Efficient Sequential Monte Carlo Using Interpolation

Josh A. Taylor and Franz S. Hover

July 31, 2009

Abstract

A limitation common to all sequential Monte Carlo algorithms is the computational demand of accurately describing an arbitrary distribution, which may preclude real-time implementation for some systems. We propose using interpolation to construct a high accuracy approximation to the importance density. The surrogate density can then be efficiently evaluated in place of sampling the true importance density, allowing for the propagation of a large number of particles at reduced cost. Numerical examples are given demonstrating the utility of the approach.

1 Introduction

Sequential Monte Carlo (SMC), also commonly referred to as particle filtering, is a general solution to the recursive Bayesian estimation problem. The first practically applicable SMC algorithm appeared in [17] under the name bootstrap filter. Because the algorithm was both general and simple, many variations soon appeared, some of the most successful of which are assembled in [2].

SMC differs fundamentally from Kalman filtering [14] in how the distribution of the state is represented. Kalman filters propagate only the mean and covariance of the state because Gaussian distributions are uniquely determined by their first two moments. In contrast, SMC algorithms approximate arbitrary distributions with sets of weighted points called particles. A significant shortcoming is computational cost, however; SMC algorithms trade efficiency for generality. Often a large number of particles must be carried to obtain an accurate representation of the distribution.

Sampling from the importance density often comprises the most computationally expensive component of SMC. This can be cumbersome even in the bootstrap filter, which uses the prior as its importance density, if the prior is a complicated function to evaluate, e.g. a differential equation discretized over multiple time steps. When using more sophisticated importance densities such as local linearization approaches [11], sampling the density can become expensive even for systems with simple evolutions.

For many SMC algorithms, sampling the importance density can be separated into two steps: a deterministic function evaluation followed by the incorporation of a noise sample. It is usually the former which accounts for most of the computational load. In our approach, rather than evaluating the deterministic portion at each particle's position, we evaluate it at a small set of designed points (hereafter referred to as nodes), so as to construct an accurate approximation to the true function (called an interpolant). Particles can then be propagated by evaluating the interpolant and incorporating the appropriate noise sample. Because the interpolant is inexpensive to evaluate, a large number of particles can be efficiently propagated. Alternatively, by alleviating the burden of carrying many particles, surplus computational resources can be put towards higher resolution or more frequent sensing, both of which can be bottlenecks in SMC applications.

As an example, consider the bootstrap filter applied to a dynamical system whose state evolves according to a state space difference equation. One iteration of the filter involves evaluating the system equations at each particle's location, adding a process noise sample to the evolved particles, and then adjusting the weights according to the likelihoods. In our approach, the state equation is evaluated at special nodes to construct an interpolant, which is evaluated at each particle's location. Process noise is then incorporated into the location of each interpolated particle, and calculation of weights is carried out as in the original bootstrap filter.

More broadly, one may view our approach as *a parameterization of the deterministic portion of the importance density* which is used in place of the actual importance density. In this paper we employ Barycentric Lagrange polynomial interpolation [4, 9, 19], although any method is valid. We have chosen to use Barycentric Lagrange polynomial interpolation for its speed and stability, and because it lends itself to a formulation that is general over many filter characteristics. Other approaches, such as splines and trigonometric interpolation [34], should be considered if more appropriate to a particular problem.

One-dimensional schemes are extended to multiple dimensions by Cartesian products; this approach suffers the 'Curse of Dimensionality', which is that convergence slows exponentially with increasing dimension. Due to this fact, we expect that the new filter will provide computational gains in situations with a relatively low number of dimensions but for which the importance density is expensive to evaluate. Various multidimensional schemes exist for exploiting dimensional structure, e.g. sparse grids [3,33] and dimension adaptivity [16], the use of which may extend the applicability of the new filter to higher dimensions. However, there are many important problems of the scale developed here, including target tracking [18], control and system identification [1], and mobile robot localization [35].

The paper is organized as follows: Section 2 outlines interpolation and the general recursive Bayesian estimation framework, and discusses some basic SMC formulations. Section 3 details the new filter, and section 4 shows the results of numerical simulations on two benchmark systems.

2 Background

In this section, we provide background material needed for explaining how the interpolation filter is constructed, and for subsequently assessing its performance.

2.1 Interpolation

Interpolation [9] is the approximation of function values using evaluations of that function at other points (nodes) in the domain. For smooth functions, polynomial based interpolation methods are attractive for their fast rates of convergence. Here we use Barycentric Lagrange polynomial interpolation, some strengths of which were relatively recently explored [4, 19]. We prefer it over other methods for its numerical stability and, more significantly, its speed - using n nodes, the interpolant can be evaluated in $O(n)$ operations, and all quantities requiring greater than $O(n)$ computation can be precomputed. Furthermore, these precomputable quantities are function independent - regardless of how the importance density changes through time, the only necessary online computations pertaining to interpolation are function evaluations at the nodes and $O(n)$ evaluations of the interpolant. This is not the case with Newton polynomial interpolation [9,34], for which $O(n^2)$ calculations must be performed each time a new function is interpolated.

The primary criterion for polynomial interpolation schemes is that the nodes have the asymptotic density $1/\sqrt{1-x^2}$ as $n \to \infty$. This distribution prevents the weights of equation (1) below from differing in size exponentially with n [4]. Node schemes satisfying this density are typically the roots of orthogonal polynomials. Well-known sets of orthogonal polynomials are the Legendre and Chebyshev polynomials [29], and Legendre polynomials with Stieltjes polynomial extension [12], the associated numerical integration schemes of which are Gaussian [8, 31], Clenshaw-Curtis [7], and Gauss-Kronrod-Patterson quadratures [28], respectively. The latter two can be generated in nested sequences, enabling more economical constructions in certain multivariate schemes.

