it does not account well for only having partial measurements of the signal. As a simple example, if we have range only measurements of a position-velocity model in the plane, then one observation has no information about velocity and only partial measurement of position. If complete resampling were used, then particles would (with high probability) accumulate on an arc with matching velocities and other positional component. If these velocities are all wrong (which can easily happen with a finite number of particles), then the majority of the particles will head in the wrong direction.[2] To avoid these types of situations, one should generically choose a larger r_n when it would take several observations to get a good idea about whole signal. Indeed, we are yet to see a problem where either no resampling or full resampling (like the resampled particle filters) preforms best. Hence, we expect to use an $r_n \in (1, \infty)$.

In our Residual and Combined methods, we have taken $r_n \equiv r$ to be constant but that can be a poor choice. For example, it is very likely that a different amount of resampling should be done in the situation of a few high weights, reflecting (possibly false) confidence that these particles are better than the others, compared to fairly even weights. We have adaptive resampling when r_n depends upon the branching particle system. One example of this is *dynamic branching*, where

$$r_n = \exp\left(c \left[\frac{1}{N_n} \sum_{k=1}^{N_n} (\ln \widehat{\mathbb{L}}_{n+1}^k)^2 - \left(\frac{1}{N_n} \sum_{k=1}^{N_n} \ln \widehat{\mathbb{L}}_{n+1}^k \right)^2 \right]^{\frac{q}{2}} \right),$$

with $c, q > 0$. To understand r_n, we note that we start with a variance in the square brackets, which must be non-negative, take it to the power $\frac{q}{2}$ and then multiply it by positive constant c so r_n must be at least 1. A larger $q > 1$ (smaller $q < 1$), with c adjusted to maintain the same average amount of branching, would be used in the situation where one wanted more resampling when system entropy low (high).[3] To explain, we imagine $\xi^k = \ln \widehat{\mathbb{L}}_{n+1}^k - \ln \mathbb{A}_{n+1}$ are independent, zero-mean and Gaussian. Then, r_n with $q = 1$ would correspond to a relatively fixed (67% when $c = 1$) number of particles being resampled regardless of disorder. Taking $q < 1$ and compensating with $c > 1$ to maintain the same average amount of resampling

[2]Similarly, for bearing-only measurements the particles would accumulate in a line of sight, matching state information of a few particles that could be wrong.

[3]System entropy refers to the amount of disorder or unevenness in the weights, which in turn indicates wide variety in perceived quality of the particles' states.

would then cause more resampling at times when there is more entropy. This is investigated experimentally below. Actually, it is not clear whether more or less resampling[4] should be done in the high entropy case as is explained immediately below. Hence, the prudent thing to do is let the data tell us which q are good for a problem of interest.

Another, more direct, way to handle uneven weights is through the *effective* number of particles estimate, N^{eff}, as discussed in Doucet et. al. (2000). In our setting, the effective and non-effective particle estimates are:

$$\mathbb{N}_{n+1}^{eff} = \frac{N^2 \mathbb{A}_{n+1}^2}{\sum\limits_{k=1}^{\mathbb{N}_n} \left(\widehat{\mathbb{L}}_{n+1}\right)^2}, \quad \mathbb{N}_{n+1}^{noneff} = \mathbb{N}_{n+1} - \mathbb{N}_{n+1}^{eff}.$$

It very reasonable to anticipate better results when branching either more or fewer particles in the situation there are few effective ones. A first intuition might lead us to the conclusion that it is better to branch more in order to obtain more effective particles immediately. However, reviewing the range (and bearing) only example mentioned previously also leads us to the realization that this too could be wrong. We do not assume either a priori but rather in *effective particle branching* set

$$r_n = \frac{c^{eff}\mathbb{N}_{n+1}^{eff} + c^{noneff}\mathbb{N}_{n+1}^{noneff}}{\mathbb{N}_{n+1}} = c^{noneff} + \left(c^{eff} - c^{noneff}\right)\frac{\mathbb{N}_{n+1}^{eff}}{\mathbb{N}_{n+1}} \tag{3.4}$$

for experimentally determined constants c^{eff}, $c^{noneff} > 0$ and let the data decide.

3.4. **Stability and Particle Control.** Most authors have rejected branching particle filters due to possible instability of the number of particles as well as the computational consequences of this instability. Perhaps, this is due to the particle number drift property highlighted in (3.1) for earlier-generation branching particle filters. This rejection was too hasty. Yes, our algorithms too can fail. During resampling, there is a possibility of immediately killing all particles if $\max\limits_{j \le \mathbb{N}_{n-1}} \frac{N\widehat{\mathbb{L}}_n^j}{\sum_{i=1}^{\mathbb{N}_n} \widehat{\mathbb{L}}_n^i} < 1$. Ironically, this can only happen if there are more particles than at start. However, it may still be possible to degenerate immediately to one particle when $\mathbb{N}_n \le N$. Conversely, it is not possible to increase by more than $N-1$ particles in one step. Weight variation is also a big concern: $\mathbb{L}_n^j$ can become very uneven as N increases. Some regularity results are required to ensure that there are enough effective particles and moment bounds to justify the anticipation of rate of convergence results as $N \to \infty$.

The following *uniform bounds* ensure the risk of such system irregularity decreases exponentially in the initial number of particles, a first step in *disproving* the opinion *branching particle filters are unstable in the number of particles.* Let $B(\mathbb{R}^d)$ be the bounded and $C_{++}(\mathbb{R}^d)$ be the strictly-positive continuous functions on $\mathbb{R}^d$.

[4]This decision may well be problem dependent.

