AAS 12-204

SECOND-ORDER KALMAN FILTERS USING MULTI-COMPLEX STEP DERIVATIVES

Vivek Vittaldev*, Ryan P. Russell†, Nitin Arora‡, and David Gaylor§

The Second Order Kalman Filter (SOKF) uses a second order Taylor series expansion (TSE) to account for nonlinearities in an estimation problem. In this work, the derivatives required for the SOKF are computed using multicomplex (MCX) derivatives, coded in the Matlab programming language. This method uses function overloading in order to derive or compute the derivatives to machine precision without having to compute the derivatives analytically. Thus, the SOKF can be easily implemented, while at the same time having fewer tuning parameters than other high order filters. The standard SOKF is also extended by combining it with Gaussian Mixture models (GMM), which gives promising results. The filters have been used to estimate the state of a 1 DOF falling body. The results show that the MCX computes the required derivatives just as accurately as an analytical method and the SOKF and GMM modification perform well in terms of accuracy compared to other filters. Despite the ease of use and high accuracy benefits, a current drawback of the MCX method is compute speed. Methods for improving the speed are beyond the current scope and will be addressed in future works.

INTRODUCTION

The original Kalman filter (LKF) is only applicable to linear systems and is an optimal filter for those cases.[1] However, most situations where filtering can be used are nonlinear, especially in the case of aerospace systems. Therefore, the Extended Kalman Filter (EKF), which is a nonlinear first order extension of the LKF, has become the de facto filter in the aerospace industry. The EKF works in a similar manner to the LKF, but it uses a first order Taylor series expansion (TSE) of the nonlinear system. The Second Order Kalman Filter (SOKF) is analogous to the EKF, but instead uses a second order Taylor series expansion. The SOKF has existed for decades, with the equations being available in the 1970s.[2,3] In the past decade, there have been advances in using even higher order filters such as the Unscented Kalman Filter (UKF)[4,5,6] and the Divided Difference Filter (DDF).[7] It should be noted that the DDF has a first order variant (DD1), which behaves similarly to the EKF and a second order variant (DD2), which behaves similarly to the UKF and SOKF.[8] The DDF and UKF differ from the EKF and the SOKF because they use a few select points (sigma points) in order to estimate the probabilistic characteristics of the state. Both the UKF and DD2 are capable of filtering with up to third order accuracy, under some common conditions.[4,9,7] Therefore, it is expected that the SOKF will perform better than the EKF and the DD1, but on par with the UKF and DD2. However, the SOKF is expected to perform slightly worse than the UKF and DD2.

*Graduate Student, The University of Texas at Austin, v.vittaldev@utexas.edu.
†Assistant Professor, The University of Texas at Austin, ryan.russell@austin.utexas.edu.
‡Research Scholar,The University of Texas at Austin, n.arora@gatech.edu.
§Vice President, Emergent Space Technologies, Inc., dave.gaylor@emergentspace.com.

MULTI-COMPLEX STEP DERIVATIVES

Originally proposed by Squire & Trapp in 1998,[10] the complex step method stems from the Taylor series expansion of a function f about a complex variable centered at its real component and perturbed in the imaginary direction by a magnitude h:

$$f(x+ih) = f(x) + \frac{df}{dx}ih + \mathcal{O}\left(h^2\right) \tag{1}$$

Neglecting the higher order terms and comparing the imaginary part from both sides of Eq. (1), the approximation of the first derivative is computed as follows:

$$\frac{\partial f}{\partial x} = \lim_{h\to 0} \frac{Im\left[f(x+ih)\right]}{h} \tag{2}$$

Since there is no subtraction, h can be arbitrarily small (e.g. 10^{-50}) and the first derivative can therefore be computed accurately to machine precision as long as f is analytic and is computed in the complex plane. The drawback of this approach is the increased overhead associated with the complex arithmetic. Lantoine, Russell and Dargent[11,12] have recently extended this method to higher dimension through the use of multicomplex (MCX) variables, thereby making it possible to obtain derivatives up to any order.

MCX numbers are the extension of complex numbers into multiple dimensions. Considering only two dimensions, a bicomplex number, z is given as

$$z = a + bi_1 + ci_2 + di_1i_2 : a,b,c,d \in \mathbb{R} \tag{3}$$

where there are two pure imaginary directions, i_1 andi_2; and one coupled direction, $i_1\, i_2$. The imaginary products apply such that $i_1^2 = i_2^2 = -1$ and $i_1i_2 = i_2i_1$.

To derive the second order MCX derivative for a scalar function with respect to one variable, perform a Taylor series expansion of a bicomplex variable evaluated on the real axis at x and perturbed by $h\left(i_1+i_2\right)$

$$\begin{aligned} f\left[x+h\left(i_1+i_2\right)\right] =& f(x) + h\left(i_1+i_2\right)f'(x) + \frac{h^2}{2}\left(i_1+i_2\right)^2 f''(x) \\ &+ \frac{h^3}{3!}\left(i_1+i_2\right)^3 f'''(x) + O\left(h^4\right) \\ =& f(x) + h\left(i_1+i_2\right)f'(x) + \frac{h^2}{2}\left(i_1i_1 + 2i_1i_2 + i_2i_2\right)f''(x) \\ &+ \frac{h^3}{3!}\left(i_1^3 + 3i_1^2i_2 + 3i_1i_2^2 + i_2^3\right)f'''(x) + O\left(h^4\right) \end{aligned}$$

Noting that $i_1^2 = i_2^2 = -1$ results in

$$\begin{aligned} f\left[x+h\left(i_1+i_2\right)\right] =& f(x) + h\left(i_1+i_2\right)f'(x) + h^2\left(i_1i_2\right)f''(x) - h^2f''(x) \\ &+ \frac{4h^3}{3!}\left(-i_1-i_2\right)f'''(x) + O\left(h^4\right) \end{aligned} \tag{4}$$

Now take the coefficient of i_1i_2 from both sides,

$$\mathrm{Im}_{12}\left(f\left[x+h\left(i_1+i_2\right)\right]\right) = h^2f''(x) + O\left(h^4\right) \tag{5}$$

noting that there are no $i_1 i_2$ terms of $O\left(h^3\right)$ while terms of $O\left(h^4\right)$ indeed have nonzero $i_1 i_2$ coefficients (for example, $i_1 i_2 i_1 i_1$ is $-i_1 i_2$).