2.1.1 Barycentric Lagrange polynomial interpolation

For function f and node set $\chi = \{\chi^1, ..., \chi^n\}$, the one-dimensional barycentric interpolation formula of the second form is given by

$$I_n^f\left[\chi\right](x) = \frac{\sum_{i=1}^{n} \frac{b^i}{x - \chi^i} f(\chi^i)}{\sum_{i=1}^{n} \frac{b^i}{x - \chi^i}}, \qquad b^i = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\chi^i - \chi^j}. \tag{1}$$

The b^i are referred to as barycentric weights. $I_n^f\left[\chi\right]$ is itself a polynomial function, which is referred to as the interpolant.

2.1.2 Multivariate Interpolation

Univariate interpolation schemes can be extended in a general fashion to multiple dimensions via full grid tensor, or, for our purposes, Cartesian products. Abstractly, the d-dimensional, level-$\boldsymbol{n}$ interpolant may be written

$$I_{\boldsymbol{n}}^f[\boldsymbol{\chi}] = I_{n_1} \otimes \cdots \otimes I_{n_d}[\boldsymbol{\chi}]. \tag{2}$$

$\boldsymbol{n} = (n_1, ..., n_d)$ is a vector in the multivariate case; the interpolant may have different resolutions in different dimensions, and it should if a priori knowledge of the function suggests that not every dimension warrants the same computational effort. The corresponding multivariate barycentric formula of the second form for (2) is given by

$$I_{\boldsymbol{n}}^f\left[\boldsymbol{\chi}\right](\boldsymbol{x}) = \frac{\sum_{i_d=1}^{n_d} c_d^{i_d}(x_d) \times \cdots \times \sum_{i_1=1}^{n_1} c_1^{i_1}(x_1) f(\boldsymbol{\chi}^{i_1,...,i_d})}{\prod_{j=1}^{d} \sum_{i_j=1}^{n_j} c_j^{i_j}(x_j)}, \tag{3}$$

where

$$c_j^{i_j}(x_j) = \frac{b_j^{i_j}}{x_j - \chi_j^{i_j}}, \qquad j = 1, ..., d, \qquad i_j = 1, ..., n_j. \tag{4}$$

The variables in (3) and (4) are written component-wise: x_j is the j^{th} component of $\boldsymbol{x}$, and $\chi_j^{i_j}$ is the j^{th} component of the node with multi-index $\{i_1, ..., i_d\}$. Although it is possible to write (3) as (1) using a single summation, the component-wise expression is more informative. Details for efficient implementation can be found in [21].

The computational cost of evaluating $I_{\boldsymbol{n}}^f[\boldsymbol{\chi}]$ is $O(\prod_{i=1}^d n_i)$, which is the total number of nodes; generally, this quantity increases exponentially with dimension, and should be a consideration when deciding whether or not to use interpolation.

Node placements can sometimes be made more effective in higher dimensions by using different grid structures. Sparse grids [3, 15, 33] exploit weak dimensional coupling, and tend to perform better than full grids in slightly higher dimensions. Adaptive techniques such as [16] may further extend performance; however, the computational overhead of an adaptive algorithm may outweigh the gains from efficiently covering a space.

2.2 Recursive Bayesian Estimation

We are interested in the distribution of the state $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$, $k \in \mathbb{N}$ conditioned on all available measurements $\boldsymbol{y}_i \in \mathbb{R}^{n_y}$, $i = 1, ..., k$ of the discrete dynamical system

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{f}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k) \\ \boldsymbol{y}_k &= \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k), \end{aligned} \tag{5}$$

where $\boldsymbol{f}_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ and $\boldsymbol{h}_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_y}$ are respectively the state transition and measurement functions, and $\boldsymbol{w}_k \in \mathbb{R}^{n_w}$ and $\boldsymbol{v}_k \in \mathbb{R}^{n_v}$ are independent, identically distributed process and measurement noise vectors. The posterior distribution, $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$, can be expressed via Bayes rule as

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) &= \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})} \\ &= \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})}{\int p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})d\boldsymbol{x}_k}. \end{aligned} \tag{6}$$

The prior can be computed using $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ and the posterior at time $k-1$ through the relation

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})d\boldsymbol{x}_{k-1}. \tag{7}$$

$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ is sampled by evaluating the system equations, and can be written

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = \int \delta(\boldsymbol{x}_k - \boldsymbol{f}_k(\boldsymbol{x}_{k-1}, \boldsymbol{w}_k))p(\boldsymbol{w}_k)d\boldsymbol{w}_k. \tag{8}$$

This is the simulation component of SMC. Similarly, the conditional measurement distribution $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ is determined by the relationship

$$p(\boldsymbol{y}_k|\boldsymbol{x}_k) = \int \delta(\boldsymbol{y}_k - \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k))p(\boldsymbol{v}_k)d\boldsymbol{v}_k. \tag{9}$$

The Kalman filter is an analytical solution to this problem when $\boldsymbol{f}$ and $\boldsymbol{h}$ are linear and the state and noises have Gaussian distributions. In contrast, SMC algorithms carry the statistics of $\boldsymbol{x}_k$ in a set of weighted particles. Let $\{\boldsymbol{x}_k^i, \lambda_k^i\}, i = 1, ..., n$ be a set of particles $\boldsymbol{x}$ with weights λ at time k, such that $\sum_i \lambda_k^i = 1$ for all k. This comprises a discrete probability mass function, which can approximate $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ such that

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = \lim_{n\to\infty} \sum_{i=1}^{n} \lambda_k^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i). \tag{10}$$

