Title Adaptive Sparse-grid Gauss-Hermite Filter

Accepted for publication Journal of Computational and Applied Mathematics

Authors Abhinoy Kumar Sing, Rahul Radhakrishnan, Shovan Bhaumik and Paresh Date

Institutional affiliations of authors

Abhinoy Kumar Singh,

Department of Electrical Engineering,

Indian Institute of Technology Patna, Bihar-801103, India

E-mail: abhinoy@iitp.ac.in

Rahul Radhakrishnan,

Department of Electrical Engineering,

Indian Institute of Technology Patna, Bihar-801103, India

E-mail: rahul.pee13@iitp.ac.in

Dr. Shovan Bhaumik,

Department of Electrical Engineering,

Indian Institute of Technology Patna, Bihar-801103, India

Phone +91 612 255 2049, Fax: +91 612 2277383

E-mail: shovan.bhaumik@iitp.ac.in

Dr. Paresh Date,

Department of Mathematics,

College of Engineering, Design and Physical Sciences,

Brunel University, London Uxbridge, UB8 3PH, United Kingdom

Phone: +44 1895 265613, Fax: +44 1895 269732

Email: Paresh.Date@brunel.ac.uk

Abstract

In this paper, a new nonlinear filter based on sparse-grid quadrature method has been proposed. The proposed filter is named as adaptive sparse-grid Gauss-Hermite filter (ASGHF). Ordinary sparse-grid technique treats all the dimensions equally, whereas the ASGHF assigns a fewer number of points along the dimensions with lower nonlinearity. It uses adaptive tensor product to construct multidimensional points until a predefined error tolerance level is reached. The performance of the proposed filter is illustrated with two nonlinear filtering problems. Simulation results demonstrate that the new algorithm achieves a similar accuracy as compared to sparse-grid Gauss-Hermite filter (SGHF) and Gauss-Hermite filter (GHF) with a considerable reduction in computational load. Further, in the conventional GHF and SGHF, any increase in the accuracy level may result in an unacceptably high increase in the computational burden. However, in ASGHF, a little increase in estimation accuracy is possible with a limited increase in computational burden by varying the error tolerance level and the error weighting parameter. This enables the online estimator to operate near full efficiency with a predefined computational budget.

Keywords - Nonlinear filtering, Gauss-Hermite quadrature rule, Product rule, Smolyak rule, Complexity reduction, Adaptive sparse-grid.

1 Introduction

In this article, we address the state estimation problem of a discrete nonlinear dynamic system with additive noise. The process and measurement model of the nonlinear system can respectively be defined as

$$\mathbf{x}_k = \phi(\mathbf{x}_{k-1}) + w_k \tag{1}$$

and

$$y_k = \gamma(\mathbf{x}_k) + v_k, \tag{2}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ represents the unknown states of the system, $y_k \in \mathbb{R}^p$ denotes the measurement at any discrete time k. $\phi(\cdot)$ and $\gamma(\cdot)$ are known nonlinear functions. The process and measurement noises are represented by $w_k \in \mathbb{R}^n$ and $v_k \in \mathbb{R}^p$ respectively. They are assumed to be uncorrelated and normally distributed with zero mean and covariance, Q and R respectively.

Bayesian estimation framework is a widely employed method for addressing a filtering problem. In this framework, by using the measurement likelihood and the predicted motion of the unknown states, the posterior probability density functions (pdf) are computed [1].

During filtering of nonlinear systems, a set of intractable integrals appear and hence no optimal solution exists. In a widely accepted approach, the conditional pdfs are approximated as Gaussian and characterized with mean and covariance. Under this approach, a variety of filters like extended Kalman filter (EKF) [1], unscented Kalman filter (UKF) [2] and its extensions [3,4], cubature Kalman filter (CKF) [5] and its extension [6], central difference filter (CDF) [7] *etc.* are proposed. In a different approach, particle filter (PF) [8] is developed which approximates the true probability density function (pdf) with the help of particles and their assigned weights. Although the particle filter has high accuracy, its high computational burden restricts applicability in real time applications.

To achieve a higher accuracy under assigned computational budget, another Gaussian filter named Gauss-Hermite filter (GHF) [9] was introduced. GHF makes use of Gauss-Hermite quadrature rule for univariate systems. This univariate quadrature rule is extended to multidimensional domain by using the product rule, which in turn results in an exponential rise in multivariate quadrature points and hence suffers from the *curse of dimensionality* problem. This hinders the practical applicability of the filter for higher dimensional problems. We focus our study on decreasing the computational load of Gauss-Hermite filter without hampering its accuracy.

In an earlier approach, sparse-grid Gauss-Hermite filter (SGHF) was introduced which achieves similar accuracy as compared to the GHF, with reduced computational load [10]. In this technique, the univariate quadrature rule is extended to multivariate case with the help of Smolyak rule [11, 12].

In this paper, we propose a novel approach which further reduces the computational burden of Gauss-Hermite filtering. The proposed method is named as adaptive sparse-grid Gauss-Hermite filter (ASGHF). The conventional sparse-grid method treats all the dimensions equally, by default, resulting in no immediate advantage for problems where the dimensions are of differing nonlinearity. But the proposed method uses adaptive sparse-grid technique [13] which automatically finds the dimensions with comparatively lower degree of nonlinearity and generate fewer points for approximation along them which further results in reduced computational cost.

Another advantage of using this method is that it provides a smooth relation between accuracy and computational burden. Unlike the GHF and the SGHF, a small rise in computational burden is possible in the proposed method for a corresponding small increase in the accuracy, by varying the predefined tolerance level and error weighting parameters. It enables the system to work with maximum efficiency possible within the allotted computational budget.

2 Sparse-grid Gauss-Hermite filter

While computing the mean and covariance matrix in an approximate Gaussian filter such as the GHF or SGHF, one encounters integrals of the form:

$$\mathbf{I}_n(f^n(\mathbf{x})) = \int_{R^n} f^n(\mathbf{x})\mathcal{N}(\mathbf{x}; 0, \mathtt{I}_n)d\mathbf{x}, \tag{3}$$

where $f^n(\mathbf{x})$ is an n-dimensional nonlinear function and $\mathtt{I}_n$ is an n-dimensional unity matrix. In SGHF, this integral is approximated using Smolyak rule which makes use of difference formulas $\triangle_l f^1(\mathbf{x}) = (I_l - I_{l-1})f^1(\mathbf{x})$; $I_0 = 0$. Here I_l is a single dimensional quadrature rule with $(2l-1)$ univariate quadrature points. The set of points and weights for I_l can be generated using any of the moment matching method and Golub's Technique [9]. Using Smolyak rule [13],

$$\begin{aligned}\mathbf{I}_n(f^n(\mathbf{x})) &= \sum_{|\mathbb{I}|_{n,L} \leq L+n-1} (\triangle_{l_1} \otimes \cdots \otimes \triangle_{l_n}) f^n(\mathbf{x}) \\ &= \sum_{\Xi \in N_q^n} (\triangle_{l_1} \otimes \cdots \otimes \triangle_{l_n}) f^n(\mathbf{x}),\end{aligned} \tag{4}$$

where $|\mathbb{I}|_{n,L}$ represents an n dimensional index set with accuracy level L and $\otimes$ stands for tensor product. $\Xi = [l_1 \quad l_2 \cdots l_n]^T$ represent a vector and N_q^n is defined as

$$\begin{aligned} N_q^n &= \left\{ \Xi : \sum_{j=1}^{n} l_j = n + q \right\} \qquad \text{for} \quad q \geq 0 \\ &= \varnothing \qquad\qquad\qquad\qquad\quad \text{for} \quad q < 0 \end{aligned}$$

where $\varnothing$ is null set and q is an integer *i.e.* $L - n \leq q \leq L - 1$.