Theorem 2. *Suppose residual branching with* $0 \le a_n < b_n \le \infty$*;* $h \in B(\mathbb{R}^d)$*; and* $g \in C_{++}(\mathbb{R}^d)$*. Then, there are* $\epsilon_n > 0$*,* $C_n > 1$ *and* $\mathbb{D}_n^N \in \sigma\{\mathbb{N}_l,\ l \le n\}$ *such that* $\mathbb{D}_{n+1}^N \subset \mathbb{D}_n^N$ *for all* $n = 0, 1, 2...$*;* $Q^Y(\mathbb{D}_n^N) \ge 1 - 2ne^{-\epsilon_n N}$ *for* $N \ge 1$*; and*

$$\frac{1}{C_n} \le \frac{\mathbb{N}_l}{N}, \mathbb{L}_l^i, \mathbb{A}_l^N \le C_n \ \forall i \in \{1, ..., \mathbb{N}_l\}, l \in \{0, ..., n\} \text{ on } \mathbb{D}_{n-1}^N.$$

For ease of assimilation, we provide a direct corollary of this Theorem. The second claim follows from (1.1) and Bayes rule.

Corollary 1. *Supppose the conditions of Theorem 2 hold. Then, for any fixed* n *there are* $\epsilon_n > 0$*,* $C_n > 1$ *and* $K_n > 1$ *such that*

$$Q^Y\left(\frac{1}{C_n} \le \frac{\mathbb{N}_l}{N}, \mathbb{L}_l^i, \mathbb{A}_l^N \le C_n \ \forall i \in \{1, ..., \mathbb{N}_l\}, l \in \{0, ..., n\}\right) \ge 1 - 2ne^{-\epsilon_n N}$$

and

$$P\left(\frac{1}{C_n} \le \frac{\mathbb{N}_l}{N}, \mathbb{L}_l^i, \mathbb{A}_l^N \le C_n \ \forall i \in \{1, ..., \mathbb{N}_l\}, l \in \{0, ..., n\} \middle| \mathcal{F}_n^Y\right) \ge 1 - K_n e^{-\epsilon_n N}$$

for $N = 1, 2, ...$

This shows the probability of approaching extinction or explosion decays exponentially in the initial number of particles.

Remark 4. *There are* no new assumptions *in Theorem 2 as the sensor function* h *was bounded and the noise density* g *was strictly postive and continuous in our Girsanov's theorem. This setting is reasonably general, handling both the Cauchy and Normal densities used in our examples. Moreover, our purpose is to introduce novel and superior particle methods in a clear a manner and leave potential technical generalizations (of these bounds and the algorithms) to future work. Still, it is acknowledged that neither the Girsanov theorem stated here nor Theorem 2 are as general as possible but rather are stated under conditions that will prove convenient for establishing a central limit theorem.*

Remark 5. *There are two vital aspects to this result: 1) The number of particles and weights remain uniformly bounded up until any fixed final time of interest* n*. (The value of* C_n *may depend upon* n*.) These are theoretical bounds established without any model specifics that are meant to supplement the observed simulation bounds (given immediately below). These bounds will be used in further theoretical work that will give more precise probabilistic behaviour of the algorithm. 2) The theorem also says the algorithm remains well behaved for at least one step into the future on the large exchangeable set* $\mathbb{D}_n^N$*. This part of the theorem will be important for future theoretical work like Marcinkiewicz law of large numbers and central limit theorems for these algorithms.*

Remark 6. *This result is for residual branching but adding stratification as in combined, dynamic and effective particle branching reduces particle number fluctuation through the negatively correlated* ρ*'s. This is illustrated below for* combined *branching.*

Remark 7. *This theorem is proved in Appendix B.*

We now look at particle variation experimentally. The expected number of particles is always the initial number N for complete branching by (3.2). Figure 4 shows that there is very little variation around the mean over long periods of time. We plot the standard deviation as a percentage of the initial number of particles N over the first 5000 time points versus method and N.

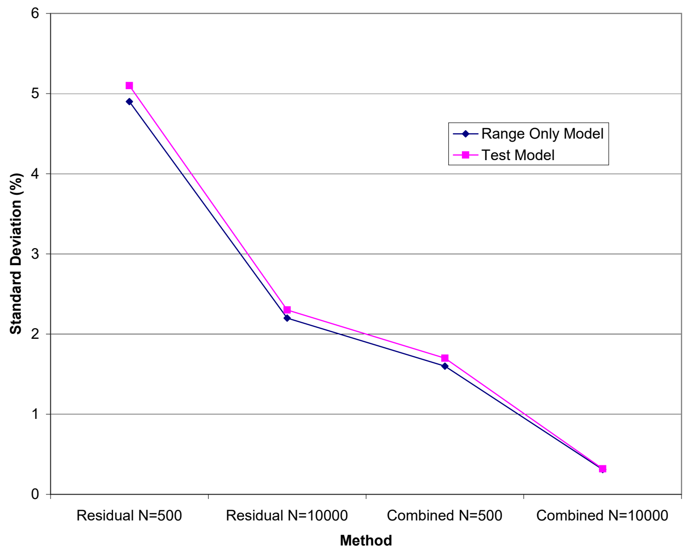

FIGURE 4. Branching Particle Number Variation over 5000 Time Steps (s)

Figure 4 shows the particle number fluctations are very modest, especially for the combined method where negative correlations are used. It also shows that the relative fluctations decrease in the initial number of particles used. While these results are random, they were typical of several retrials of these and other experiments. Certainly, the results to follow on speed and performance are supportive of small particle variations.

We summarize two of the particle variation experiments performed: When $N = 500$ initial particles were used with the Range Only model (defined in Section 4), the Residual Branching particle numbers were in the range $473-522$ approximately 67% of the 5000 time points, meaning one standard deviation was 24.5 or 4.9%. The Combined Branching algorithm showed a significant improvement, having the typical range of $492-508$ and one standard 8 or 1.6% over the 5000 points. When $N = 10,000$ initial particles were used in the Test model the one standard deviation rangers were $9739-10181$ and $9968-10032$ for the residual and combined methods respectively. (These correspond to 2.21% and 0.32% variations).

We can make the following experimental conclusions:

(1) The particle numbers did not vary much, provided enough particles were used to be able to estimate the signal reasonably. Indeed, we will see below that the particle variations did not adversely affect error nor execution speed.