2.3 Sequential Importance Sampling

At the foundation of SMC is the sequential importance sampling (SIS) algorithm [2,11]. Importance sampling refers to how the weights in equation (10) are determined. Ideally, samples would be drawn directly from $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ to build an approximation to itself; often, this is not possible. Suppose there is another similar distribution $q(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ with the same support as the posterior and which is easy to sample; this is referred to as an importance density (also proposal distribution). If the importance density is chosen such that $q(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{y}_{1:k})q(\boldsymbol{x}_{1:k-1}|\boldsymbol{y}_{1:k-1})$, the weights can then be computed using the update rule

$$\lambda_k^i = \lambda_{k-1}^i \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)}{q(\boldsymbol{x}_k^i|\boldsymbol{x}_{1:k-1}^i, \boldsymbol{y}_{1:k})}. \tag{11}$$

A convenient choice for $q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{y}_{1:k})$ is the prior, $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^i)$. When using this importance density, the weight update simplifies to $\lambda_k^i = \lambda_{k-1}^i p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)$.

Algorithm: SIS

- Sample $\boldsymbol{x}_k^i$ from $q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{y}_{1:k})$.
- Compute λ_k^i for each $\boldsymbol{x}_k^i$ using (11).

- Normalize $\lambda_k^i = \lambda_k^i / \sum_j \lambda_k^j$.

While simple, this implementation is somewhat naive. The prior is in many cases an inefficient choice for an importance density because it does not utilize the most recent observation. Furthermore, for any importance density, the same particles are used at each iteration, and often all weights except that of one particle will approach zero, a phenomenon which has been explained from a mathematical standpoint [23].

2.4 SMC algorithms

2.4.1 Sampling Importance Resampling

Resampling is a straightforward solution to the weight degeneracy problem of the SIS filter [6, 17, 24]. Resampling eliminates particles with low weights and replaces particles with large weights with multiple particles. Consider the quantity $N_{eff} = 1/\sum_{i=1}^{n}(\lambda^i)^2$; this is an approximate measure of how balanced the weights are. The algorithm, called sampling importance resampling (SIR), is identical to that of the SIS with an additional step: if $N_{eff} < N_{thr}$, resample the particles.

A common pitfall of SIR is for the distribution to collapse when process noise is small, leading to a set of identical particles; this is known as sample impoverishment.

2.4.2 Optimal and suboptimal importance densities

The optimal importance density, i.e. that which minimizes the variance of each importance weight λ_k^i conditioned on $\boldsymbol{x}_{1:k-1}^i$ and $\boldsymbol{y}_{1:k}$, is $q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}, \boldsymbol{y}_{1:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$. The associated weight update is $\lambda_k^i = \lambda_{k-1}^i p(\boldsymbol{y}_k|\boldsymbol{x}_{k-1}^i)$. This density can be difficult to sample from, and furthermore the weight update usually has no analytical form. An important case for which the optimal importance density is easily implemented is when the observation equation is linear and all noises are Gaussian. When noises are Gaussian but the observation equation is nonlinear, it sometimes works well to linearize the observation equation and construct a suboptimal approximation to the optimal density; this is known as local linearization [11]. Consider the system

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{f}_k(\boldsymbol{x}_{k-1}) + \boldsymbol{w}_k, \qquad \boldsymbol{w}_k \sim N(\boldsymbol{0}, \boldsymbol{Q}_k), \\ \boldsymbol{y}_k &= \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{v}_k, \qquad \boldsymbol{v}_k \sim N(\boldsymbol{0}, \boldsymbol{R}_k). \end{aligned} \tag{12}$$

Let $\boldsymbol{H}_k = \frac{d\boldsymbol{h}_k(\boldsymbol{x}_k)}{d\boldsymbol{x}_k}\Big|_{\boldsymbol{x}_k=\boldsymbol{f}_k(\boldsymbol{x}_{k-1})}$, and define

$$\begin{aligned} \boldsymbol{\Sigma}_k^{-1} &= \boldsymbol{Q}_k^{-1} + \boldsymbol{H}_k^T \boldsymbol{R}_k^{-1} \boldsymbol{H}_k & (13) \\ \boldsymbol{m}_k &= \boldsymbol{\Sigma}_k \left(\boldsymbol{Q}_k^{-1} \boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{H}_k' \boldsymbol{R}_k^{-1} \times \right. \\ & \quad (\boldsymbol{y}_k - \boldsymbol{h}_k(\boldsymbol{f}_k(\boldsymbol{x}_{k-1})) + \boldsymbol{H}_k \boldsymbol{f}_k(\boldsymbol{x}_{k-1}))) \, . & (14) \end{aligned}$$

The distribution $N(\boldsymbol{m}_k, \boldsymbol{\Sigma}_k)$ is an approximation to the optimal importance density. In the case that $\boldsymbol{h}_k$ is linear, the above expressions simplify to those for the optimal importance density.

2.4.3 Regularization

Regularization [2, 25] addresses sample impoverishment by perturbing each resampled particle's position with samples from either an Epanechnikov (optimally) or Gaussian (simply) distribution. We use the acronym RPF to denote the regularized particle filter in our numerical examples below.

Many other improvements to the basic SMC formulation exist [10,24,32], e.g. Markov Chain Monte Carlo, Rao-Blackwellisation, and numerous techniques for designing importance densities. We consider only those discussed above for the purpose of demonstrating our approach, but note that most others are compatible as well.

3 The Interpolation Particle Filter

3.1 Main Filter Algorithms

This section discusses the algorithm for the interpolation particle filter, which we refer to as IPF. As stated in the introduction, the main insight is the division of sampling from the importance density into a deterministic function evaluation followed by the incorporation of a noise sample. By evaluating the deterministic portion at interpolation nodes, a functional approximation (the interpolant) is constructed, which can then be easily evaluated to allow the propagation of a large number of particles at a reduced cost. We apply the interpolation procedure of Section 2.1 in building the interpolant.