3 Adaptive sparse-grid Gauss-Hermite filter

As discussed earlier, SGHF reduces the computational load of GHF. But still it suffers from two disadvantages:

1. Although the computational burden of the SGHF is lower than the GHF, it rises sharply with dimension of the system.

2. The accuracy level L is the only parameter to control the accuracy versus computational burden relation for SGHF. Even a unit increase in it often results in drastic increase in computational burden. Due to this, the online system usually works below its efficiency under the assigned computational budget.

To overcome the above mentioned shortcomings, we propose a modification to SGHF which is named as adaptive sparse-grid Gauss-Hermite filter (ASGHF). The proposed method reduces the computational burden of SGHF without compromising with accuracy. Further, the accuracy and the computational burden can be controlled using two predefined parameters, namely the local error indicator and the error tolerance, which are discussed in subsequent part of this section. This provides a far better tuning control on the accuracy versus computational burden relation than the use of accuracy level in SGHF. Hence the online system can be made to work near its full efficiency.

3.1 Notation

1. Index set $\mathbb{I}_n$: Let Z^ϕ with $\phi = 1, 2, \cdots$ denotes subsets of $\mathbb{Z}_+$ (set of all positive integers), each of which may or may not be finite. Then, an index set of dimension n can be defined as $\mathbb{I}_n^\phi = \{\lambda : \lambda = (\lambda_1\ \lambda_2\ \cdots\ \lambda_n),\ \lambda_i \in Z^\phi\}$. A possible example of an index set for $n = 2$ is $(1\ 1), (2\ 1), (1\ 2), (2\ 2), (3\ 1), \cdots$. Since the order of subsets Z^ϕ is of no relevance to the subsequent discussion, we drop the superscript ϕ henceforth; keeping in mind that each $\mathbb{I}_n$ has a different set of positive integer valued vectors in general. Moreover, one can notice here that the index set $\mathbb{I}_n$ is an ordered set.

2. Forward index: A set of forward indices for each fixed vector λ is given by $\lambda + e_j,\ j = 1, 2, \cdots, n$, where e_j is the j^{th} unit vector and $\lambda \in \mathbb{I}_n$ is a member of the index set.

3. Backward index: A set of backward indices for any index λ with $\lambda_j > 1$, is defined as $\lambda - e_j$.

4. Admissible set: An ordered index set $\mathbb{I}_n$ is said to be admissible if all the backward indices of any index $\lambda \in \mathbb{I}_n$ lies in $\mathbb{I}_n$. Mathematically, it can be represented as $\lambda - e_j \in \mathbb{I}_n \quad \forall \quad \lambda \in \mathbb{I}_n, 1 \le j \le n, \lambda_j > 1$. For example, an index set $\{(1,1), (2,1), (1,2), (2,2)\}$ is admissible while $\{(1,1), (2,1), (2,2)\}$ is not admissible.

5. Local error indicator (g_λ): This indicates the error at each index. In algorithm, it is used to achieve a trade-off between accuracy and computational complexity. It is given as [13]

$$g_\lambda = max\left\{\psi \frac{|\triangle_\lambda f|}{|\triangle_{\mathrm{I}_1} f|}, (1-\psi)\frac{\varpi_{\mathrm{I}_1}}{\varpi_\lambda}\right\}, \tag{5}$$

where $|\triangle_\lambda f|$ stands for the first norm of absolute $\triangle_\lambda f$, $\mathrm{I}_1 = (1, 1, \cdots, 1)$ represents the first entry of the ordered index set $\mathbb{I}_n$, $\psi \in [0, 1]$ is error weighting parameter and ϖ_λ defines the number of function evaluations as a proxy for computational load. Later in this section, it will be understood that ϖ_{I_1} is unity. The difference formula $\triangle_\lambda f$ for a vector $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_n)$ is defined as

$$\triangle_\lambda f = (\triangle_{\lambda_1} \otimes \triangle_{\lambda_2} \otimes \cdots \otimes \triangle_{\lambda_n}) f. \tag{6}$$

6. Active index set $\mathbb{A}$: This set contains the indices whose error indicators have already been calculated and the error indicators of its forward neighbours have not been examined.

7. Old index set $\mathbb{O}$: This set holds all the other indices of $\mathbb{I}_n$ which are not included in $\mathbb{A}$.

8. Global error estimate $\mho$: It gives the sum of all g_λ present in active index set $\mathbb{A}$.

9. TOL: Error tolerance value which is predefined by the user. This value decides the termination of the computation when some specific accuracy is achieved.

3.2 Algorithm for approximation of multivariate integrals

The objective is to approximate the intractable integrals appeared during approximate nonlinear filtering. In SGHF, this approximation is performed with the help of tensor product of difference formulas over the indices appeared in index set N_q^n, as shown in (4). In the proposed method, we modify the original sparse-grid construction. The selection of the index set over which the difference formula is computed and summed, is rederived in order to ignore the unwanted entries of N_q^n and reduce the computational burden. To achieve this, the proposed method adopts an adaptive approach which generates fewer points along the lower nonlinear dimensions. To this regard, the N_q^n used in SGHF is replaced by an admissible index set $\mathbb{I}_n$ in ASGHF. The generation of $\mathbb{I}_n$ could be understood in a later part of this section.

The modified sparse-grid construction based on the admissible index set $\mathbb{I}_n$ can be expressed as [13]

$$\mathbf{I}_n(f) \approx \sum_{\lambda \in \mathbb{I}_n} \triangle_\lambda f^n(\mathbf{x}) = \sum_{\lambda \in \mathbb{I}_n} (\triangle_{\lambda_1} \otimes \cdots \otimes \triangle_{\lambda_n}) f^n(\mathbf{x}). \tag{7}$$

From the above expression, it is clear that the main challenge is to generate the admissible index set $\mathbb{I}_n$.

The generation of the admissible index set $\mathbb{I}_n$ and functioning of the proposed algorithm can be described as follows:

Initialization:

- First of all, two predefined controlling parameters, the error weighting parameter and the tolerance level are set with a numeric value. Proper selection of these parameters leads to a good trade-off between the accuracy and computational burden.

- The algorithm starts with the index $\mathrm{I}_1 = (1, 1, \cdots, 1)$, which is the first entry of index set $\mathbb{I}_n$.
- At the beginning, define the old index set and the active index set with a null set and I_1 respectively.
- Compute the local error indicator for the index I_1 appeared in active index set. Also, initialize the global error estimate with an arbitrarily large value.

Processing:

The algorithm follows the following steps:

Step 1: Check the condition 'global error estimate $> TOL$'. If it is satisfied, the index with largest error indicator value is transferred from the active index set to the old index set.

Step 2: Deduct the error indicator value of the transferred index from the last stored value of global error estimate.

Step 3: Now, check for the forward indices of the transferred index. Add each of the forward index in active index set, if this addition does not disturb the admissibility condition.