(2) Combined Branching did a better job of keeping the number of particles constant than the basic Residual Branching.
(3) The relative variation of the number of particles decreased in the number of initial particles. It is speculated that a law of large numbers effect takes place.
(4) While it is easy to add (extra, unbiased) particle control to keep even particle numbers, there is no evidence of need from these experiments. There is already some particle control through the constant expected number of particles and in the negative correlations when using stratified random variables.

4. Comparison of Tracking and Model Selection

Effective parallelization of resampled particle filters is difficult (see e.g. Vergé et. al. (2013) for a good attempt). This is reason enough to choose our branching particle filters over the resampled ones as they have a strong advantage[5] when parallelizing (as will be shown in future work). However, we can also consider all algorithms with single-processor implementations on tracking problems and on model selection. The rest of this section is organized as follows: We first introduce the two simple problems, our *Test* and *Range-Only* problems, that will be used for comparison purposes. Then, we compare the various resampled particle systems discussed above on these problems. Next, we compare the worst of our branching algorithms to the best resampled particle system discussed above and show even this most basic branching algorithm significantly outperforms all resampled particle systems. Finally, we compare all our branching algorithms to determine which variation performs the best. For consistency, all results herein are either a *typical path* or an *average over 200 different sample paths* of *35 time steps.* (We reduce our number of time steps from 5000 in Subsection 3.4 down to 35 because of the inefficiency of the bootstrap and other resampled particle systems. We could have easily considered a much larger number of time steps if we were only running our branching particle filters. However, it took us weeks to get simulations we needed for the resampled particle systems on our limited computer resources.)

4.1. **Test Model.**

The *Test Model* refers to the scalar signal and observation pair:

$$X_n = 0.95X_{n-1} + 0.3W_n, \quad Y_n = X_{n-1} + V_n,$$

where X_0, $\{W_n\}$ and $\{V_n\}$ are independent with standard Cauchy distribution. This is a linear, non-Gaussian filtering problem. The Kalman filter does not apply since the noise is Cauchy. Indeed, conditional expectations of state do not exist since the noise is so heavy-tailed. However, this problem is in other respects simple.

For model selection, we introduce alternative models and show that we select the correct one. We keep most of the *Test Model* the same and just vary two coefficients:

[5]When a particle branches it only needs its own information as well as the total weight of all particles. It does not need the locations nor weights of other particles.

Model Number	Signal Equation	Observation Equation
-2	$X_n = 0.93X_{n-1} + 0.28W_n$	$Y_n = X_{n-1} + V_n$
-1	$X_n = 0.94X_{n-1} + 0.29W_n$	$Y_n = X_{n-1} + V_n$
0	$X_n = 0.95X_{n-1} + 0.3W_n$	$Y_n = X_{n-1} + V_n$
1	$X_n = 0.96X_{n-1} + 0.31W_n$	$Y_n = X_{n-1} + V_n$
2	$X_n = 0.97X_{n-1} + 0.32W_n$	$Y_n = X_{n-1} + V_n$

Hence, the real model is model 0, the null model.

4.2. **Range Only Model.** The *Range Only Model* refers to the four dimensional signal and scalar observation model:

Position	*Velocity*
$X_n = \alpha X_{n-1} + U_{n-1} + 0.3\alpha_n$	$U_n = 0.95U_{n-1} + \gamma_{n-1}$
$Z_n = \alpha Z_{n-1} + V_{n-1} + 0.3\beta_n$	$V_n = 0.95V_{n-1} + \theta_{n-1}$

$$\alpha = 0.5, \quad Y_n = \sqrt{X_{n-1}^2 + Z_{n-1}^2} + 0.1\psi_n,$$

TABLE 1. Range Only Model

where X_0, Z_0, U_0, V_0, $\{\gamma_n\}$, $\{\theta_n\}$, $\{\alpha_n\}$, $\{\beta_n\}$, $\{\psi_n\}$ are independent. X_0, Z_0 have 10 times the standard Cauchy distribution and U_0, V_0 have 5 times the standard Normal distribution. Signal noise sources γ_n and θ_n have standard normal distribution while α_n and β_n have standard Cauchy distribution. ψ_n is the observation noise with standard Cauchy distribution. This is a nonlinear problem but is otherwise simple.

The idea of this model comes from radar detection. Suppose that there is a radar station at the origin in the plane, (X_n, Z_n) describes the position of a ship and (U_n, V_n) its velocity. The radar produces a noise-corrupted distance observation between the ship and itself, which is Y_n in our model. The objective of this problem is to estimate the state of the ship using the (back) observations.

For model-selection alternative models, we will keep most of the *Range Only Model* the same and just vary the coefficient in front of X_{n-1} and Z_{n-1} slightly:

Model Number	Signal Parameter	Observation Equation
-2	$\alpha = 0.48$	$Y_n = \sqrt{X_{n-1}^2 + Z_{n-1}^2} + 0.1\psi_n$
-1	$\alpha = 0.49$	$Y_n = \sqrt{X_{n-1}^2 + Z_{n-1}^2} + 0.1\psi_n$
0	$\alpha = 0.50$	$Y_n = \sqrt{X_{n-1}^2 + Z_{n-1}^2} + 0.1\psi_n$
1	$\alpha = 0.51$	$Y_n = \sqrt{X_{n-1}^2 + Z_{n-1}^2} + 0.1\psi_n$
2	$\alpha = 0.52$	$Y_n = \sqrt{X_{n-1}^2 + Z_{n-1}^2} + 0.1\psi_n$

Hence, the real model is model 0 with the others differing slightly through α.

4.3. **Comparison within Resampled Particle Systems.** First, we compare resampled particle systems based on error (of root-mean-square type between positional tracking estimation and the real value) versus execution time. For our *Test Model*, the error is defined as

$$error = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(\pi_k^N(f) - f(X_k))^2}, \quad f(x) = \begin{cases} 30 & : \quad x > 30 \\ x & : \quad -30 \leq x \leq 30 \\ -30 & : \quad x < -30 \end{cases} .$$

where π_k^N is the filter approximation at time instant k and N particles and X_k is the signal. The results for the *Test Model* are shown in Figure 5 for the algorithms defined in Section 2. The actual points shown are for $N = 100$, 400, 2000 and 10000 for the Test Model.