Finally, divide by h^2 and the second order formula for the MCX derivative is achieved

$$f''(x) = \frac{\mathrm{Im}_{12}\left(f\left[x + h\left(i_1 + i_2\right)\right]\right)}{h^2} + O\left(h^2\right) \tag{6}$$

The error is of order h^2 and because there is no precision loss due to subtraction (unlike numerical differencing), h can be extremely small, say 10^{-50} (as long as h^2 is representable to machine precision on a computer).

In the case of a scalar function of multiple variables, the second order cross derivative between any two variables, x and y, can be similarly shown to be

$$\frac{\partial^2 f}{\partial x \partial y} \approx \frac{\mathrm{Im}_{12}\left(f\left[x + hi_1, y + hi_2\right]\right)}{h^2} \tag{7}$$

The extension to higher order derivatives is straightforward and accomplished through a convenient property of MCX variables that allow succesive orders to be defined recursively. For details on bicomplex and the higher order MCX variables and derivatives please see References 11 and 12.

Implementation of the MCX Library

The MCX-based sensitivity computation approach is implemented through a recursive class in Matlab and requires overloading of basic arithmetic operations like $(+, -, \cdot, /$ etc..). There are two methods of implementing the MCX library as described in References 11 and 12. The first method deals directly with higher dimensioned numbers such as complex numbers (first order represented as $a+bi_1$) and bicomplex numbers (second order represented as $a+bi_1+ci_2$). The second approach reformulates the complex algebra into a real algebra dealing with larger dimensioned matrices of real numbers.

Considering a preliminary implementation using the Matlab programming language, the matrix approach is pursued currently because all the Matlab inbuilt functions are well suited for real matrices. Analytic Maple expressions have been derived to overload some operators like the division operator. Second order derivatives are implemented as a recursive operation on their first order counterparts. Currently, the MCX library is capable of generating the first and second order derivatives using the matrix formulation. This library can be extended in the future to accommodate derivatives up to the n^{th}-order. Table 1 lists all the operators that have been overloaded as part of the current implementation.

Working Example

In this section a representative example is presented to demonstrate the mechanics of computing derivatives using the MCX library. To evaluate the performance of the new alternative matrix formulation, consider the following scalar function of two variables:

$$f(x, y) = \frac{xy}{x^2 + y^2} \tag{8}$$

Table 1. List of overloaded functions

Function	Overloaded derivative order
basic operators($+, -, *, /$)	1 and 2
logarithmic (log, log1p)	1 and 2
exponential	1 and 2
trigonometric (sin,cos,tan)	1 and 2
inverse trigonometric (asin, acos, atan)	1
hyperbolic (sinh, cossh, tanh)	1
relational operators (lt,gt,eq)	1 and 2
absolute	1 and 2
norm	1 and 2
round	1 and 2
square root	1 and 2
power (integers only)	1 and 2

To compute both the first and second derivatives of the above function with respect to x and y, the function is perturbed in the bicomplex plane in both directions.[12] Hence, the two real inputs are replaced with their perturbed bicomplex counterparts Z_x, Z_y given as follows:

$$Z_x = x + i_1 h + i_2 h \tag{9a}$$

$$Z_y = y + i_1 h + i_2 h \tag{9b}$$

For computing the cross derivatives, we perturb the function separately in each direction for each real input. Hence, for obtaining the cross derivative of the function with respect to x and y, we introduce another pair of bi-complex inputs, Z_{xc} and Z_{yc} given as follows:

$$Z_{xc} = x + i_1 h \tag{10a}$$

$$Z_{yc} = x + i_2 h \tag{10b}$$

Where h is a small perturbation (e.g. 10^{-30}). The matrix representation of the bicomplex numbers Z_x, Z_y,Z_{xc},and Z_{yc} are given as follows:

$$M_{Zx} = \begin{bmatrix} x & -h & h & 0 \\ h & x & 0 & h \\ -h & 0 & x & -h \\ 0 & -h & h & x \end{bmatrix} \tag{11a}$$

$$M_{Zy} = \begin{bmatrix} y & -h & h & 0 \\ h & y & 0 & h \\ -h & 0 & y & -h \\ 0 & -h & h & y \end{bmatrix} \tag{11b}$$

$$M_{Zxc} = \begin{bmatrix} x & -h & 0 & 0 \\ h & x & 0 & 0 \\ 0 & 0 & x & -h \\ 0 & 0 & h & x \end{bmatrix} \tag{11c}$$

$$M_{Zyc} = \begin{bmatrix} y & 0 & h & 0 \\ 0 & y & 0 & h \\ -h & 0 & y & 0 \\ 0 & -h & 0 & y \end{bmatrix} \tag{11d}$$

Note that, unlike typical numerical differencing schemes, h in the complex step formulation can be made extremely small as there are no errors associated with subtraction. Furthermore, these matrices can be recursively formulated for their higher order multicomplex counterparts.