Step 4: As soon as all the eligible forward indices are transferred to active index set, a fresh global error estimate is calculated by adding the error indicator values of all the indices appeared in the active index set.

Step 5: Return to step 1. If it fails to satisfy the condition discussed in step 1, we get the desired index set $\mathbb{I}_n$ by calculating the union of active index set and old index set.

Step 6: Compute the difference formula over the index set $\mathbb{I}_n$ and add all the individual results to get the approximation of integrals, as discussed in Eq. (7).

Pseudo algorithm for integral approximation

```
$integrate(f)$
$i := (1, 1, \cdots, 1)$
$\mathbb{O} := \phi$
$\mathbb{A} := \{i\}$
$r := \triangle_i f$
$\mho_1 = g_i$
```

```
$initialize\ \mho\ with\ arbitrarily\ high\ value.$
$if\quad (\mho_1 > TOL)$
$select\ i\ from\ \mathbb{A}\ with\ largest\ g_i$
$\mathbb{A} := \mathbb{A}\ - \{i\}$
$\mathbb{O} := \mathbb{O} \cup \{i\}$
$\mho_1 := \mho_1 - g_i$
$for\ j := 1, 2, \cdots, n$
$\lambda = i + e_k$
$if\ \lambda - e_q \in \mathbb{O}\ for\ all\ q = 1, 2, \cdots, n,\ then$
$\mathbb{A} := \mathbb{A} \cup \{\lambda\}$
$s := \triangle_\lambda f$
$r := r + s$
$\mho_1 := \mho_1 + g_\lambda$
$\mho = \mho_1$
$end\ if$
$end\ for$
$end\ if$
$return\ r$
```

where $\triangle_i f$= integral increment, $\otimes_{k=1}^{n} \triangle_{i_k} f$
and r= computed integral value, $\sum_{i\in\mathbb{O}\cup\mathbb{A}} \otimes_{k=1}^{n} \triangle_{i_k} f$.

Note: In the pseudo algorithm, the parameter $\mho_1$ is actually the global error estimate $\mho$. But, it is defined separately in order to discard the stopping of algorithm before entering to the body of '$if(\mho > TOL)$'. Hence throughout the paper, no difference has been considered between these two.

3.3 Generation of points and weights

In the proposed method, the integral approximation is a recursive process and in each recursion, the function to be approximated is used. Hence the proposed method is model dependent. In each iteration, the function is approximated using the difference formula expressed in Eq. (6) which use a set of sample points and the corresponding weights to approximate the integral. The final approximation of the integral is the sum of function approximation over all iterations. Hence, if we store all the points and corresponding weights used over different iteration, the integral can be approximated numerically using the stored points and weights.

To this regard, the integrals of interest with process and measurement functions are approximated offline with a predefined tolerance, before using them in the filtering algorithm. All the points and weights used over different iterations during approximation of the integral are stored. Then the filtering algorithm mentioned in [14] is used.

3.4 Adapting to the degree of nonlinearity along different dimensions

As discussed earlier, the proposed method puts less effort towards the dimension with lower nonlinearity *i.e.* generates fewer quadrature points along those dimensions. For this purpose, it does not use a dedicated method to identify the degree of nonlinearity but the error parameter g_λ helps to accomplish the objective adaptively.

It is obvious that the difference formula $\triangle_\lambda$ will provide poor approximation *i.e.* the absolute value of $\triangle_\lambda f$ will be higher if the system has higher nonlinearity. So while checking the forward indices (for next entry), a comparison between the absolute values of $\triangle_{\lambda^i} f$ (λ^i is i^{th} λ for $i = 1, 2, \cdots, n$) may help to identify the dimension for which the nonlinearity is highest (the dimension with highest absolute value of $\triangle_{\lambda^i} f$). As soon as such dimension is identified, the next choice of index may be brought from this dimension which may help to put more effort along the dimensions with higher nonlinearity. In this regard, an error parameter g is defined for each indices which makes comparison of errors computed from the difference formula.

From the above discussion, it is apparent that $g = \psi \dfrac{|\triangle_\lambda f|}{|\triangle_{\mathrm{I}_1} f|}$ may be sufficient for identifying different nonlinearity in different dimensions and putting less efforts along the lower nonlinear dimensions. However, it will not help the practitioners to restrict the computational cost below a preassigned computational budget even if they are ready for a limited compromise with the accuracy. Hence, as shown in Eq. (5), a second term is incorporated in the expression of g which enables the practitioners to take a control over the computational cost. It justifies the earlier statement that the local error indicator g also helps to access a trade-off between the accuracy and computational budget.

Remark 1. *If for an online implementation, the accuracy is crucial and the system is equipped to* *afford a high computational burden, the designer should choose a higher value of ψ and vice versa. Hence, ψ acts as controlling parameter which helps the algorithm to take sensible decisions when comparatively low error or high computational work is encountered.*

Remark 2. *The expressions for prediction and update in approximate nonlinear filtering methods such as SGHF are widely discussed in the literature and are omitted here for brevity; the reader is referred to [1] for example.*

3.5 Selection of error weighting parameter and tolerance level

As discussed earlier, the selection of error weighting parameter and tolerance level depends on available computational budget. To this regard, an offline implementation over the expected model will be required before going for real-life implementation.

It is apparent from the above discussions that a high value of error weighting parameter (which lies between 0 and 1) and a low tolerance level are responsible for high accuracy but, at the same time, need a high computational time. To this regard, the practitioners may begin with a near unity value for error weighting parameter with a significantly low tolerance level. Then the required computational time for the specific selection should be compared with the available computational budget. If the required computational time remains higher, another attempt should be made with a reduced error weighting parameter or increased tolerance level or both. The same procedure should be repeated until the required computational time is not less than the available computational budget. Once this condition is achieved, the specific set of error weighting parameter and tolerance level should be chosen for online implementation.

It is to be noted here that a prior offline implementation is a common practice for many other purposes as well, like noise parameter selection, model validation *etc*. Subsequently, an offline implementation constrain does not affect the reliability of the algorithm.

3.6 Illustration

In this subsection, the working of the algorithm is illustrated for a two dimensional nonlinear dynamic system. The dynamic behavior of the system is given as

$$[x_{1,k+1} \;\; x_{2,k+1}]^T = \left[e^{-x_{1,k}} \;\; e^{-x_{2,k}^2}\right]^T. \tag{8}$$

To evaluate the integral, initial mean and covariance is taken as $[0 \;\; 0]^T$ and $diag([0.4 \;\; 0.2])$ respectively. The error weight and tolerance are assumed as 0.725 and 0.05 respectively.

1. At starting, $\mathbb{O} = \phi$, $\mathbb{A} = \{(1,1)\}$, $\mho_1 = 50$ and $\mho = g_{(1,1)} = 0.725$. $\triangle_{(1,1)}f$ (difference formula for $(1,1)$) is calculated and result is stored in r, so $r = [1\ 1]^T$.

2. The index with largest error indicator value is searched in $\mathbb{A}$. It is $(1,1)$, as there is single point. Its corresponding error value is subtracted from the global error, hence $\mho = 0$. Then, $(1,1)$ is transferred to old index set and its forward indices are added into $\mathbb{A}$. This happens only if all the backward indices of incoming index is already present in old index set. Hence $\mathbb{O} = (1,1)$ and $\mathbb{A} = \{(2,1),(1,2)\}$. The difference formula for new indices are evaluated and the results are summed with the values in r. The error indicator value of all indices appeared in $\mathbb{A}$ are added to compute the global error value. So, $\mho_1 = \mho = 0.137$.