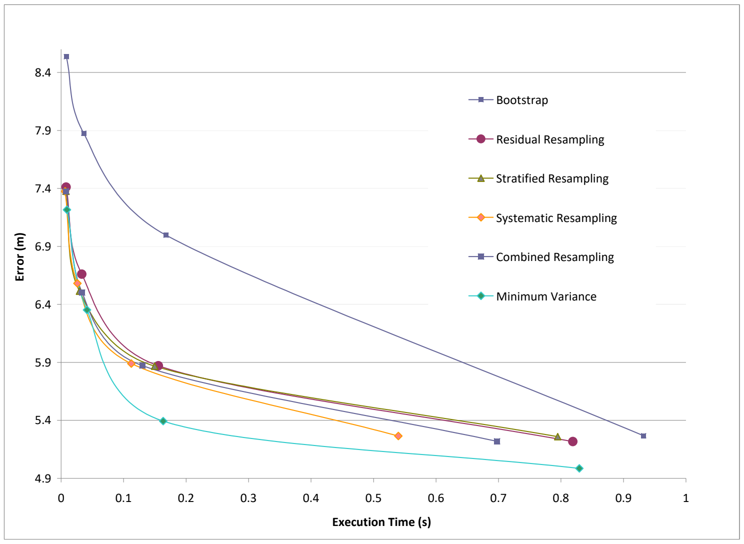

FIGURE 5. Resampled Particle Filters on Test Model

All five improved resampling methods show a significant improvement over the bootstrap. All these methods approach the optimal filter as the number of particles, hence execution time, increases. In difficult real life problems, one often has to limit the execution time and performance with limited execution time is important.

Residual and combined resampling preserve some particles without resampling saving some of the resampling computations. The stratified, combined and systematic resampling, save computations related to ordering the uniform random variables in the bootstrap method. For simple signal models (like Test), a large portion of the time is consumed generating and ordering the uniform resampling random numbers. This is efficiently done with stratification so it is reasonable that stratified, combined and systematic method can improve the speed of the *Test Model* greatly. The minimum variance method's power allows comparison with fewer particles hence fewer random numbers and less ordering.

For our *Range Only Model*, the error is

$$error = \frac{1}{n}\sum_{k=1}^{n}\sqrt{(\pi_k^N(g_x) - g(X_k))^2 + (\pi_k^N(g_z) - g(Z_k))^2},$$

where π_k^N is the normalized filter approximation at time instant k and N initial particles, X_k, Z_k are the positional components of the real signal,

$$g(x) = \begin{cases} 1000 & : \quad x > 1000 \\ x & : \quad -1000 \le x \le 1000 \\ -1000 & : \quad x < -1000 \end{cases}.$$

and g_x, g_z denote g applied to the x and z (positional) components of the signal. The result for *Range Only Model* are shown in Figure 6 for the algorithms defined in Section 2. The actual points shown are 500, 2000, 10000 and 50000 for the Range Only Model.

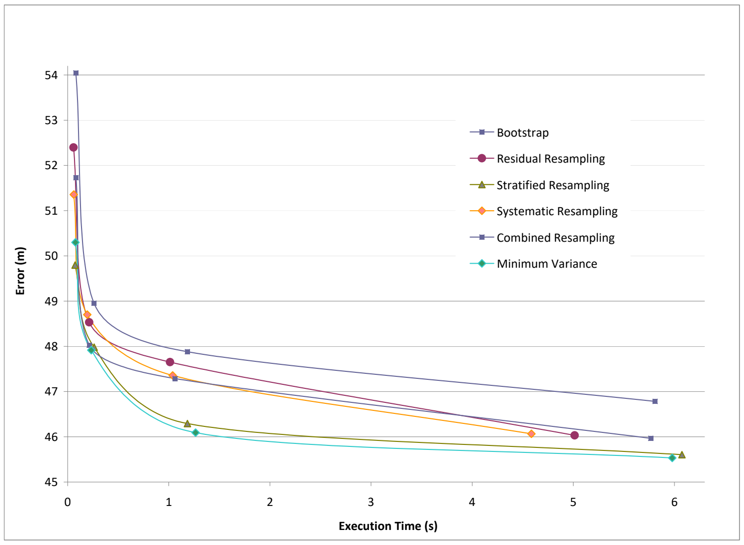

FIGURE 6. Resampled Particle Filters on Range Only Model

With slightly larger signals such as our *Range Only Model*, which has a four dimensional signal, a lot of time is spent copying particles. Thus, the execution time may depend more on how many particles need to be copied or resampled. Hence, it is also reasonable that residual and combined methods, which reduce the number of particles resampled, can improve execution time. Naturally, the systematic method is very fast as it only uses one uniform random variable. These speed factors are reflected in the corresponding points in Figure 6 being further left.

To combine performance and speed, we define the "Bootstrap Factor" as:

$$Bootstrap \quad Factor = \frac{t_{bootstrap}}{t},$$

where $t_{bootstrap}$ and t are the execution times for the bootstrap and method of interest to reach a fixed error. We use $error = 5.0$ in the *Test Model* and $error = 46.0$ in the *Range Only Model*, then show the minimum particle number, execution time and "Bootstrap Factor" for both in Tables 2 and 3.

	N	**Time (s)**	**Bootstrap Factor**
Bootstrap	20000	1.9152	1
Residual Resampling	20000	1.6399	1.1679
Stratified Resampling	10000	0.7947	2.4099
Systematic Resampling	25000	1.5582	1.2291
Combined Resampling	10000	0.6976	2.7454
Minimum Variance Scheme	9500	0.8131	2.3554

TABLE 2. Bootstrap Factor of Test Model with fixed $Error = 5.0$

The Bootstrap Factor compares speed of a method to the bootstrap filter for a given performance, combining accuracy and efficiency factors. Combined Resampling is the best method for both models with Bootstrap Factors of 2.7454 and 6.5116. However, *every* branching algorithm will significantly outperform this.