We assume that sampling from the importance density can be expressed as the composition of functions, such that sampling at time k can be written $\boldsymbol{x}_k = \boldsymbol{u}_k(\boldsymbol{g}_k(\boldsymbol{x}_{k-1}), \boldsymbol{z}_k)$. $\boldsymbol{g}_k$ is entirely deterministic and is subject to interpolation. $\boldsymbol{z}_k$ is a noise sample, and $\boldsymbol{u}_k$ is a function which incorporates $\boldsymbol{z}_k$ into a particle's position at time k. Note that $\boldsymbol{g}_k$ may also be a function of

$\boldsymbol{y}_{1:k}$; because there is only one measurement per time step, there is no need to interpolate over the space of measurements, and so we proceed as though $\boldsymbol{y}_{1:k}$ is implicit in the structure of $\boldsymbol{g}_k$.

The essence of our approach is as follows. A given number of nodes are scaled so as to cover all particles at time $k-1$. The deterministic portion of the importance density $\boldsymbol{g}_k$ is then evaluated at these nodes. Following the procedure of Section 2.1, an interpolant is constructed and evaluated at particle locations at time $k-1$. The appropriate noise sample $\boldsymbol{z}_k$ is then incorporated into the position of the interpolated particles via the function $\boldsymbol{u}_k$. We propose two IPF's, one in which the interpolant is constructed every iteration and is scaled to contain all particles, and another in which a single interpolant is constructed over a fixed region and reused at each iteration.

In our notation, $\boldsymbol{\chi}^j$, $j = 1, ..., n_N$ refers exclusively to nodes and $\boldsymbol{x}^i$, $i = 1, ..., n_P$ to particles. Underscored symbols correspond to non-dimensional quantities in the hypercube $[-1, 1]^d$, where d is the dimension of $\boldsymbol{x}_k$, and those not underscored refer to quantities in the system's state space. The absence of a superscript implies the entire set of nodes or particles. Fig. 1 gives a visualization of our first IPF algorithm, written as an algorithm below.

Algorithm: IPF$_a$
Precompute

- Compute nodes $\{\underline{\boldsymbol{\chi}}^j \in [-1, 1]^d,\ j = 1, ..., n_N\}$ and barycentric weights for the orthogonal polynomials used.

Online

- Compute the width of the particles at time $k-1$:

$$\boldsymbol{S}_{k-1} =$$

$$a \cdot diag\left[\max\left\{\max_{i=1,...,n_P} \boldsymbol{x}^i_{k-1} - \min_{i=1,...,n_P} \boldsymbol{x}^i_{k-1}, \boldsymbol{b}\right\}\right],$$

  where both maximums and the minimum are taken independently for each dimension of the state. Note that $\boldsymbol{S}_{k-1}$ is a $d \times d$ matrix. Guidelines on how the parameters a and $\boldsymbol{b}$ should be chosen are given in Section 3.2.

- Compute the center of the particles at time $k-1$:

$$\boldsymbol{\mu}_{k-1} = \frac{1}{2}\left(\max_{i=1,...,n_P} \boldsymbol{x}^i_{k-1} + \min_{i=1,...,n_P} \boldsymbol{x}^i_{k-1}\right) + \boldsymbol{c},$$

again with the maximum and minimum taken independently for each dimension, so that $\boldsymbol{\mu}_{k-1}$ is a vector the same size as the state at time $k-1$. Again see Section 3.2 for guidelines on choosing $\boldsymbol{c}$.

- Map the particles at time $k-1$ to the hypercube $[-1,1]^d$:

$$\underline{\boldsymbol{x}}_{k-1}^i = 2\boldsymbol{S}_{k-1}^{-1}(\boldsymbol{x}_{k-1}^i - \boldsymbol{\mu}_{k-1}).$$

- Transform the precomputed nodes so that the interpolant's domain encompasses all particles at time $k-1$:

$$\boldsymbol{\chi}_{pre}^j = \frac{1}{2}\boldsymbol{S}_{k-1}\underline{\boldsymbol{\chi}}^j + \boldsymbol{\mu}_{k-1}.$$

- Evaluate the deterministic portion of the importance density at transformed node locations:

$$\boldsymbol{\chi}_{post}^j = \boldsymbol{g}_k(\boldsymbol{\chi}_{pre}^j).$$

- Evaluate the interpolant $I_{n_N}[\boldsymbol{\chi}_{post}]$ at particle locations at time $k-1$. Evaluate the interpolant at each particle's location:

$$\boldsymbol{x}_{k*}^i = I_{n_N}[\boldsymbol{\chi}_{post}](\underline{\boldsymbol{x}}_{k-1}^i).$$

  I_{n_N} is defined in Section 2.1.

- Incorporate the appropriate noise sample into interpolated particles with the function $\boldsymbol{u}_k$:

$$\boldsymbol{x}_k^i = \boldsymbol{u}_k(\boldsymbol{x}_{k*}^i, \boldsymbol{z}_k).$$

- Perform other SMC steps, e.g. compute weights and resample the particles. Increment k and return to the first online step.

In the second implementation of the IPF, an interpolant is constructed over a fixed region of the state space and reused at each iteration. This formulation is applicable only to time-invariant systems with bounded state spaces or trajectories that are expected to remain within a known region. Strong confidence in the prescribed region is necessary, as any particle that departs it will likely cause divergence. Note that this formulation requires no online evaluation of the deterministic portion of the importance density, only interpolation. We refer to this algorithm as the IPF$_b$.