3. As the global error is still greater than the tolerance value, the algorithm will proceed further.

4. g is a set, *i.e.* g_i indicates the error indicator value of i^{th} index of $\mathbb{A}$, then $g = \{0.0688 \;\; 0.0688\}$.

5. As both the indices have same value, any one can be selected, we select $(2,1)$.

6. $(2,1)$ is transferred to $\mathbb{O}$ and its forward indices $(2,2)$ and $(3,1)$ are looked up. All the backward indices of $(2,2)$ are not present in old index set, so only $(3,1)$ is transferred to $\mathbb{A}$. Hence $\mathbb{O} = \{(1,1) \;\; (2,1)\}$ and $\mathbb{A} = \{(1,2) \;\; (3,1)\}$.

7. The corresponding error indicator set is calculated as $g_i = \{0.0688 \;\; 0.0344\}$.

8. Again the index with maximum error indicator value, *i.e.* $(1,2)$, is selected from $\mathbb{A}$ and transferred to $\mathbb{O}$.

9. Similar procedure continues until we get the global error below the predefined tolerance level, 0.05. The final approximation of integral will be the value stored in r.

3.7 Advantages of ASGHF over GHF and SGHF

1. The computational load of ASGHF is lower than GHF and SGHF at similar accuracy levels.

2. There are two controlling parameters in ASGHF algorithm, $viz.$ the error tolerance TOL and the error weighting parameter ψ. These two parameters help the algorithm to find the dimensions with a higher degree of nonlinearity and refine them accordingly.

3. In GHF and SGHF, an increase in accuracy level by unity will lead to a sharp increase in the computational load. In the proposed filter, a small increase in accuracy can be acheived by varying the tolerance level or the error weighting parameter, with a proportionately small increase in the computational cost. Thus, the proposed filter gives a comparatively smoother relation between estimation accuracy and computational cost.

4. Under assigned computational budget, the ASGHF enables the online estimator to work near the full efficiency, as the trade-off between the accuracy and computational cost can be fine-tuned, as mentioned above.

4 Simulation

In this section, the adaptive sparse-grid Gauss-Hermite quadrature rule (used in ASGHF) is first implemented for approximation of a simple multidimensional integral, and then to two real-life nonlinear filtering problems. For all the problems its performance is compared with GHF and SGHF. During the implementation for real-life nonlinear filtering problems, a 3-point GHF and a 3^{rd}-degree of accuracy level for SGHF (*i.e.* $L = 3$) have been considered.

4.1 Problem 1: Approximate evaluation of a multidimensional integral

Let us assume, $\mathbf{x} = [x_1\ x_2 \cdots x_n]^T$ be an n-dimensional vector, and the integral under consideration to be

$$I_n = \int_{-\infty}^{\infty} \sum_{i=1}^{n} x_i^{2i} d\mathbf{x}. \tag{9}$$

The above integral (with $n = 6$) is approximated using the adaptive sparse-grid Gauss-Hermite (ASGH) quadrature rule, the sparse-grid Gauss-Hermite (SGH) quadrature rule and the Gauss-Hermite (GH) quadrature rule. The results have been compared in Table 1 where GH_t represents a t-point GH rule, SGH_L represents SGH rule with accuracy level L and ASGH_$\{\psi, TOL\}$ represents ASGH rule with error weighting parameter ψ and tolerance level TOL. From the table, it could be concluded that the ASGH rule requires a significantly small number of sample points for achieving similar accuracy with respect to GH and SGH quadrature rules.

Filters	% Error	Number of sample points
GH_3	77.8843	729
GH_4	36.3747	4096
GH_5	7.8139	15625
GH_6	0.4784	46656
SGH_3	7.8066	97
SGH_4	0.0042	533
ASGH_{ 0.1,5 }	0.0138	64
ASGH_{ 0.4,5 }	0.0107	88
ASGH_{ 0.4,1.6 }	0.0042	110

Table 1: % error and sample point requirement for different quadrature rules

4.2 Problem 2: Estimation of multiple superimposed sinusoids

In this problem, we estimate the amplitude and frequency of multiple superimposed sinusoids. Such problems practically appear in many fields like communication systems [15], power systems [16] *etc.*

We consider, the number of sinusoids as three, then the state variable will be $\mathbf{x} = [f_1\ f_2\ f_3\ a_1\ a_2\ a_3]^T$, where f_i and a_i are the frequency and amplitude of i^{th} sinusoid. The discretized process model is

$$\mathbf{x}_k = \mathtt{I}_6\mathbf{x}_{k-1} + w_k, \tag{10}$$

where $\mathtt{I}_6$ is a six dimensional unit matrix and w_k is process noise normally distributed with zero mean and covariance $Q = diag([\sigma_f^2\ \ \sigma_f^2\ \ \sigma_f^2\ \ \sigma_a^2\ \ \sigma_a^2\ \ \sigma_a^2])$ with σ_f and σ_a being the standard deviations for frequency and amplitude.

The measurement equation is [17]

$$y_k = \begin{bmatrix} \sum_{j=1}^{3} a_{j,k} cos(2\pi f_{j,k} kT) \\ \sum_{j=1}^{3} a_{j,k} sin(2\pi f_{j,k} kT) \end{bmatrix} + v_k,$$

where v_k is Gaussian noise with zero mean and covariance $R = diag([\sigma_n^2\ \ \sigma_n^2])$ with σ_n being the standard deviation for measurement noise. T is the sampling time which is considered as 0.1667 ms.

The initial truth and estimates are considered as $[200\ 1000\ 2000\ 5\ 4\ 3]^T$ and $[150\ 900\ 1800\ 4\ 4\ 2]^T$ respectively. Varying the initial error covariance and noise covariances, we consider two different scenarios as

scenario 1:

$\sigma_f^2 = 151\mu Hz^2/ms^2$, $\sigma_a^2 = 80\mu V^2/ms^2$, $\sigma_n^2 = 0.09V^2$, and $P_{0|0} = diag([20^2\ 20^2\ 20^2\ 0.05\ 0.05\ 0.05])$.

scenario 2:

$\sigma_f^2 = 300\mu Hz^2/ms^2$, $\sigma_a^2 = 160\mu V^2/ms^2$, $\sigma_n^2 = 0.18V^2$, and $P_{0|0} = diag([50^2\ 50^2\ 50^2\ 0.5\ 0.5\ 0.5])$.

For ASGHF, the simulation is performed by considering the error weighting parameters as 0.6 and 0.5, while tolerance as 0.53 and 0.6655 for process and measurement equations respectively. The states are estimated for 500 steps and the results are averaged over 2000 Monte Carlo runs. At each step, a combined error parameter (ERR) is evaluated for frequency and amplitude, which is defined as

$$ERR_k = \sqrt{\frac{MSE_{1,k} + MSE_{2,k} + MSE_{3,k}}{3}}, \tag{11}$$

where, for M number of Monte Carlo runs, $MSE_{i,k}$ is

$$MSE_{i,k} = \frac{1}{M}\sum_{j=1}^{M}(\mathbf{x}_{i,k,j} - \hat{\mathbf{x}}_{i,k,j})^2. \tag{12}$$

Filters	Relative comp. time (Prob. 1)
GHF	1
SGHF	0.17
ASGHF	0.056

Table 2: Relative computational time for various filters

The ERR for frequency and amplitude are plotted for two different scenarios in Fig. 1 and Fig. 2 using the proposed ASGHF, SGHF and GHF. The ERR is similar for all the filters and hence it could be concluded that the accuracy of the proposed algorithm is similar to the conventional GHF and SGHF. On the other hand, from the Table 2, it could be concluded that the computational burden for the proposed ASGHF is almost 3 times lower than the SGHF and 18 times lower than the conventional GHF.