	N	**Time (s)**	**Bootstrap Factor**
Bootstrap	60000	6.9134	1
Residual Resampling	50000	5.0141	1.3788
Stratified Resampling	10000	1.1844	5.8522
Systematic Resampling	50000	4.5857	1.5076
Combined Resampling	10000	1.0617	6.5116
Minimum Variance Scheme	10000	1.2641	5.4690

TABLE 3. Bootstrap Factor of Range Only Model with fixed $Error = 46.0$

4.4. **Comparison between Branching and Resampled Particle Systems.** In this section, we compare the best resampled particle systems to the most basic of our new branching systems. We will use the same models, error definitions and numbers of particles as in previous sections. We show that the new branching algorithms can improve both performance and execution time. For now, we use $(a_n, b_n) = (1/r, r)$, where $r \in [1, \infty]$ is referred to as the branching parameter. All particles will branch when $r = 1$, which we call complete resampling. No particle will branch when $r = \infty$, which means we have the weighted particle filter. We found a good fixed choice of r for the *Test Model* was 2.25 and for the *Range Only Model* was 5. (Later, we will explore better methods with state-dependent r.)

The results for the *Test Model* are shown in Figure 7 for the algorithms defined in Section 3. The actual points shown are for $N = 100$, 400, 2000 and 10000 particles.

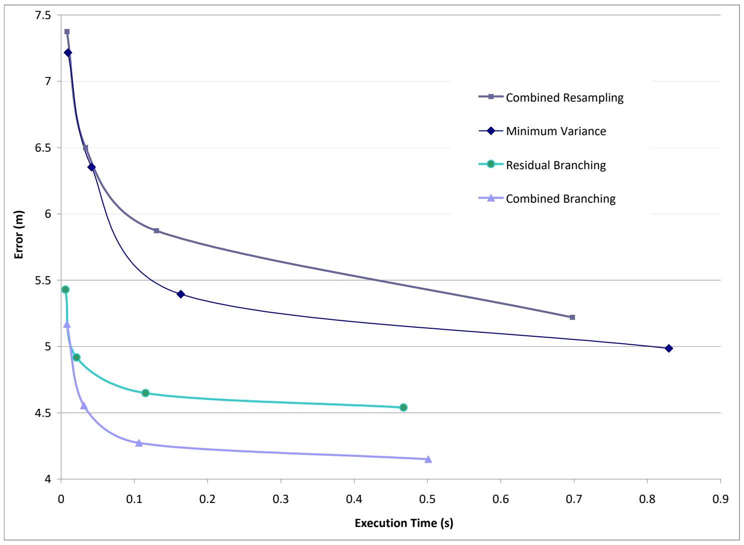

FIGURE 7. Branching Filter Improvement on Test Model

Residual branching is much better than the best resampled systems, combined resampling and minimum variance. Combined branching is yet better. A close inspection shows that the new branching methods win on both speed and performance (for fixed particles), especially speed.

The result for *Range Only Model* are shown in Figure 8 for the algorithms defined in Section 3. The actual points shown are for $N = 500$, 2000, 10000 and 50000.

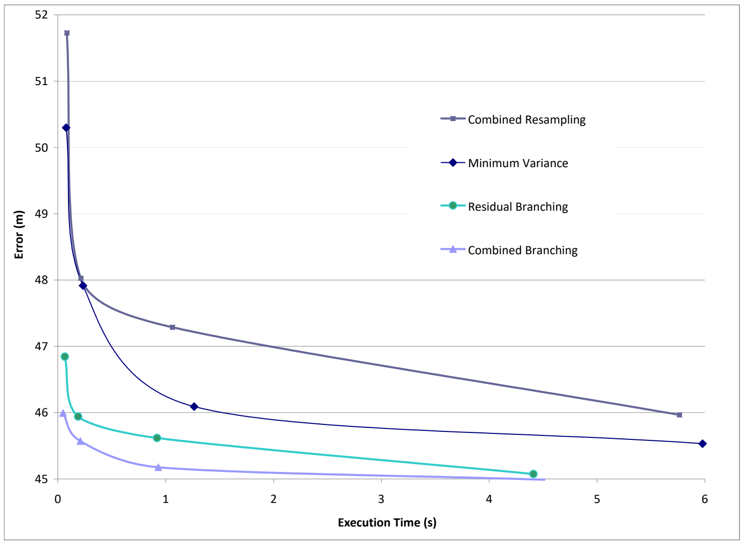

FIGURE 8. Resampled Particle Filters on Range Only Model

Again, the basic new branching algorithms demonstrate a significant improvement over the resampled systems.

To evaluate the advantage of branching on both performance and speed, we provide the Bootstrap Factor in Tables 4 and 5. In the *Test Model*, residual branching is 91.2 and 33.2 times better than bootstrap and combined resampled respectively. In *Range Only Model*, the improvement is also significant at 36.7148 and 5.64. Our better branching algorithms will be shown below to outperform yet a lot more.

	N	**Time (s)**	**Bootstrap Factor**
Bootstrap	20000	1.9152	1
Combined Resampling	10000	0.6976	2.7454
Residual Branching	400	0.0210	91.2000
Combined Branching	150	0.0085	225.3176

Table 4. Bootstrap Factor of Test Model with Error 5.0

	N	**Time (s)**	**Bootstrap Factor**
Bootstrap	60000	6.9134	1
Combined Resampling	10000	1.0617	6.5116
Residual Branching	2000	0.1883	36.7148
Combined Branching	500	0.0480	144.0292

Table 5. Bootstrap Factor of Range Only Model with Error 46

Model selection ability is also extremely important. For comparison purposes, we fix the initial number of particles to be $N = 10,000$ for all model selection experiments and show the execution time for model selection in Table 6. Residual branching is the fastest for model selection as it was for tracking. Indeed, branching has another small inherent advantage here since model selection is based upon the unnormalized filter, which is already computed in the branching methods.