Next, the basic arithmetic operations like $(+, -, \cdot, /$ etc.. $)$ are overloaded in Matlab to make the matrix formulation work without making changes to the user defined functions. The first and second derivatives of the function with respect to x and y are then simply given as follows:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{\text{OutputX}(1,2)}{h} \tag{12a}$$

$$\frac{\partial f(x,y)}{\partial y} \approx \frac{\text{OutputY}(1,2)}{h} \tag{12b}$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} \approx \frac{\text{OutputX}(1,4)}{h^2} \tag{12c}$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} \approx \frac{\text{OutputY}(1,4)}{h^2} \tag{12d}$$

$$\frac{\partial^2 f(x,y)}{\partial x \partial y} \approx \frac{\text{OutputXY}(1,4)}{h^2} \tag{12e}$$

$$\frac{\partial^2 f(x,y)}{\partial y \partial x} \approx \frac{\text{OutputXY}(1,4)}{h^2} \tag{12f}$$

where

$$\text{OutputX} = f(M_{Zx}, y) \tag{13a}$$

$$\text{OutputY} = f(x, M_{Zy}) \tag{13b}$$

$$\text{OutputXY} = f(M_{Zxc}, M_{Zyc}) \tag{13c}$$

It should be noted that the matrices M_{Zx}, M_{Zy}, M_{Zxc}, and M_{Zyc} are treated as a single bicomplex number for the argument of the function $f(x, y)$. Therefore, a matrix inverse (solved analytically) is required when $f(x, y)$ involves a division, otherwise the common matrix and scalar algebra rules apply. The Output(a, b) indicates the a^{th} row and b^{th} column of the resulting matrix. Note that all 6 of the necessary first and second order derivatives are computed using only the 3 function calls given in Equation 13.

Figures 1 and 2 show the error in first and second derivatives of the function $f(x, y)$ when compared to the analytic solution. Both complex and finite difference approaches are evaluated and it can be seen that the finite difference approach looses accuracy for small values of h due to the subtraction errors. The MCX approach gives accurate derivatives up to machine precision for step sizes smaller than 10^{-8}.

In this study, bicomplex numbers are used to compute the Jacobians and Hessians of the measurement functions and of the state equations associated with a SOKF. Considering a state vector of dimension n and taking advantage of symmetries in the Hessian, the bicomplex computation of the Jacobian and Hessian of a single function requires $n(n+1)/2$ function evaluations in the bicomplex plane. Note that a vector function, such as the state equations, is treated as multiple scalar functions. Therefore the Jacobian and Hessian of each component is computed after $n(n+1)/2$ evaluations of the full vector function.

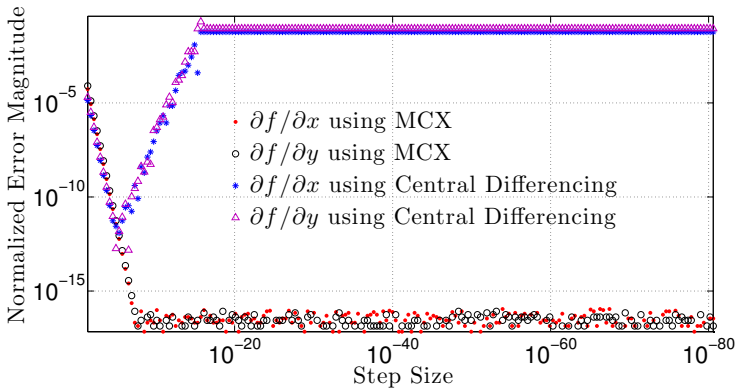


Figure 1. Errors (compared to the analytic solution) in the first derivative of the function including the central difference approach

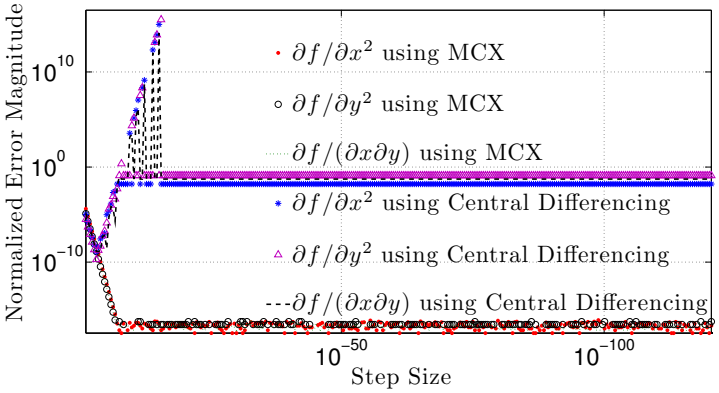


Figure 2. Errors (compared to the analytic solution) in the second derivative of the function including the central difference approach

SECOND ORDER KALMAN FILTER

In this section, equations for the SOKF are presented.[13] As is common for most aerospace filter applications the state propagation is carried out in a continuous manner and the measurement updates are carried out discretely. The dynamics of the state are assumed to be

$$\dot{\mathbf{x}}_k = \mathbf{f}\left(\mathbf{x}_k\right) + \mathbf{w}_k \tag{14}$$

$$\mathbf{w} \sim (\mathbf{0}, \mathbf{Q}) \tag{15}$$

where $\mathbf{w}$ is the system noise, which is normally distributed with a mean of $\mathbf{0}$ and covariance $\mathbf{Q}$.

The measurement is provided by

$$\mathbf{y}_k = \mathbf{h}\left(\mathbf{x}_k\right) + \mathbf{v}_k \tag{16}$$

$$\mathbf{v} \sim (\mathbf{0}, \mathbf{R}) \tag{17}$$

where $\mathbf{v}$ is the system noise, which is normally distributed with a mean of $\mathbf{0}$ and covariance $\mathbf{R}$.

Unlike the EKF, a second order Taylor series expansion is used for the SOKF. This changes the state and covariance propagation equations to

$$\dot{\hat{\mathbf{x}}}_k = \mathbf{f}\left(\hat{\mathbf{x}}_k\right) + \frac{1}{2}\sum_{i=1}^{n} \boldsymbol{\phi}_i Tr\left[\mathbf{f}_{\mathbf{xx}}\mathbf{P}_k^+\right] \tag{18}$$

$$\dot{\mathbf{P}}_k = \mathbf{f}_{\mathbf{x}}\mathbf{P}_k^+ + \mathbf{P}_k^+{\mathbf{f}_{\mathbf{x}}}^T + \mathbf{Q}_k \tag{19}$$

where $\boldsymbol{\phi}_i$ is a unit vector in the i^{th} direction. The Kalman gain can then be calculated.

$$\mathbf{K}_k = \mathbf{P}_k^-{\mathbf{h}_{\mathbf{x}}}^T\left(\mathbf{h}_{\mathbf{x}}\mathbf{P}_k^-{\mathbf{h}_{\mathbf{x}}}^T + \mathbf{R}_k + \boldsymbol{\Lambda}_k\right)^{-1} \tag{20}$$

$$\boldsymbol{\Lambda}_k\left(i,j\right) = \frac{1}{2}Tr\left[\mathbf{D}_{k,i}\mathbf{P}_k^-\mathbf{D}_{k,j}\mathbf{P}_k^-\right] \tag{21}$$

$$\mathbf{D}_{k,i} = \frac{\partial^2 \mathbf{h}_i}{\partial \mathbf{x}^2} \tag{22}$$

The state can then be updated.