4.3 Problem 3: Maneuvering target tracking

The second problem is a tracking problem of a target following coordinated turn model [1]. The discretized target dynamics can be represented as

$$\mathbf{x}_{k+1} = F_k\mathbf{x}_k + w_k, \tag{13}$$

where $\mathbf{x} = [\mathrm{x}\ \dot{\mathrm{x}}\ \mathrm{y}\ \dot{\mathrm{y}}\ \omega]^T$ with x and y being the positions in x and y directions respectively, ω is the angular turn rate and

$$F_k = \begin{bmatrix} 1 & \dfrac{\sin(\omega_k T)}{\omega_k} & 0 & -\dfrac{1-\cos(\omega_k T)}{\omega_k} & 0 \\ 0 & \cos(\omega_k T) & 0 & -\sin(\omega_k T) & 0 \\ 0 & \dfrac{1-\cos(\omega_k T)}{\omega_k} & 1 & \dfrac{\sin(\omega_k T)}{\omega_k} & 0 \\ 0 & \sin(\omega_k T) & 0 & \cos(\omega_k T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Model	scenario 1		scenario 2	
	ψ	*TOL*	ψ	*TOL*
Process model	0.55	0.5	0.525	0.5
Measurement model	0.6	0.48	0.6	0.48

Table 3: Two different scenarios for Problem 3

The nonlinear measurement equation can be described in general as

$$y_k = \gamma(\mathbf{x}_k) + v_k. \tag{14}$$

Here, we assume that both the range and bearing angle are available from measurements and hence, the measurement equation can be written as

$$y_k = \left[\sqrt{\mathrm{x}_k^2 + \mathrm{y}_k^2} \quad atan2(\mathrm{y}_k, \mathrm{x}_k) \right]^T + v_k,$$

where $atan2$ is the four quadrant inverse tangent function. w_k and v_k are considered to be white Gaussian noise with zero mean and covariances Q and R respectively. The process noise covariance is

$$Q = q \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} & 0 & 0 & 0 \\ \frac{T^2}{2} & T & 0 & 0 & 0 \\ 0 & 0 & \frac{T^3}{3} & \frac{T^2}{2} & 0 \\ 0 & 0 & \frac{T^2}{2} & T & 0 \\ 0 & 0 & 0 & 0 & 0.009T \end{bmatrix},$$

where $q = 0.1$ is a given constant and T=0.5 seconds is the sampling time. $R = \text{diag}([\sigma_r^2 \ \sigma_t^2])$, where $\sigma_r = 120$m and $\sigma_t = \sqrt{70}$mrad.

The initial truth value is considered as $\mathbf{x}_0 = [1000\text{m} \ \ 30\text{m/s} \ \ 1000\text{m} \ \ 0\text{m/s} \ \ \omega^\circ/s]^T$, while the initial covariance is $P_{0|0} = \text{diag}([200\text{m}^2 \ \ 20\text{m}^2/\text{s}^2 \ \ 200\text{m}^2 \ \ 20\text{m}^2/\text{s}^2 \ \ 100\text{mrad}^2/\text{s}^2])$. The initial estimate is considered to be normally distributed with mean $\mathbf{x}_0$ and covariance $P_{0|0}$. For ASGHF, we consider two different scenarios by selecting different set of predefined parameters as shown in Table 3.

The motivation of considering two parametric scenarios is to study the experimental behavior of the proposed method over the accuracy and the computational burden by varying these predefined parameters. To this regard, the simulation is performed for 100 seconds and the results are obtained in terms of root mean square error (RMSE) of the range and the velocity for 500 independent Monte Carlo runs. The performance of the proposed method is studied and compared for varying turn rate. The RMSEs are plotted for $\omega = 3^\circ/sec$ and $\omega = 4.5^\circ/sec$ in Fig. 3 and Fig. 4 respectively for scenario 1 and in Fig. 5 and Fig. 6 respectively for scenario 2. At the same time, the relative computational burdens are listed in Table 4.

Filters	Relative comp. time (Prob. 2)	
	scenario 1	*scenario 2*
GHF	1	1
SGHF	0.36	0.36
ASGHF	0.12	0.28

Table 4: Relative computational time for various filters

Under the first scenario, from the Table 4 and Fig. 3 and 4, it could be concluded that the computational burden for the proposed method is around $1/8^{th}$ and $1/3^{rd}$ times lower than the conventional GHF and SGHF respectively, but it lags behind in terms of accuracy. In order to achieve similar accuracy, the error weighting parameter is tuned to a value as presented in scenario 2 (Table 3). The computational load increases in this scenario but it still remains less than the GHF and SGHF, while a similar accuracy to these filters could be obtained. It is to be noted that the different computational times are obtained on a personal computer with 64-bit operating system, 4 GB RAM and 3.33 GHz clock speed, on a MATLAB version 2010b.

From the tables, it could be concluded that the improvement in computational efficiency using the proposed method is not similar for both the problems. This dissimilarity is because of the different dimension and the different degree of nonlinearity for process and measurement models. As any of these two factors increases, the degree of improvement in computational efficiency rises.

5 Discussions and conclusions

The Gauss-Hermite quadrature rule based filters, namely the GHF and the SGHF are among the most accurate nonlinear filtering approximations available in literature. However, these filters are often not fit for real-life on-board implementation because of their high computational burden. Apart from high computational burden, another serious disadvantage with these filters is that a unit increase in the accuracy level or the number of univariate quadrature points leads to an exponential rise in the computational burden. Hence, the online estimators mostly work much below their full efficiency.

To overcome these disadvantages, this paper proposes a new Gauss-Hermite quadrature rule based filtering technique, named as adaptive sparse-grid Gauss-Hermite filter (ASGHF). It could reduce the computational burden without compromising with the estimation accuracy. Moreover, the presence of two predefined control parameters, namely the tolerance level and the error weighting parameter, help in obtaining a better trade off between the accuracy and computational load.

The proposed filter as well as the GHF and SGHF are implemented to solve two different state estimation problems. Simulation results show that the accuracy of ASGHF is similar to the GHF and SGHF while the computational burden is considerably less. Hence, the proposed method has potential to replace the existing filters for real-life applications.