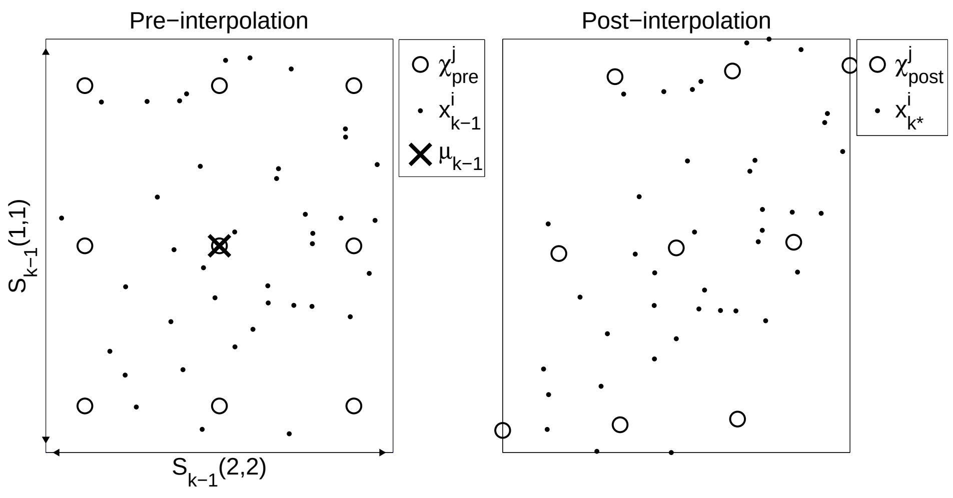

Figure 1: Visualization of IPF interpolation procedure in two dimensions with Legendre interpolation nodes and $a = 1.1$.

Algorithm: IPF$_b$

Precompute

- Precompute nodes $\{\underline{\boldsymbol{\chi}}^j \in [-1,1]^d, j = 1, ..., n_N\}$ and barycentric weights.
- Choose the center $\boldsymbol{\mu}$ and width $\boldsymbol{S}$ of the interpolant's domain.
- Map the nodes to the prescribed region:

$$\boldsymbol{\chi}^j_{pre} = \frac{1}{2}\boldsymbol{S}\underline{\boldsymbol{\chi}}^j + \boldsymbol{\mu}.$$

- Evaluate the deterministic part of the importance density at the nodes:

$$\boldsymbol{\chi}^j_{post} = \boldsymbol{g}(\boldsymbol{\chi}^j_{pre}).$$

Online

- Map the particles at time $k-1$ to $[-1,1]^d$:

$$\underline{\boldsymbol{x}}^i_{k-1} = 2\boldsymbol{S}^{-1}(\boldsymbol{x}^i_{k-1} - \boldsymbol{\mu}).$$

- Interpolate the particles:

$$\boldsymbol{x}^i_{k*} = I_{n_N}[\boldsymbol{\chi}_{post}](\underline{\boldsymbol{x}}^i_{k-1}).$$

- Incorporate the appropriate noise sample
$$\boldsymbol{x}_k^i = \boldsymbol{u}(\boldsymbol{x}_{k*}^i, \boldsymbol{z}_k).$$
- Perform other SMC steps. Increment k and return to the first online step.

3.2 Numerical issues

The parameters a, $\boldsymbol{b}$, and $\boldsymbol{c}$ are necessary for stability of the IPF$_a$ using barycentric Lagrange polynomial interpolation. a is a scalar constant which prevents scalings that result in points being interpolated on the boundary of the interpolant's domain. For Legendre or Stieltjes orthogonal polynomials, this can lead to compromised numerical stability, and for Chebyshev, division by zero. In our numerical examples, we set $a = 1.01$; any value slightly above one is sufficient. Although it is possible for a particle in the interior of the interpolant's domain to coincide with a node, this is highly unlikely, and it is not a scenario we prepare for.[1]

The parameters $\boldsymbol{b}$ and $\boldsymbol{c}$ prevent division by zero in the event of extreme sample impoverishment. Because all particles are coincident in this case, the interpolation region shrinks to a point and the $\boldsymbol{S}$ matrix, which must be inverted, becomes singular. $\boldsymbol{b}$ fixes a minimum size for the interpolation region and hence forces the $\boldsymbol{S}$ matrix (which is diagonal) to be invertible. $\boldsymbol{c}$ addresses a second issue arising from sample impoverishment. Even if the size of the interpolation region is greater than zero, all particles will be situated in its exact center. Because the center of the interpolation region is a node in most schemes, this leads to the same instability discussed in the previous paragraph. Offsetting the location of the region by $\boldsymbol{c}$ prevents this failure. In the implementations shown here we used $\boldsymbol{b}_i = \boldsymbol{c}_i = 0.01$, $i = 1, ..., d$. Note that $\boldsymbol{b}$ and $\boldsymbol{c}$, unlike a, affect the interpolation region additively, and so the size of the region should be a factor in choosing these parameters.

Newton polynomial interpolation [9,29] handles interpolation on nodes in a more natural fashion, and would eliminate the need for the parameters a

[1]An obvious solution is to recognize that a true evaluation is available when interpolating on a node, and for many applications, this is sensible. Filtering often demands real-time implementation, and the logical statements necessary to handle this scenario add computational burden, particularly in higher dimensions. Furthermore, the loss in accuracy from interpolating over a slightly larger domain is usually negligible.

and $\boldsymbol{c}$. However, since a and $\boldsymbol{c}$ impose no significant computational burden, and because Newton interpolation introduces further numerical instabilities, it is the authors' opinion that barycentric Lagrange polynomial interpolation is a superior choice for this setting.

3.3 Implementation

The IPF depends heavily on the particular importance density being used; the algorithms of the previous section are intended to be general, and hence omit details pertaining to specific implementations. In this section, some examples are described with the intention giving intuition on how to implement interpolation for a variety of importance densities.