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left[\mathbf{y}_k - \mathbf{h}\left(\hat{\mathbf{x}}_k^-\right)\right] - \frac{1}{2}\mathbf{K}_k\sum_{i=1}^{m}\boldsymbol{\phi}_i Tr\left[\mathbf{D}_{k,i}\mathbf{P}_k^-\right] \tag{23}$$

Finally, the covariance can be updated.

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{h}_{\mathbf{x}}\mathbf{P}_k^- \tag{24}$$

The equations for the covariance propagation and update are identical to the equations used in the EKF. It should be noted that the SOKF has the same tuning parameters as the EKF, i.e. the system and the measurement noise. The UKF on the other hand, has three additional tuning parameters and it is very sensitive to them.[14] The use of MCX numbers is important for Equations 18 - 21. These equations require the computation of the Jacobian and Hessian for both the dynamics and measurement equations. While the MCX method is used in the present study, the necessary derivatives are conventionally computed either analytically or via numerical differentiation. Deriving and implementing the analytical derivatives is often tedious, but is fast and accurate at runtime. Numerical differentiation, on the other hand is very easy to implement, but gives inaccurate derivatives due to subtraction error.

GAUSSIAN MIXTURE MODELS

A Gaussian Sum Filter (GSF) uses Gaussian Mixture Models (GMM) in order to improve performance on nonlinear problems. A GMM works under the proposition that any probability distribution can be approximated by using a sum of Gaussian probability distribution functions.[15, 16, 17, 18, 19]

$$p\left(\mathbf{x}\right) = \sum_{i=1}^{N}\alpha_i p_g\left(\mathbf{x};\boldsymbol{\mu}_i,\mathbf{P}_i\right) \tag{25}$$

where N is the total number of Gaussian probability distribution functions, and α_i is a weight, which satisfies the following constraint:

$$\sum_{i=1}^{N}\alpha_i = 1 \tag{26}$$

The weights can be computed by minimizing the L_2 distance between p and p_g,[15,18,19] which turns into a nonlinear optimization problem. Fortunately, libraries for the univariate case have been pre-computed[18,19] that can be simply extended to the multivariate problem. Libraries for $N = 4$ (see Reference 19) and for $N = 3 \ldots 5$ (see Reference 18) have been computed. In this work, the library for $N = 5$ is used for all the implementations.[18] From the available libraries, $N = 5$ is chosen as a responsible compromise between improved performance weighted against the diminishing returns and extra compute burdens of further increasing N.[18]

Table 2. The 5-component univariate splitting library.[18]

Component	w	μ	Standard Deviation
1	0.0763216490701042	−1.689972911128078	0.442255386310084
2	0.2474417859474436	−0.800928383429953	0.442255386310084
3	0.3524731299649044	0	0.442255386310084
4	0.2474417859474436	0.800928383429953	0.442255386310084
5	0.0763216490701042	1.689972911128078	0.442255386310084

The same univariate splitting library can be utilized to split multivariate distributions. This method[18,19] applies the univariate splitting library along an arbitrary direction of the square-root factor of the covariance matrix. In order to give physical significance to the direction along which the univariate splitting library is used, spectral factorization of $\mathbf{P}$ is carried out using its eigenvalues and eigenvectors. An example is presented of the splitting of a 2-dimensional Gaussian distribution with mean and covariance found in Equation 27. The covariance ellipses of the resulting split distribution along x_1 and x_2 can be seen in Figures 3(a) and 3(b), respectively.

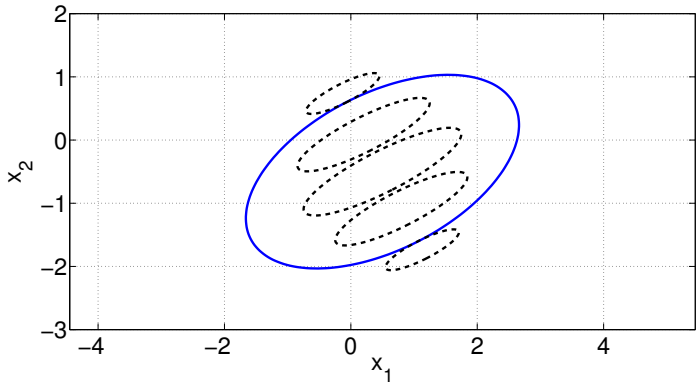
$$\boldsymbol{\mu} = \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} \tag{27a}$$

$$\mathbf{P} = \begin{bmatrix} 2.05 & 0.7 \\ 0.7 & 1.03 \end{bmatrix} \tag{27b}$$