Model	**Test Model**	**Range Only Model**
Bootstrap	1.213	1.613
Combined Resampling	0.960	1.602
Residual Branching	0.736	1.347

Table 6. Average Execution Time (s) of Model Selection

Inasmuch as the results are similar for both the Test and Range Only models, we just demonstrate these three algorithm on our *Range Only Model*. Define *Bayes Factor* $= \frac{\sigma^0(1)}{\sigma^k(1)}$, where $k \in \{-2, -1, 0, 1, 2\}$ is the index for the different models described in Subsections 4.1 and 4.2. All three algorithms select the correct model convincingly. It appears from the pictures that bootstrap had the hardest time distinguishing models, Combined Resampling distinguished the correct model from model 1 the best, while Residual Branching distinguished the other three best.

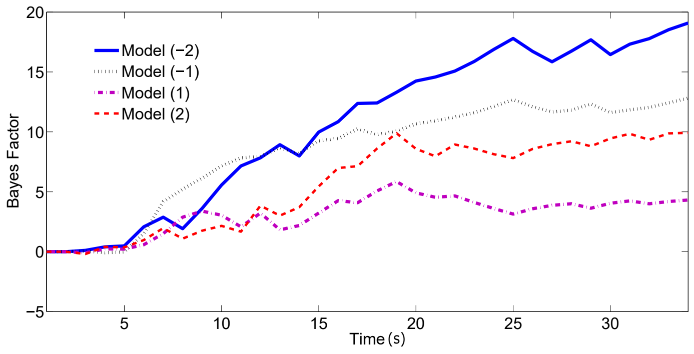

FIGURE 9. Bootstrap Model Selection of *Range Only Model*

Typically, bootstrap has the most difficult time distinguishing models and residual branching is slightly better at distinguishing models than combined resampling.

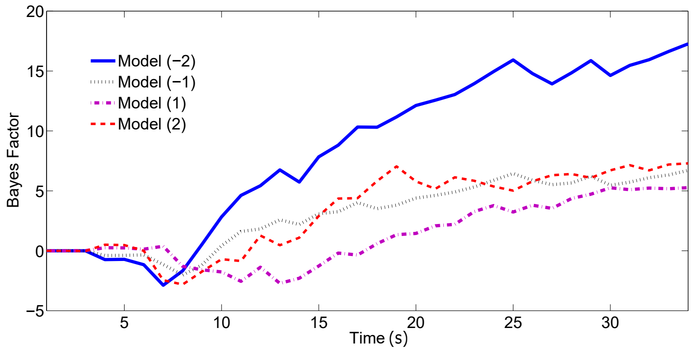

FIGURE 10. Combined Resampling Model Selection of *Range Only Model*

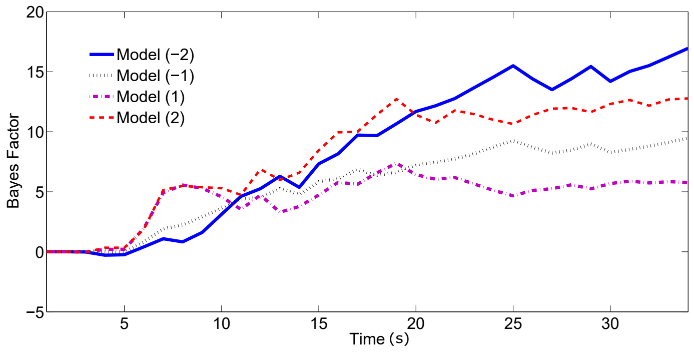

FIGURE 11. Residual Branching Model Selection of *Range Only Model*

4.5. **Comparison within Branching Particle Systems.** There are many ways to produce unbiased branching filters. The residual branching and combined branching algorithms, introduced in Subsections 3.1 and 3.2, are two of the simplest to implement. Hitherto, we have taken a_n and b_n in these algorithms constant. However, the performance and speed can improve if a_n and b_n depend on the system state at time step n. Hence, now we consider a dynamic r_n such that $a_n = \frac{1}{r_n}$ and $b_n = r_n$ will depend on system state in two different ways. We call these two implementations Dynamic Branching and Effective Particle Branching as mentioned in Subsection 3.3.

We continue performance versus execution time and "Bootstrap Factor" but now within branching particle systems. Insomuch as the conclusions to be formed on the Test and Range Only models would be very similar, we conserve space and only present our experimental results on the dynamic and effective particle methods for the Range Only model. In particular, we apply residual branching (with $r = 5$), combined branching (with $r = 5$), dynamic branching and effective particle branching to our Range Only Model. It turns out that this good choice of $r = 5$ translates into branching about three quarters of the time in the residual and combined methods. Also, since combined branching (where stratified random numbers are used) outperforms residual branching, we use both residual and stratified techniques within dynamic and effective particle branching. For clarity, these approaches differ from combined branching by replacing a fixed r with a state dependent r_n as defined in Subsection 3.3.

For *dynamic branching*, we set q to be different value and find the best c, which means the minimum error, and show the results in Table 7. It shows that the resampling percentage is always around 76%, which is a similar amount as used in with the good choice of static r in the residual and combined branching filters.

q	**c**	**Error (m)**	**resampling percent**
0.5	0.80	46.3298	76.19%
1.0	0.60	45.8457	76.21%
1.5	0.36	45.9732	76.51%
2.0	0.25	46.3665	76.52%

TABLE 7. Average Error of dynamic branching method with 500 particles

However, this table demonstrates that the conditions for branching, not just the overall amount of branching, affect error. The error is minimal with $q = 1$, which corresponds to a constant amount of branching regardless of system entropy. The Residual and Combined methods would correspond to the case of more branching when there is more entropy and the case $q = 2$ would correspond to the case of more branching when there is less entropy. Indeed, we will see below that this dynamic branching with $q = 1$ beats combined branching at all particle numbers considered.