First consider the bootstrap filter [17]. The importance density is the prior, which is composed of an evaluation of the state transition followed by the incorporation of a process noise sample. Interpolation is used on the state transition, and process noise is included afterwards as in the conventional bootstrap filter. Filters using auxiliary variables [2, 30] pose no additional complication because the prior is essentially being used twice, and hence can be interpolated twice. As noted previously, features unrelated to the importance density such as resampling and regularization [2, 25] have no bearing on the interpolation component.

Now consider the (approximate) optimal importance density of Section 2.4.2 [11]. For the IPF$_a$, both the mean $\boldsymbol{m}_k$ and the square root of the covariance $\sqrt{\boldsymbol{\Sigma}_k}$ can be interpolated. Note that because the covariance is a function of the state, interpolating it does not increase the dimension of the interpolant. For a nonlinear observation equation, this will be a time-varying importance density due to the linearization about the state, and thus the IPF$_b$ cannot be used in the same fashion as the IPF$_a$. However, if the state transition is time-invariant, an interpolant can then be constructed just for it and used in (14), in place of the true state transition. Note that $\boldsymbol{\Sigma}_k^{-1}$ or $\boldsymbol{\Sigma}_k$ could be interpolated in place of $\sqrt{\boldsymbol{\Sigma}_k}$, leaving the matrix inversion or the matrix inversion and square root for after the interpolation. These can be expensive operations, and by lumping them into the interpolant greater efficiency is achieved.

In the formulation of Section 2.4.2, although both the mean and covariance of each particle are being interpolated, they are both solely functions of particle position. In the extended Kalman and unscented particle filters [36], each particle carries a position and a covariance. Again both the mean and covariance can be interpolated, but now both quantities are functions of the mean and covariance at the previous time step as well. Because the deterministic portion of the importance density is a function of both these quantities, the interpolant must be constructed over the $d + d(d+1)/2$ distinct states of the mean and covariance - while feasible, applying interpolation to these importance densities is somewhat impractical for systems with more than one or two states.

3.4 Convergence

The IPF approximates conventional SMC algorithms in the sense that interpolating a function is an approximation to evaluating the function. Standard interpolation convergence results including rates can be found in [9,13]. Typically, assuming the number of nodes is increased uniformly in all dimensions, the errors of polynomial interpolation of smooth functions decay exponentially as

$$\max_{x \ni [-1,1]} |f(x) - I_n(x)| \leq CK^{-n/d} \tag{15}$$

for some constants C and $K > 1$. The factor of d reflects the Curse of Dimensionality, which is for our purposes is the exponential increase with dimension of the number of points necessary to effectively sample a function over a space. This fact reduces applicability in higher dimensions; we note however that unlike similar approaches in [5] and [20], we are interpolating solely over the hidden state and not the noise sample, resulting in a lower dimensional approximation (e.g. half the dimension if there is independent noise added to each state).

In our application, we are replacing $\boldsymbol{g}_k$ with the interpolant $I_{n_N}[\boldsymbol{\chi}_{post}]$. As n_N approaches infinity, $I_{n_N}[\boldsymbol{\chi}_{post}]$ approaches $\boldsymbol{g}_k$ according to (15), and so drawing samples by evaluating $\boldsymbol{u}_k(I_{n_N}[\boldsymbol{\chi}_{post}](\cdot), \cdot)$ is identical to drawing samples from the original importance density in the limit of the number of nodes.

Whether or not the estimated posterior asymptotically approaches the true posterior is implementation dependent, as we can explain by dissecting (11), the weight update. It is known that any importance density q may be used, even a constant one [24], so long as it has the same support as the true posterior, and hence an interpolated importance density is just as much a valid importance density as that which it approximates. However, the exact prior $p(x_k|x_{k-1})$ must be used in (11) for the weight update to be correct. In some cases, the full importance density is expensive relative to the prior, and it is then sensible to interpolate the importance density and use the actual prior; this is done in example 4.1. Conversely, if the prior does account for most of the computational load, one would want to interpolate both the prior and the full importance density. If the interpolated prior is a good approximation to the true prior, using (11) with the substituted approximation may still result in a strong filter. Furthermore, from a real-world perspective, the prior used by the algorithm is almost never an exact model of the actual system, and so there is little qualitative difference in using an accurate approximation.

4 Numerical Examples

Two examples are considered here: the scalar, nonlinear system of [17] and the Kraichnan-Orszag system [27, 37]. The first example demonstrates the performance of the algorithm on a simple system with a complex importance density, while the second example shows performance in on a system with an expensive prior.

Conventional and interpolated versions of regularized and SIR particle filters are compared. An 'I' is appended to each acronym when interpolation has been used. The regularized filters sample from the prior $p(x_k|x_{k-1})$, and the SIR filters use either the optimal or a linearization of the optimal importance density. $N_{thr} = n/3$ was used as a resampling threshold, and perturbations for regularization were sampled suboptimally from the appropriate Gaussian distributions [25].

In each example, small population filters are shown to provide comparisons on the basis of system evaluations and large population filters to demonstrate the consistency of the interpolated filters with conventional filters. The small population filters propagate the same number of particles as nodes used by the interpolated filters (n_N). The large population filters propagate n_P particles, the number of particles interpolated by the interpolated filters. Interpolated filters are written I{PF}-n_N/n_P, and the conventional filters are written {PF}-n, where n is the number of particles propagated.

For each example, the mean of the mean absolute error over time is given:

$$\text{Error} = \frac{1}{T}\sum_{k=1}^{T}\frac{1}{M}\sum_{i=1}^{M}|\overline{x}_k^i - x_k^i|, \tag{16}$$

where $\overline{x}_k^i$ is the filter's estimate of the conditional mean for the i^{th} trial, $\mathbb{E}[x_k^i|y_{1:k}^i]$, and x_k^i is the true state for that trial. The mean over all trials of the time taken for the online portion of each filter is given as a measure of computational efficiency. $M = 100$ trials were performed for each example. Times reported are for a standard desktop computer running Matlab.