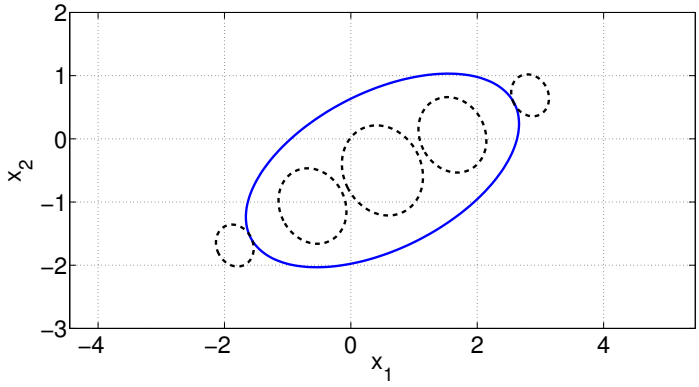
SECOND-ORDER KALMAN GAUSSIAN SUM FILTER

The Second-Order Kalman Gaussian Sum Filter (SOKGSF) is presented in this section, along with some simulations that showcase its estimation properties. A GSF simply runs a bank of filters, Kalman-based in this case, on each of the Gaussian mixtures. The filters can be the EKF (EKGSF), SOKF (SOKGSF), UKF (UKGSF) etc. Therefore, the equations are the standard ones available in the literature, and in the case of SOKF Eqs. (14) - (24). When using a GSF, the weights of each of the Gaussian mixtures are meant to be updated after each measurement update. The weights can also be adapted after the propagation phase.[15,20] However, in this study the propagation phase updates are not included because the extra phase of updating weights is claimed to not improve performance.[15] The weight updates are discussed more in the following section.

The method for testing the filters is a 1 Degree Of Freedom (DOF) problem that considers a vertically falling body with a constant ballistic coefficient. The altitude and velocity of the body are estimated using range measurements from a stationary radar. This particular problem, due to its nonlinearities, is a common benchmark to test filter performance.[2,13,8]



(a) Split along x_1



(b) Split along x_2

Figure 3. Multispectral split of a multivariate pdf with the ellipses being the 65% confidence regions. Solid ellipse is for the original distribution and the dashed ellipses are the split distributions.

Filter Equations

As mentioned earlier, a GSF results in a bank of filters running in parallel with consolidation occurring when the weights are updated. This consolidation can occur either after propagation and after measurement updates, or only after measurement updates. There are also methods of combining a few mixtures when they are either very close together, or the weights are below a certain threshold. However, in this study, for simplicity the number of mixtures is not changed throughout the estimation problem.

The standard EKF or SOKF equations are used on the individual mixture elements. After the measurement update on each of the mixture elements, the weights are updated according to the following equation:[16]

$$\alpha_i = \frac{\alpha_i^{'}\beta_i}{\sum_{j=1}^{N}\alpha_j^{'}\beta_j^{'}} \tag{28}$$

where N is the total number of weights, $\alpha_i^{'}$ is the pre-update weight, and α_i is the post-update weight. The term β_i is[16]

$$\beta_i \equiv p_g\left(\mathbf{z} - \mathbf{h}\left(\boldsymbol{\mu}_i\right), \mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^T + \mathbf{R}\right) \tag{29}$$

The above β_i is only the case for an EKF. However, if the SOKF is used, β_i instead becomes

$$\beta_i \equiv p_g\left(\mathbf{z} - \hat{\mathbf{z}}_i, \mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^T + \mathbf{R} + \boldsymbol{\Lambda}\right) \tag{30}$$

where $\boldsymbol{\Lambda}$ is defined in Equation 21, $\mathbf{h}(\boldsymbol{\mu}_i)$ is the measurement predicted by the mean, $\mathbf{P}_i$ is the covariance, and $\mathbf{H}_i$ is $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\Big|_{\boldsymbol{\mu}_i}$ of the i^{th} mixture element.

The derivation for the β_i of the EKF is carried out using Bayes' rule.[16,18] It is simply the probability of the actual measurement, based on a Gaussian probability distribution defined by the measurement estimate as the mean and the *a priori* state covariance transformed through the measurement equation as the covariance. The difference between the β_i for the SOKF and the EKF versions is due to the second order term present in the covariance propagation found in Equations 20 - 22.

Athans Problem Simulation and Results

The problem chosen to test the filters is a 1 DOF problem of a falling body. This example is common for testing filtering algorithms and can be seen in Figure 4. The state is 3-dimensional where x_1 is the altitude, x_2 is the downward velocity, and x_3 is a ballistic parameter. The dynamics are given in Equation 31. The measurement of the range from the radar to the falling body is calculated using Equation 32.

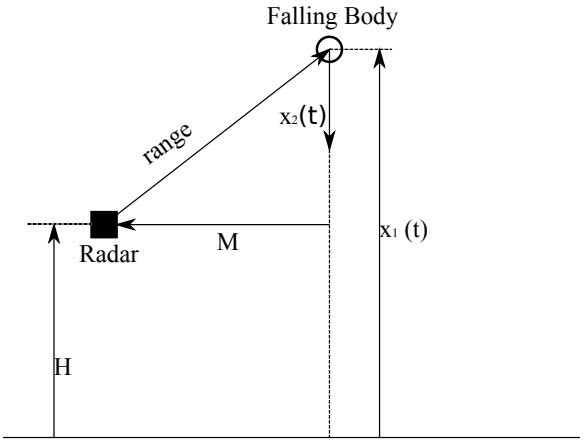


Figure 4. A 2D falling body problem[2]

$$\dot{x}_1 = -x_2 \tag{31a}$$

$$\dot{x}_2 = -\exp\left(-\gamma x_1\right) x_2^2 x_3 \tag{31b}$$

$$\dot{x}_3 = 0 \tag{31c}$$

$$h = \sqrt{M^2 + \left(x_1 - H\right)^2} \tag{32}$$

The truth trajectory is simulated using a very stringent tolerance (abserr = 10^{-14}, relerr = 10^{-14}) with the variable step-size RK4-5 Matlab integrator.[21] The constants required to simulate the trajectory are found in the Table 3.

Table 3. Constants required to run the simulation

H	M	γ
$100,000$ ft	$100,000$ ft	5×10^{-5} ft^{-1}

The various filters are evaluated on the problem using the initial conditions found in Table 4. The state estimate and the covariance are propagated in time using a custom RK4 fixed step-size integrator and a time step-size of $1/50$ seconds. The first measurement is received after 5 s with a frequency of 1 Hz and a measurement noise of $10,000$ ft^2. The UKF is run with the default tuning parameters of $\alpha = 0.003$, $\beta = 2$, and $\kappa = 0$.[9]

Table 4. Constants required to run the simulation

	$\mathbf{x}_0$	$\hat{\mathbf{x}}_0$	Initial Variance
x_1	$300,000$ ft	$300,000$ ft	10^6 ft^2
x_2	$20,000$ ft/s	$20,000$ ft/s	4×10^6 ft^2/s^2
x_3	10^{-3}	3×10^{-5}	10^{-4} ft^{-1}

The CPU time required for the various filters can be seen in Table 5.