References

[1] Bar-Shalom, Y., Li, X. R., Kirubarajan, T.: ‘Estimation with applications to tracking and navigation: theory algorithms and software’ (John Wiley & Sons, 2004)

[2] Julier, S. J., Uhlmann, J. K.: ‘A new extension of the Kalman filter to nonlinear systems’. Int. symp. aerospace/defense sensing, simul. and controls, Orlando, July 1997, pp. 3–2

[3] Chang, L., Hu, B., Chang, G., Li, A.: ‘Marginalised iterated unscented Kalman filter’, IET Control Theo. Appl., 2012, **6**, (6), pp. 847–854

[4] Bhaumik, S., Sadhu, S., Ghoshal, T. K.: ‘Risk-sensitive formulation of unscented Kalman filter’, IET Control Theo. Appl., 2009, **3**, (4), pp. 375–382

[5] Arasaratnam, I., Haykin, S.: 'Cubature Kalman filters', IEEE Tran. Autom. Control, 2009, **54**, (6), pp. 1254–1269

[6] Zhang, Y., Huang, Y., Li, N., Zhao, Li.: 'Interpolatory cubature Kalman filters', IET Control Theo. Appl., 2012, **9**, (11), pp. 1731–1739

[7] Liu, G., Worgotter, F., Markelic, I.: 'Nonlinear estimation using central difference information filter'. IEEE Stat. Signal Proc. Work., Nice, June 2011, pp. 593–596

[8] Arulampalam, M. S., Maskell, S., Gordon, N., Clapp, T.: 'A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking', IEEE Tran. Signal Proc., 2002, **50**, (2), pp. 174–188

[9] Arasaratnam, I., Haykin, S., Elliott, R. J.: 'Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature', IEEE Proc., 2007, **95**, (5), pp. 953–977

[10] Jia, B., Xin, M., Cheng, Y.: 'Sparse-grid quadrature nonlinear filtering', Automatica, 2012, **48**, (2), pp. 327–341

[11] Smolyak, S. A.: 'Quadrature and interpolation formulas for tensor products of certain classes of functions', Dokl. Akad. Nauk SSSR, 1963, **4**, pp. 240-243

[12] Gerstner, T., Griebel, M.: 'Numerical integration using sparse grids', Numerical algorithms, 1998, **18**, (3-4), pp. 209-232

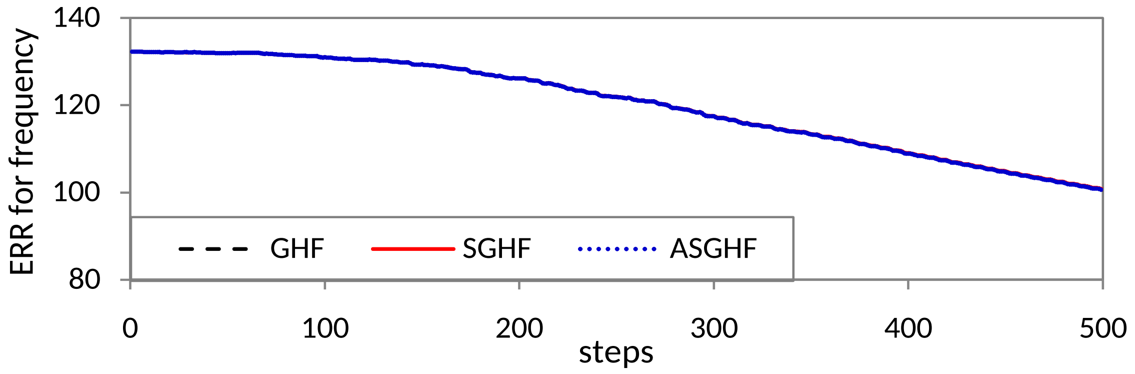
[13] Gerstner, T., Griebel, M.: 'Dimension–adaptive tensor–product quadrature', Computing, 2003, **71**, (1), pp. 65–87

[14] Singh, A. K., Bhaumik, S.: 'Higher degree cubature quadrature Kalman filter', Inter. Jr. Control Autom. Syst., 2015, **13**, (5), pp. 1097–1105

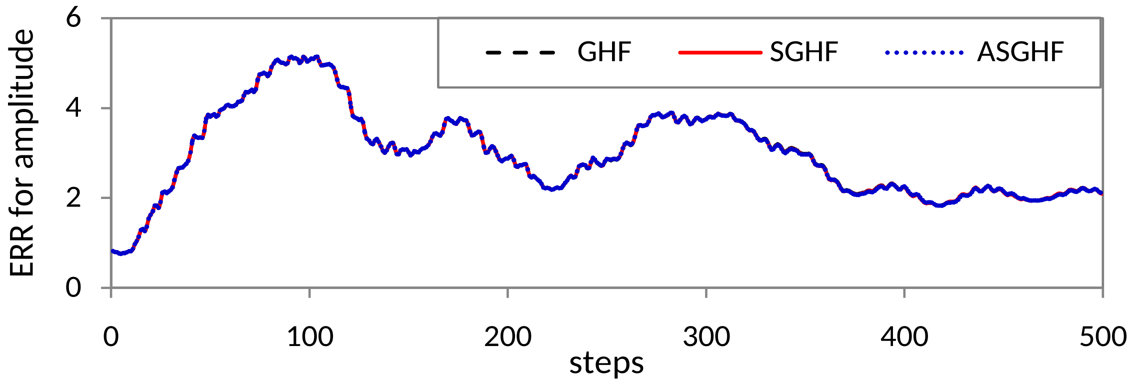
[15] Niedźwiecki, M., Kaczmarek, P.: 'Estimation and tracking of complex-valued quasi-periodically varying systems', Automatica, 2005, **41**, (9), pp. 1503–1516

[16] Reddy, J., Dash, P. K., Samantaray, R., Moharana, A. K.: 'Fast tracking of power quality disturbance signals using an optimized unscented filter', IEEE Tran. Inst. Meas., 2009, **58**, (12), pp. 3943–3952

[17] Closas, P., Prades, C. F.; Valls, J. V.: ‘Multiple quadrature Kalman filtering’, IEEE Tran. Signal Proc., 2012, **60**, (12), pp. 6125–6137

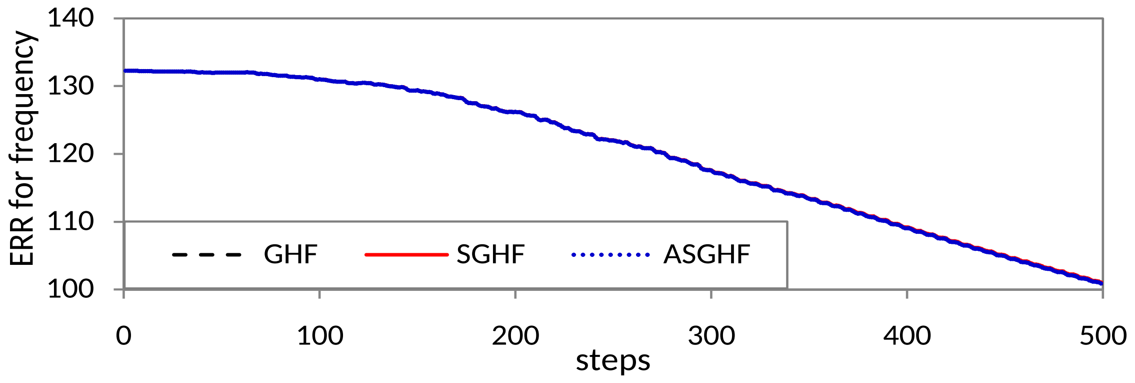


(a)