For effective particle branching, we consider the following c^{noneff} and c^{eff} choices:

c^{eff}	c^{non}	**Error (m)**	c^{eff}	c^{non}	**Error (m)**	c^{eff}	c^{non}	**Error (m)**
1	2	64.5101	1	16	45.7550	2	8	45.9141
2	1	46.2599	16	1	46.4342	8	2	47.7404
1	4	45.9427	1	32	48.2642	2	16	45.8674
4	1	53.0337	32	1	50.0621	16	2	53.2428
1	8	45.9082	2	4	56.0850	2	32	46.5672
8	1	46.1202	4	2	49.2555	32	2	54.7030

TABLE 8. Average Error with Different c^{eff} and c^{noneff} and 500 particles

The results in Table 8 show that $c^{eff} = 1$ and $c^{noneff} = 16$ for r_n in (3.4) is the best choice to minimize error. This means that more resampling would be done when there are fewer effective particles for the Range Only problem.

We compare residual, combined, dynamic and effective particle branching errors versus execution time in Figure 12. The actual points shown are for $N = 500$, 2000, 10000 and 50000 initial particles.

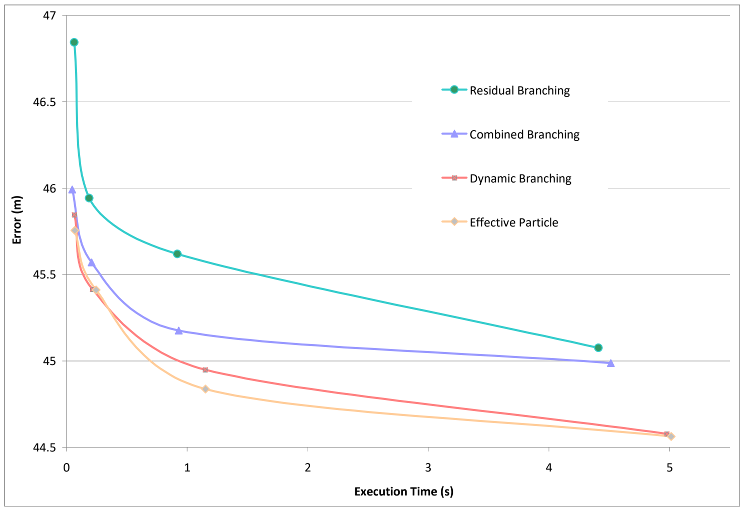

FIGURE 12. All Branching Particle Filters on Range Only Model

Both state-dependent branching methods outperform the combined branching method. The effective particle method produces the smallest error for given particles and seems to be the best method. However, the dynamic branching is faster than effective particle branching so we also consider Bootstrap Factor. Table 9 shows dynamic branching is best on a bootstrap performance per computation point of view. We can at least conclude that state-dependent branching is worthwhile.

	N	**Time (s)**	**Bootstrap Factor**
Residual Branching	2000	0.1883	36.7148
Combined Branching	500	0.0480	144.0292
Dynamic Branching	350	0.0412	167.8009
Effective Particle Branching	320	0.0465	148.6753

TABLE 9. Bootstrap Factor of Range Only with Error 46.0

To demonstrate the significant improvement of branching systems over resampled systems, we show typical *Test Model* and *Range Only Model* error versus time.

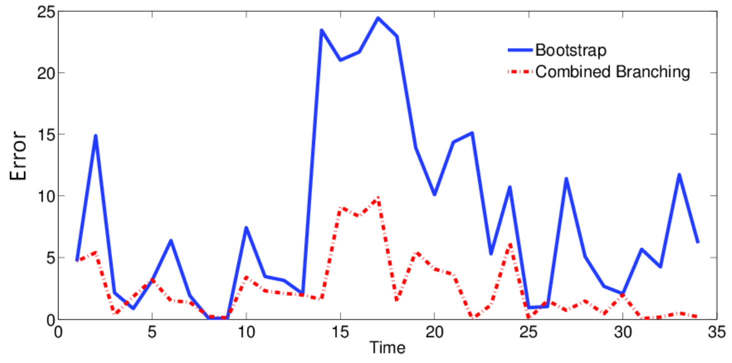

FIGURE 13. typical case in *Test Model* (m/s)

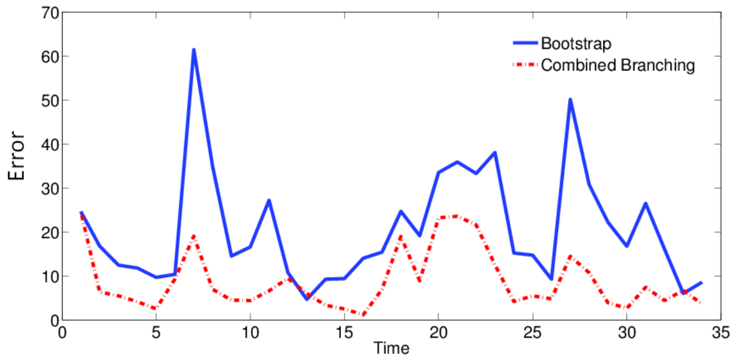

FIGURE 14. typical case in *Range Only Model* (m/s)

5. Conclusions

Based upon our experimental and theoretical results, we suggest the following:

(1) There are branching particle methods that do not suffer from particle swings.
(2) There are branching particle methods whose tracking performance and execution times can compare most favorably to the traditional resampled particle systems that have widespread appeal.
(3) Practioners should now consider the Combined, Dynamic and Effective Particle branching algorithms introduced herein.
(4) The Bootstrap Factor defined herein is a reasonable way to compare particle filtering methods.
(5) Branching particle filters also compare favorably on model selection problems and have the added advantage of using the unnormalized filter for ease of computing Bayes factor.

6. Acknowledgements

Partial funding in support of this work was provided by an NSERC discovery grant.
The author is indebted to Xingpu Wang for his help with the simulations.
The author is also indebted to the referees who provided most useful suggestions to improve the presentation of the results.

7. Appendix A: Discussion of Algorithmic Operations

Supplementary material related to this article can be found online.

8. Appendix B: Proof of Theorem 2

Supplementary material related to this article can be found online.

References

C. Andrieu, N. Chopin, A. Doucet, S. Rubenthaler (2014). Exact sampling using branching particle simulation <hal-00737040v3>.

C. Andrieu, A. Doucet and R. Holenstein (2010). Particle Markov chain Monte Carlo methods. *J.R. Statist. Soc. B* **72** 269-342.