Table 5. CPU time for filter run in Matlab

Filter	Derivative Type	CPU Time [s]
EKF	Analytical	0.5
EKF	Numerical	0.8
EKF	MCX	5.8
UKF	N/A	1.5
SOKF	Analytical	1.0
SOKF	Numerical	6.1
SOKF	MCX	70.5
EKGSF	Analytical	2.3
EKGSF	Numerical	3.6
EKGSF	MCX	30.3
SOKGSF	Analytical	4.7
SOKGSF	Numerical	25.9
SOKGSF	MCX	302.8

The filters are compared by the error in the altitude estimate. For each filter, 100 cases were run with different noise profiles. The mean error in the altitude, standard deviations, the medians, and the number of diverged cases can be seen in Table 6. A run is considered to be diverged if the error

in the altitude estimate becomes more than 10^4 ft at any time during the simulation. The altitude error profile, along with the $3-\sigma$ bounds can be seen for the various filters in Figure 5 - 9

Table 6. Errors in the altitude estimation.

	Mean (ft)	Standard Deviation (ft)	Median (ft)	# Diverged cases
EKF	284.7	188.6	225.1	1
UKF	223.9	64.6	217.6	1
UKF_b	216.1	61.0	210.0	2
SOKF	233.5	69.3	225.6	1
EKGSF	242.3	149.0	200.0	2
SOKGSF	219.3	73.4	211.2	2

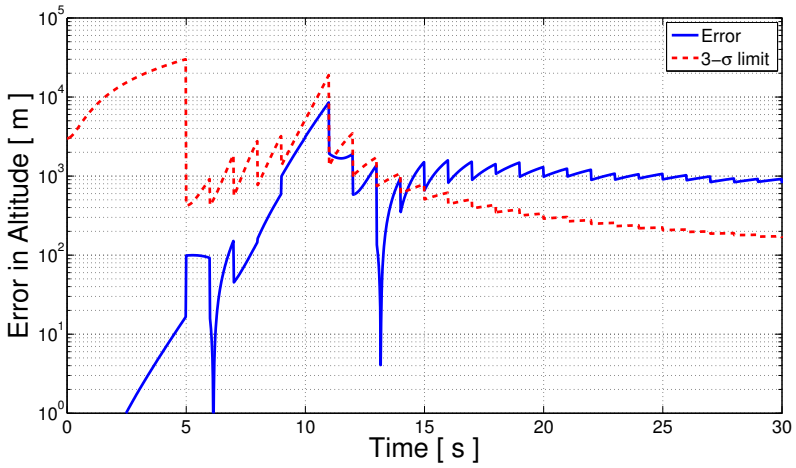


Figure 5. The altitude error using the EKF

To show that the UKF is sensitive to changes in the tuning parameters, it is run again with another set of tuning parameters ($\alpha = 2$, $\beta = 0$, and $\kappa = 2$).[14] This run is called the UKF_b, and the results of this filter can also be seen in Table 6. The difference in the mean error of the filters can be seen in Figure 10. This mean error has a mean of 20.2 ft, and a standard deviation of 17.4 ft, which is almost 10% of the mean error of either of the UKFs.

The first order EKF performs the worst, as expected. The higher order filters perform better than the EKF as they are able to better capture the nonlinearities of the dynamics and the measurement model. The slight advantage of the UKF over the SOKF may be explained by the claim that the UKF captures nonlinearities to third order in some cases.[4] The EKGSF performs better than the EKF, but fails to perform as well as the other second order filters. The SOKGSF performs better than the SOKF and equal to the UKF. The mean error is lower, although this lower error is compensated by the increase in the standard deviation. However, the increase may be more accurate to the true uncertainty so it is not necessarily an inferior result.

In this work, a very simple version of the GMM-based filters is implemented. The original estimate of the covariance is split into 5 mixtures, and these are the only ones that are propagated,