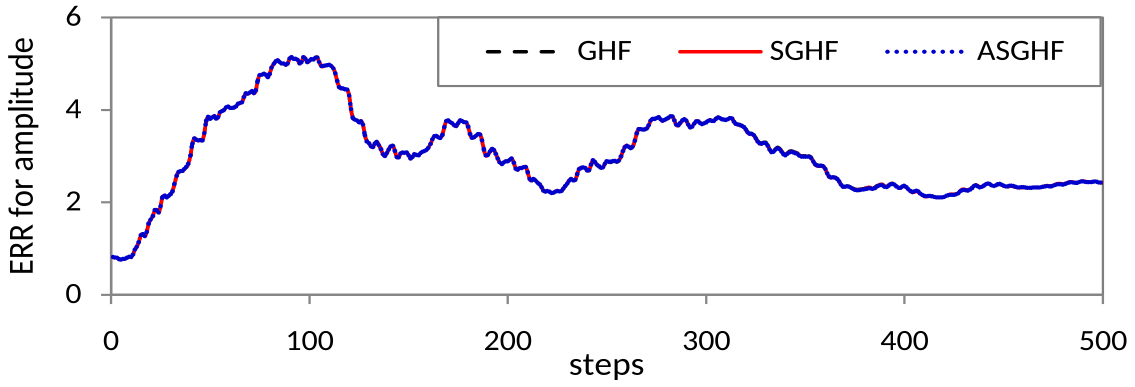


(b)

Figure 1: *Problem 1:* ERR plot for 1^{st} scenario- (a) frequency in Hz (b) amplitude in $volt$.

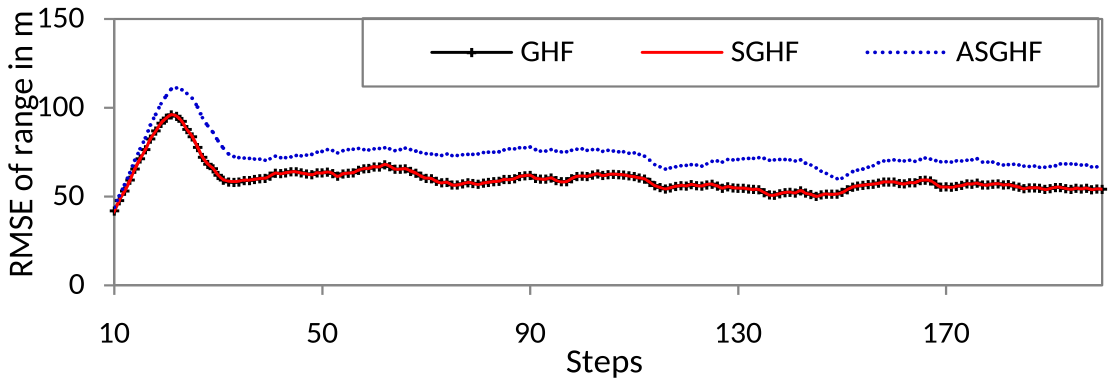


(a)

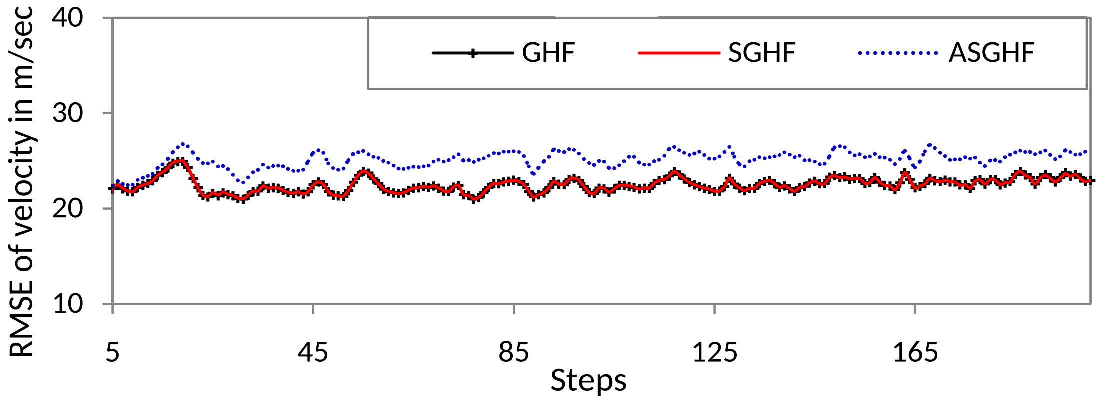


(b)

Figure 2: *Problem 1:* ERR plot for 2^{nd} scenario- (a) frequency in Hz (b) amplitude in $volt$.



(a)



(b)

Figure 3: *Problem 2:* RMSE plots for $\omega = 3^{\circ}$ under 1^{st} scenario- (a) range in m (b) velocity in m/s.

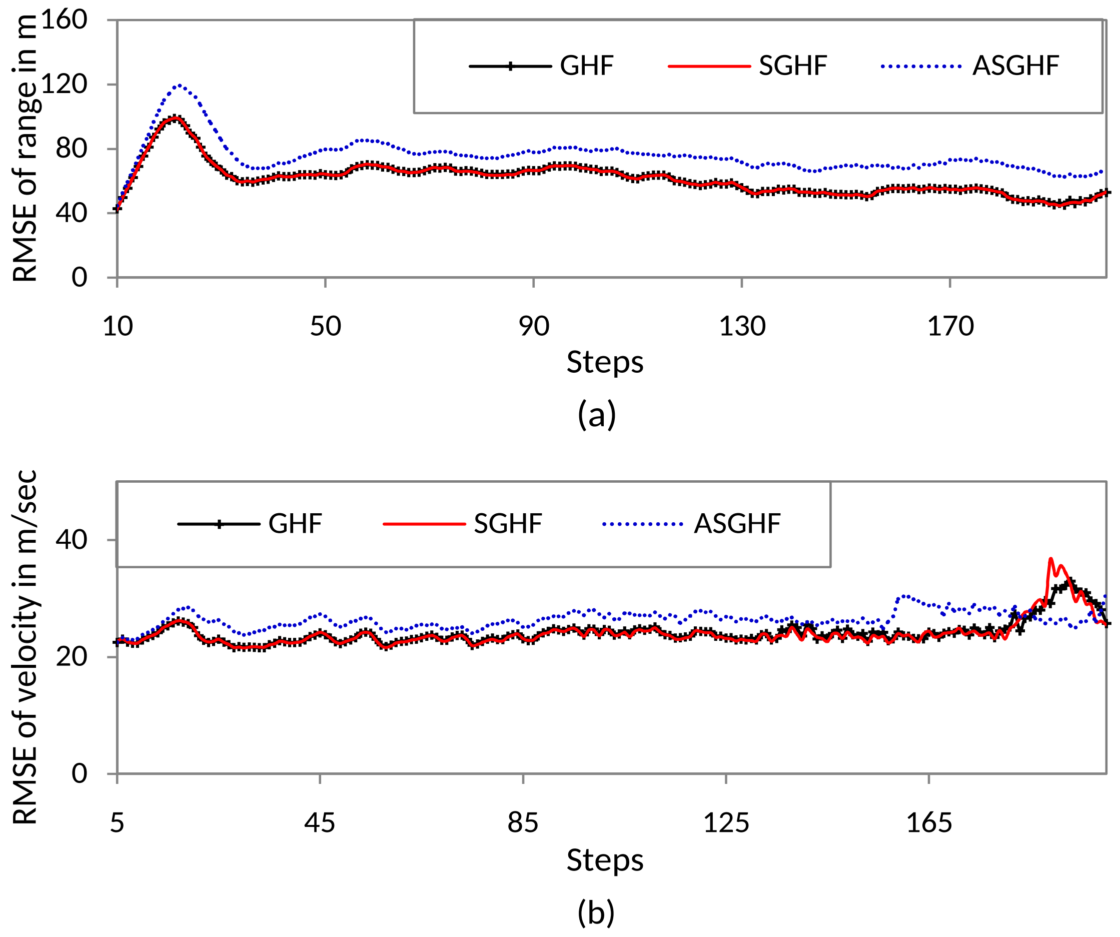
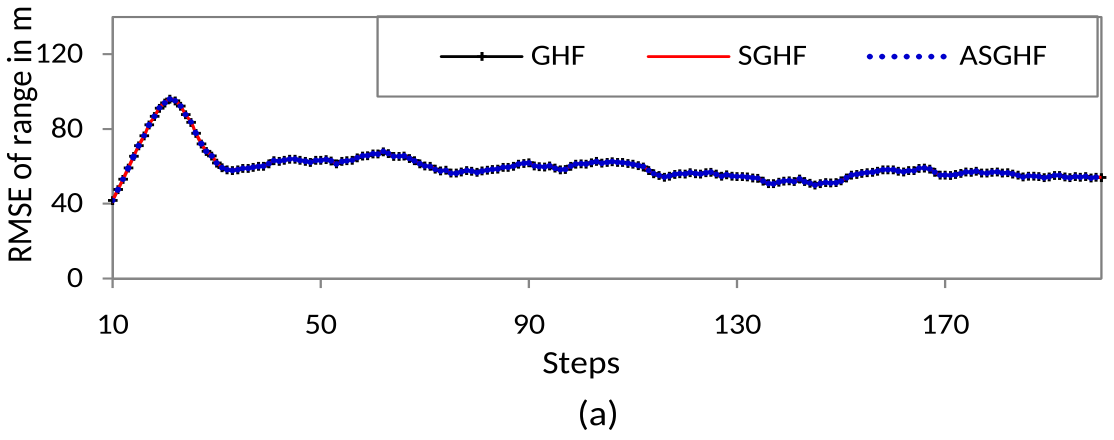