4.1 Scalar nonlinear system

Consider the following system studied in [17]:

$$\begin{aligned} x_k &= \frac{x_{k-1}}{2} + \frac{25x_{k-1}}{1+x_{k-1}^2} + 8\cos(1.2(k-1)) + w_k, \\ y_k &= \frac{x_k^2}{20} + r_k. \end{aligned} \tag{17}$$

w_k and r_k are zero-mean Gaussian noises with respective variances 10 and 1. The initial state of the true system is $x_0 = 0.1$, and the initial state used by the filters was sampled from $N(x_0, 2)$. This system is highly nonlinear, and additionally exhibits bimodal behavior if the measurement is greater than zero.

In this example, $n_N = 15$ Legendre nodes were used with $n_P = 100$ interpolated particles. Table 1 summarizes the performances of the filters.

Table 1: Scalar nonlinear system errors

Filter	x	Time (s)
RPF-15	4.08	0.04
RPF-100	2.96	0.32
IRPF-15/100	3.15	0.74
SIR-15	3.66	0.22
SIR-100	2.83	1.46
ISIR-15/100	2.85	1.12

In the regularized filters, interpolation is only used for propagating the state, and because the system equation is already inexpensive to evaluate, no improvement is seen in efficiency. However, the locally linearized optimal importance density is considerably more expensive to sample from than the prior. By applying interpolation to the mean and the square root of the covariance, the computational work is reduced by a factor of approximately 1/4. Although this is a modest improvement, it is nonetheless significant when considering the nature of this example; because it is inexpensive to evaluate, we expect that at least this level of improvement would be seen for most other systems.

4.2 Kraichnan-Orszag System

The equations for the Kraichnan-Orszag system [27,37] are given by:

$$\begin{aligned}
\dot{x}_1 &= x_2 x_3 + w_1 \\
\dot{x}_2 &= x_1 x_3 + w_2 \\
\dot{x}_3 &= -2x_1 x_2 + w_3, \\
\boldsymbol{y} &= \boldsymbol{x} + \boldsymbol{r}.
\end{aligned} \tag{18}$$

Initial conditions of the true system are $\boldsymbol{x}(0) = [0, 1, 2]'$, $\boldsymbol{P}_0 = \boldsymbol{I}_3$, and Gaussian noises $\boldsymbol{w}$ and $\boldsymbol{r}$ both have zero-mean and covariance $\boldsymbol{I}_3$. Filter initial conditions were sampled from $N(\boldsymbol{x}(0), \boldsymbol{I}_3)$. Fig. 2 shows trajectories through time of each state for one realization. An Euler step was used with a time step of $dt = 0.001$ and measurements taken at every $100dt$. We note that there are more efficient ways to design a standard SMC algorithm for this system; however, our intention is to demonstrate the gains yielded by interpolation on a system with a prior that is very expensive to sample from, and to show that although now the weight update (11) is approximate, the interpolated filter still performs quite well.

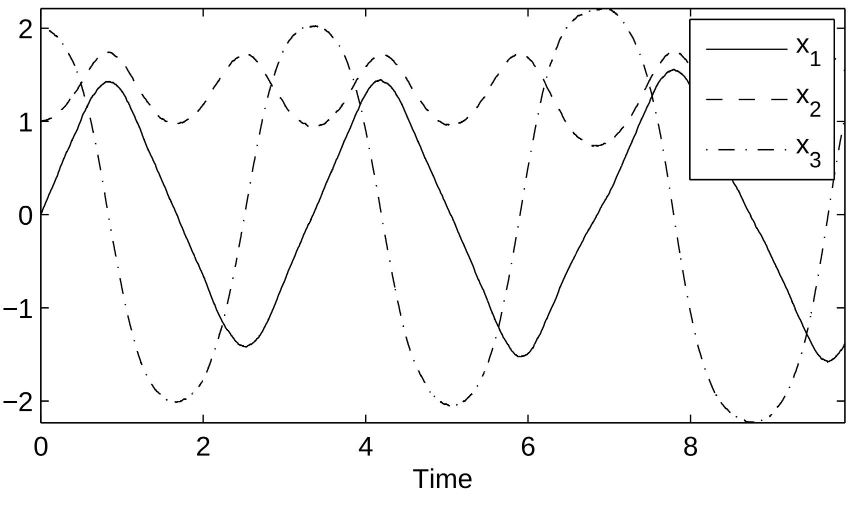

Figure 2: A realization of the Kraichnan-Orszag system.

Because this system is a stochastic ordinary differential equation [22,26] discretized over multiple time steps between measurements, process noise becomes an issue. Rigorous application of the RPF is straightforward: the importance density is the prior, so simply adding scaled samples of the process noise to the simulated trajectories at each time step between measurements is sufficient. However, this importance density cannot be straightforwardly separated into a deterministic and stochastic portion, and thus interpolation cannot be rigorously applied. Instead we incorporate process noise heuristically by adding to each particle post interpolation a zero-mean Gaussian noise sample of variance $dt^2 \cdot n_s^2 \cdot \boldsymbol{Q}$, where n_s is the number of time steps between measurements. Both types of IPF are applied, and are denoted $\text{IRPF}_a - 8/100$ and $\text{IRPF}_b - 64/100$. The former uses eight interpolation nodes at each iteration, and latter precomputes a 64-node interpolant over the fixed region $[-4, 4] \times [0, 4] \times [-4, 4]$.