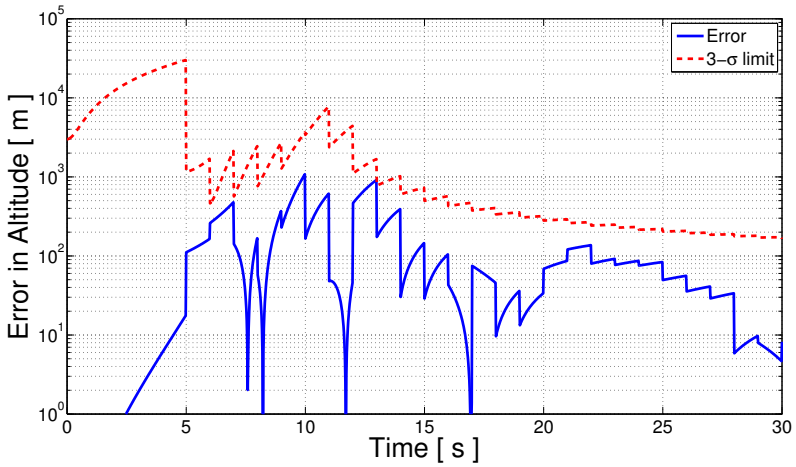


Figure 6. The altitude error using the UKF

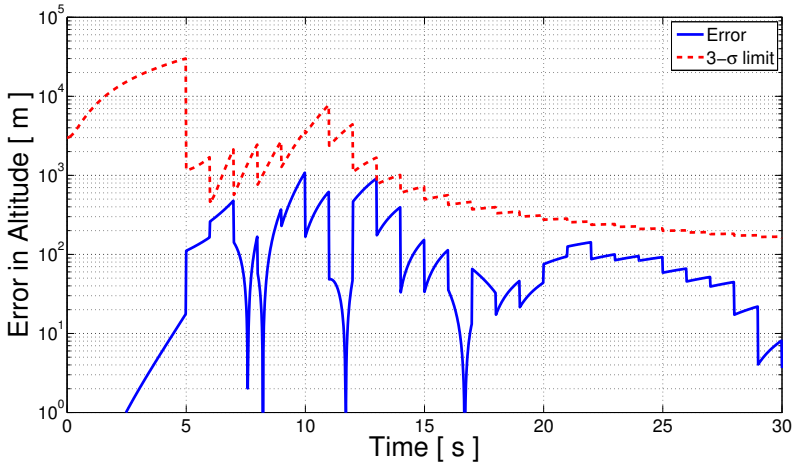


Figure 7. The altitude error using the SOKF

updated, and eliminated if deemed necessary. The only check is to see if the weight of a mixture is lower than a certain threshold, in which case the corresponding mixture is removed in order for the filter to be more computationally efficient. Another important inspection not currently considered is a check to require that none of the mixtures diverge. The mixtures that have a very small weight are the furthest away from the mean of the combined GMM. Therefore, the aforementioned checks will often remove the same mixtures.

Both the aforementioned checks often result in checking for the same case where the mixtures that have a very small weight are the furthest away from the mean of the combined GMM. The SOKGSF can be improved by using more mixtures, although this would increase the computational load. The SOKGSF can also be made more complex by combining components that are very close together, and splitting components that have a relatively larger weight.

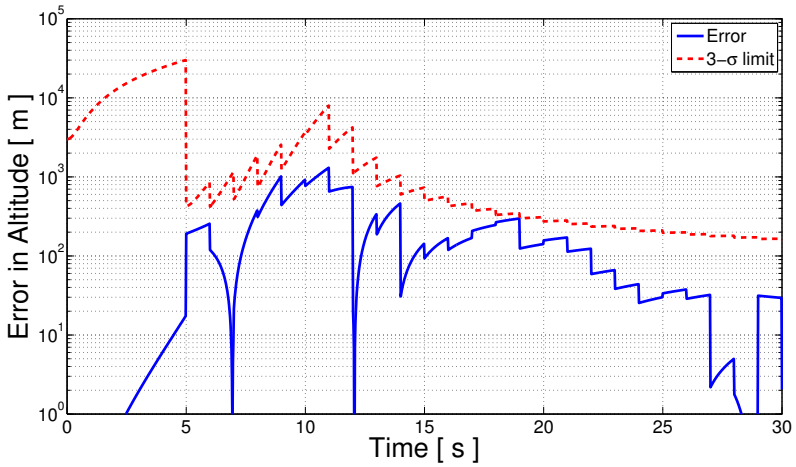


Figure 8. The altitude error using the EKGSF

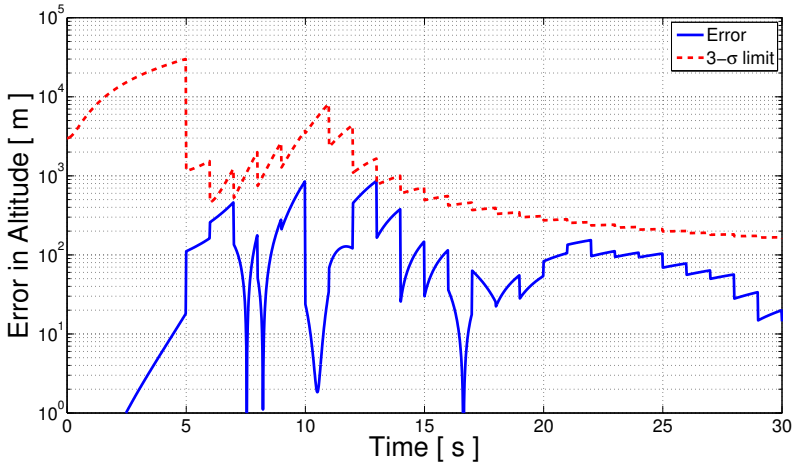


Figure 9. The altitude error using the SOKGSF

CONCLUSIONS

The SOKF, although derived many years ago, has not been used often due to the difficulty in obtaining the Hessians of the measurements and state dynamics. The recently developed MCX technique alleviates this issue, computing high order sensitivities accurately and transparently to the user. The performance of the SOKF in terms of accuracy is shown to be on par with the UKF. However, the SOKF is easier to to implement, and has fewer tuning parameters than the UKF. Combining the SOKF with GMM results in a filter that is better than the original SOKF due to it being analogous to a bank of filters running in parallel. In this work, a very simple version of the GMM-based filters has been implemented. These simple GMM filters have been shown to be effective, although future work includes a variety of more sophisticated GMM-based filters as they seem very promising for space tracking applications.[15,22,23]