Figure 4: *Problem 2:* RMSE plots for $\omega = 4.5^{\circ}$ under 1^{st} scenario- (a) range in m (b) velocity in m/s



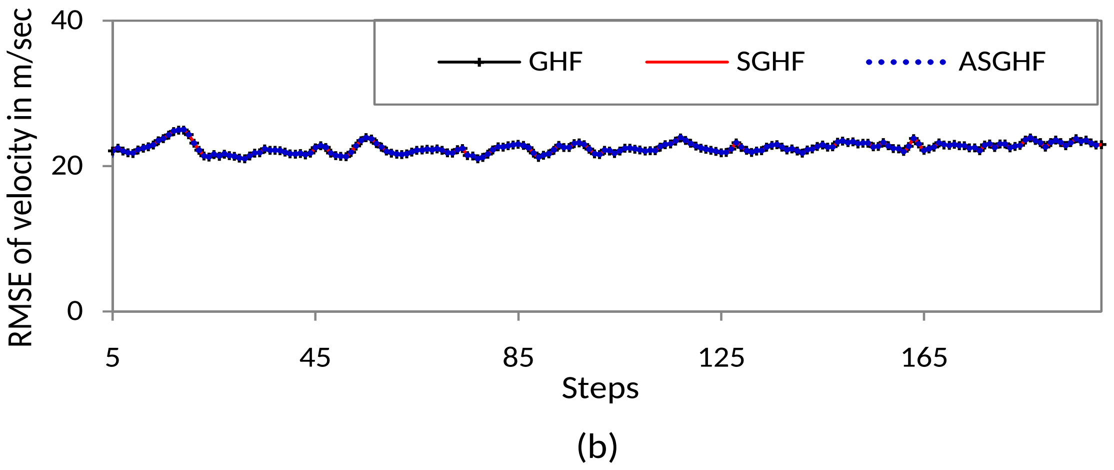
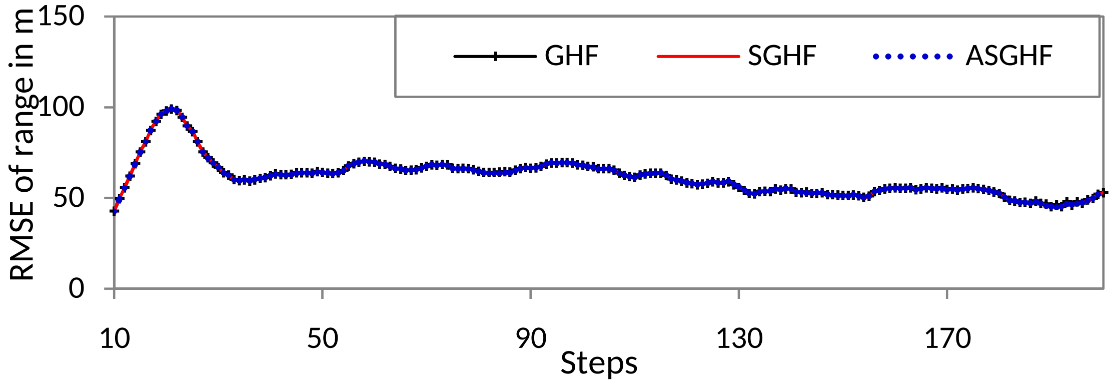
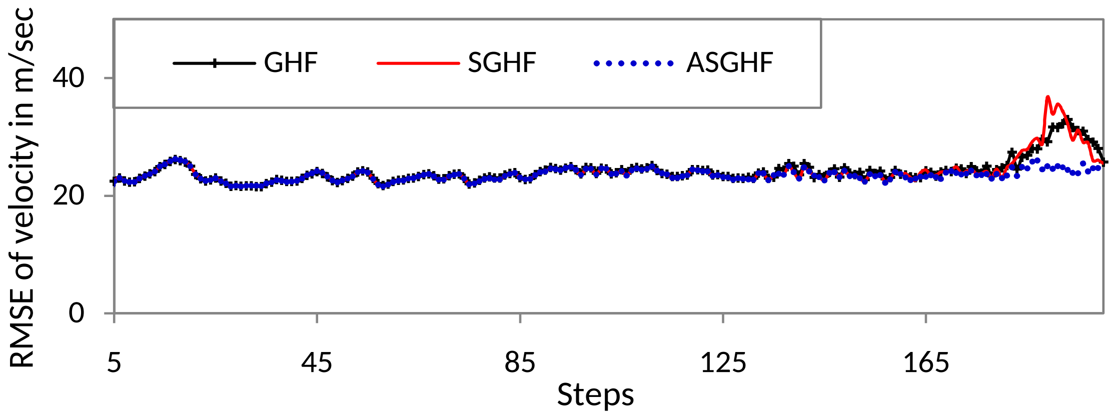


Figure 5: *Problem 2:* RMSE plots for $\omega = 3^\circ$ under 2^{nd} scenario- (a) range in m (b) velocity in m/s.



(a)



(b)

Figure 6: *Problem 2:* RMSE plots for $\omega = 4.5^{\circ}$ under 2^{nd} scenario- (a) range in m (b) velocity in m/s