Implementing SIR with the optimal importance density poses additional complications; one could augment the state vector with the process noise at each time step, but this would drive the dimension of the system into the hundreds. Instead we apply the above heuristic, using $\boldsymbol{Q}' = dt^2 \cdot n_s^2 \cdot \boldsymbol{Q}$ in place of $\boldsymbol{Q}$ in (13) and (14). This also makes the importance density separable, allowing interpolation to be applied without any further heuristic. An SIR-8, SIR-100, and an $\text{ISIR}_a - 8/100$ were compared, with interpolation being applied to the mean and the square root of the covariance in the $\text{ISIR}_a - 8/100$.

Table 2 shows the mean over time of the mean absolute errors for this example. The two IRPF's perform comparably with the RPF-100, but with drastic improvements in time, particularly for the $\text{IRPF}_b - 64/100$. The optimal importance density SIR-100 and $\text{ISIR}_a - 8/100$ filters perform nearly identically, again with a near order of magnitude improvement in time for the interpolated filter.

Table 2: Kraichnan-Orszag system errors

Filter	x_1	x_2	x_3	time (s)
RPF-8	0.68	0.47	0.97	1.84
RPF-100	0.32	0.22	0.46	23.01
IRPF_a-8/100	0.34	0.28	0.44	2.59
IRPF_b-64/100	0.35	0.29	0.46	1.40
SIR-8	0.42	0.35	0.58	1.54
SIR-100	0.22	0.20	0.28	19.28
ISIR_a-8/100	0.21	0.20	0.27	3.80

5 Conclusion

We have presented an approach in which interpolation is used to construct an inexpensive approximation to the importance density, which can be used in place of the true importance density, to propagate a large number of particles at a reduced cost. The method yields significant improvements in efficiency compared with several standard SMC algorithms, as demonstrated in two examples. The computational gains are most appealing for systems with expensive importance densities, but with few enough states that quadratures are effective. These include nonlinear differential equations with infrequent sampling.

References

[1] C. ANDRIEU, A. DOUCET, S.S. SINGH, and V.B. TADIC. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, 92(3):423–438, Mar 2004.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[3] V. Barthelmann, E. Novak, and K. Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12(4):273–288, March 2000.

[4] J. Berrut and L. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004.

[5] E. Blviken and G. Storvik. Deterministic and stochastic particle filters in state space models. *Sequential Monte Carlo in Practice*, 2001.

[6] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *Radar, Sonar and Navigation, IEE Proceedings -*, 146(1):2–7, Feb 1999.

[7] C. W. Clenshaw and A. R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2(1):197–205, 1960.

[8] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration.* Academic Press, New York, 1975.

[9] Philip J. Davis. *Interpolation and approximation.* Dover, New York, 1975.

[10] Arnaud Doucet, Nando De Freitas, and Neil Gordon, editors. *Sequential Monte Carlo methods in practice.* Springer, New York, 2001.

[11] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[12] Sven Ehrich and Giuseppe Mastroianni. Stieltjes polynomials and Lagrange interpolation. *Math. Comput.*, 66(217):311–331, 1997.

[13] Bengt Fornberg. *A Practical Guide to Pseudospectral Methods.* Cambridge Monographs on Applied and Computational Mathematics. 1996.

[14] A. Gelb. *Applied optimal estimation.* MIT Press, Cambridge, MA, 1974.

[15] T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18:209–232, 1998.

[16] T. Gerstner and M. Griebel. Dimension-adaptive tensor-product quadrature. *Computing*, 71(1):65–87, August 2003.

[17] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, Apr 1993.

[18] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *Signal Processing, IEEE Transactions on*, 50(2):425–437, Feb 2002.

[19] Nicholas J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA J Numer Anal*, 24(4):547–556, 2004.

[20] Genshiro Kitagawa. Non-gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association*, 82(400):1032–1041, 1987.

[21] Andreas Klimke. *Uncertainty modeling using fuzzy arithmetic and sparse grids.* PhD thesis, Universität Stuttgart, Shaker Verlag, Aachen, 2006.

[22] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations (Stochastic Modelling and Applied Probability).* Springer, November 2000.

[23] Augustine Kong, Jun S. Liu, and Wing Hung Wong. Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

[24] Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[25] C. Musso, N. Oudjane, and F. Legland. Improving regularised particle filters. In A. Doucet, N. de Freitas, and N. J. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. New York: Springer, 2001.

[26] Bernt Oksendal. *Stochastic differential equations (3rd ed.): an introduction with applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1992.

[27] Steven A. Orszag and L. R. Bissonnette. Dynamical properties of truncated Wiener-Hermite expansions. *Physics of Fluids*, 10(12):2603–2613, 1967.

[28] T. N. L. Patterson. The optimum addition of points to quadrature formulae. *Mathematics of Computation*, 22(3):847–856, 1968.

[29] George M. Phillips. *Interpolation and approximation by polynomials*. Springer, New York, 2003.

[30] Michael K Pitt and Neil Shephard. Filtering via simulation: auxiliary particle filters. Economics Papers 1997-W13, Economics Group, Nuffield College, University of Oxford.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[32] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon, editors. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Publishers, New York, 2004.

[33] S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR*, 148:1042–1043, 1963. Russian, Engl. Transl.: Soviet Math. Dokl. 4:240–243, 1963.

[34] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer, third edition, 2002.

[35] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dallaert. Robust monte carlo localization for mobile robots. *Artif. Intell.*, 128(1-2):99–141, 2001.

[36] Rudolph van der Merwe, Nando de Freitas, Arnaud Doucet, and Eric Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems 13*, Nov 2001.

[37] Xiaoliang Wan and George Em Karniadakis. An adaptive multi-element generalized polynomial chaos method for stochastic differential equations. *J. Comput. Phys.*, 209(2):617–642, 2005.