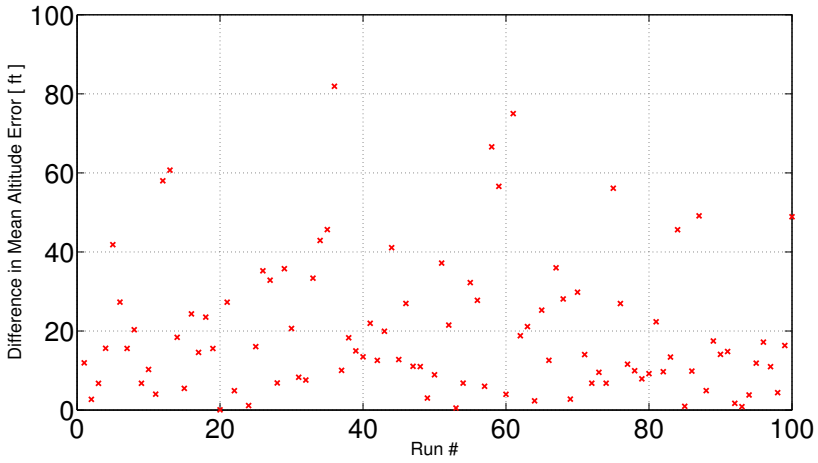


Figure 10. The difference in mean error in two UKFs due to tuning parameters

The MCX derivatives, in this work, are utilized to build a user friendly SOKF in the sense that the required first and second order derivatives are computed accurately and automatically. Beyond the current scope, the MCX numbers can be implemented for any application that requires the computation of higher order derivatives. The SOKF shown here only requires derivatives and therefore, captures the nonlinearities up to the second order. In the future the Taylor series approximation can be further expanded to higher orders. The associated higher order filters will better capture the nonlinearities of the system, potentially surpassing the performance of the UKF and other sigma point filters. The extra derivative tensors can still be computed to machine precision using the MCX numbers. The main drawback of the MCX currently is the computation speed. Presently, the MCX overloading library necessary for computing the necessary derivatives has been implemented in Matlab. The speed can be improved in Matlab by further vectorizing the code. In future works, the toolbox will be ported to a compiled language such as C++, or Fortran where overloading is natively handled more efficiently. For access to the Matlab or other MCX libraries, please contact the authors.

ACKNOWLEDGMENT

This project was supported in part by the NASA Small Business Innovation Research (SBIR) program (Grant NNX11CE08P) under a subcontract from Emergent Space Technologies, Inc.

REFERENCES

[1] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, Vol. 82, No. Series D, 1960, pp. 35–45.

[2] M. Athans, R. P. Wishner, and A. Bertolini, "Suboptimal State Estimation for Continuous-Time Nonlinear Systems from Discrete Noisy Measurements," *IEEE Transactions on Automatic Control*, Vol. AC-13, No. 5, 1968, pp. 504–514.

[3] A. Gelb, *Applied Optimal Estimation*. MIT Press, Cambridge, Massachusetts, 1974.

[4] S. J. Julier and J. K. Uhlman, "A New Extension of the Kalman Filter to Non Linear Systems," *Proceedings of the Internaitonal Symposium on Aerospace/Defense Sensing, Simulation and Controls, Orlando, FL*, 1997.

[5] R. v. d. Merwe and E. A. Wan, "The Square-Root Unscented Kalman Filter for State and Parameter-Estimation," *2001 IEEE International Conference on In Acoustics, Speech, and Signal Processing*, Vol. 6, 2001, pp. 3461–3464.

[6] E. A. Wan and R. V. d. Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," *Proceedings of IEEE Symposium on Adaptive Systems for Signal Processing, Communications and Control*, 2000.

[7] M. Nøgaard, N. K. Poulsen, and O. Ravn, "Advances in Derivatice-Free State Estimation for Nonlinear Systems," Tech. Rep. IMM-REP-1998-15, Technical University of Denmark, October 2004.

[8] K. L. Lai and J. L. Crassidis, "Second-Order Divided-Difference Filter Using a Generalized Complex-Step Approximation Second-Order Divided-Difference Filter Using a Generalized Complex-Step Approximation," *AIAA Guidance, Navigation, and Control Conference Proceedings*, 2007.

[9] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, Vol. 92, mar 2004, pp. 401–422.

[10] W. Squire and G. Trapp, "Using Complex Variables to Estimate Derivatives of Real Functions," *SIAM Review*, Vol. 40, No. 1, 1998, pp. 110–112.

[11] G. Lantoine, R. P. Russell, and T. Dargent, "Using Multi-Complex Variables for Automatic Computation of High-Order Derivatives," *Paper AAS 10-218, AAS/AIAA Space Flight Mechanics Meeting, San Diego, CA*, February 2010.

[12] G. Lantoine, R. P. Russell, and T. Dargent, "Using Multi-Complex Variables for Automatic Computation of High-Order Derivatives," *ACM Transactions on Mathematical Software*, accepted June 2011.

[13] D. Simon, *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[14] R. Turner and C. E. Rasmussen, "Model based learning of sigma points in unscented Kalman filtering," *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*, 29 2010-sept. 1 2010, pp. 178–183.

[15] J. T. Horwood, N. D. Aragon, and A. B. Poore, "Gaussian Sum Filters for Space Surveillance: Theory and Simulations," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 6, 2011, pp. 1839–1851.

[16] D. L. Alspach and H. W. Sorenson, "Nonlinear Bayesian Estimation Using Gaussian Sum Approximations," *IEEE Transactions on Automatic Control*, Vol. AC-17, No. 4, 1972, pp. 439–448.

[17] H. Tanizaki, *Nonlinear Filters: Estimation and Applications*. Springer-Verlag, 1993.

[18] K. J. DeMars, *Nonlinear Orbit Uncertainty Prediction and Rectification for Space Situational Awareness*. PhD thesis, The University of Texas at Austin, 2010.

[19] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck, "On entropy approximation for Gaussian mixture random vectors," *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, 2008, pp. 181–188, 10.1109/MFI.2008.4648062.

[20] G. Terejanu, P. Singla, T. Singh, and P. D. Scott, "Adaptive Gaussian Sum Filter for Nonlinear Bayesian Estimation," *IEEE Transactions on Automatic Control*, Vol. 56, September 2011, pp. 2151–2156.

[21] L. F. Shampine and M. W. Reichelt, "The MATLAB ODE Suite," *SIAM Journal of Scientific Computing*, Vol. 18, No. 1, 1997, pp. 1–22.

[22] K. J. DeMars, R. H. Bishop, and M. K. Jah, "Space Object Tracking In The Presence Of Attitude-Dependent Solar Radiation Pressure Effects," *Girdwood, Alaska, AAS/AIAA Astrodynamics Specialist Conference, AAS 11-582*, August 2011.

[23] K. DeMars, R. Bishop, and M. Jah, "A Splitting Gaussian Mixture Method for the Propagation of Uncertainty in Orbital Mechanics," *New Orleans, Louisiana, 21st AAS/AIAA Space Flight Mechanics Meeting*, February 2011.