A. Bain and D. Crisan (2008). *Fundamentals of Stochastic Filtering.* Springer, Berlin.

D. J. Ballantyne, H. Y. Chan, and M. A. Kouritzin (2000). A novel branching particle method for tracking. In *Signal and Data Processing of Small Targets* (O. E. Drummond, ed.) **SPIE 4048** 277-287.

O. Cappe, S. J. Godsill, and E. Moulines (2007). An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE* **95**, 899-924.

J. Carpenter, P. Clifford and P. Fearnhead (1999). An Improved Particle Filter for Nonlinear Problems. *IEE Proc. Radar Sonar Navigation* **146**, 2-7.

D. Crisan (2001). Particle filters - a theoretical perspective. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, **chapter 2**, p. 17-42. Springer, Berlin.

D. Crisan, P. Del Moral and T. Lyons (1999). Discrete Filtering Using Branching and Interacting Particle Systems *Markov Processes and Related Fields* **5**, 28 pages.

D. Crisan, J. Gaines and T. Lyons (1998). Convergence of a branching particle method to the solution of the Zakai equation. *SIAM J. Appl. Math.* **58**, 1568-1590.

D. Crisan and T. Lyons (1997). Nonlinear filtering and measure valued processes. *Prob. Theory Related Fields* **109**, 217-244.

P. Del Moral (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*, Springer, Berlin.

P. Del Moral, A. Doucet and A. Jasra (2012). On adaptive resampling strategies for sequential Monte Carlo methods. *Bernoulli* **18**, 252-278.

P. Del Moral, P.E. Jacob, A. Lee, L.M. Murray and G.W. Peters (2013). Feynman-Kac Particle Integration with Geometric Interacting Jumps. *Stochastic Analysis and Applications* **31(5)**:830-871.

P. Del Moral, M.A. Kouritzin and L. Miclo (2001). On a class of discrete generation interacting particle systems. *Electronic Journal of Probability* **6**: Paper No. 16, 26 p.

P. Del Moral and L. Miclo (2000). Branching and Interacting Particle Systems Approximations of Feynman-Kac Formulae with Applications to Non-Linear Filtering. Séminaire de Probabilités XXXIV, Ed. J. Azéma, M. Emery, M. Ledoux and M. Yor. *Lecture Notes in Mathematics*, Springer-Verlag Berlin, Vol. 1729, 1-145.

P.M. Djurić (1999). Monitoring and selection of dynamic models by Monte Carlo sampling. In *Proceedings of the IEEE Workshop on Higher Order Statistics*, **une 1999**, 191–194.

R. Douc, O. Cappé and E. Moulines (2005). Comparison of resampling schemes for particle filtering. In *Proceedings of the* 4^{th} *International Symposium on Image and Signal Processing and Analysis* 64-69.

A. Doucet, S.J. Godsill and C. Andrieu (2000). On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, **10**, 197-208.

A. Gandy and F. D-H. Lau (2016). The chopthin algorithm for resampling. *IEEE Transactions on Signal Processing* **64**, 4273-4281.

N. Gordon, D. Salmond and A. F. M. Smith (1993). Novel approach to nonlinear and non-Gaussian Bayesian state estimation. *Proc. Inst. Elect. Eng., F*, **140**, 107-113.

J.E. Handschin (1970). Monte Carlo Techniques for Prediction and Filtering of Non-Linear Stochastic Processes. *Automatica* **6**, 555-563.

J.E. Handschin and D.Q. Mayne (1969). Monte Carlo Techniques to Estimate the Conditional Expectation in Multi-stage Non-linear Filtering. *International Journal of Control* **9**, 547-559.

J. D. Hol, T. B. Schön and F. Gustafsson (2006). On resampling algorithms for particle fiters. In *Proceedings of the IEEE Nonlinear Statistical Signal Processing Workshop*, Cambridge, UK, pp. 79-82.

R.E. Kass and A.E. Raftery (1995). Bayes factors. *Journal of the American Statistical Association* **90**: 773-795.

G. Kitagawa (1996). Monte-Carlo filter and smoother for non-Gaussian nonlinear state space models. *J. Comput. Graph. Statist.* **1**, 1-25.

M. A. Kouritzin (2015). Microstructure Models with Short-Term Inertia and Stochastic Volatility. *Mathematical Problems in Engineering* **vol. 2015**, Article ID 323475, 17 pages.

M.A. Kouritzin and W. Sun (2005). Rates for Branching Particle Approximations of Continuous-Discrete Filters. *The Annals of Applied Probability* **15**, 2739-2772.

M.A. Kouritzin and Y. Zeng (2005). Bayesian Model Selection via Filtering for a Class of Micro-movement Models of Asset Price. *International Journal of Theoretical and Applied Finance* **8**, 97-122.

M.A. Kouritzin and Y. Zeng (2005). Weak convergence for a type of conditional expectation: application to the inference for a class of asset price models. *Nonlinear Analysis, Theory, Methods & Applications, Series A* **60** 231-239.

F. Le Gland and N. Oudjane (2006). A sequential particle algorithm that keeps the particle system alive. In *Stochastic Hybrid Systems: Theory and Safety Critical Applications, Henk Blom and John Lygeros, editors, Lecture Notes in Control and Information Sciences* **337**, pp. 351-389, Springer, Berlin.

M. Lin, R. Chen, and J. S. Liu (2013) Lookahead Strategies for Sequential Monte Carlo. *Statist. Sci.* **28**: 69-94.

J.S. Liu and R. Chen (1998). Sequential Monte-Carlo methods for dynamic systems. *Journal of the American Statistical Association* **93**: 1032-1044.

L.M. Murray, A. Lee and P.E. Jacob (2016). Parallel Resampling in the Particle Filter. *Journal of Computational and Graphical Statistics* **25**: 789-805.

C. Vergé, C. Dubarry, P. DelMoral and E. Moulines (2013). On parallel implementation of sequential Monte Carlo methods: the island particle model. *Statistics and Computing*, **23**: 91-107.