An Efficient Method for Generating Points Uniformly Distributed in Hyperellipsoids

Jean Dezert and Christian Musso

ONERA/DTIM, 29 Av. Division Leclerc 92320 Châtillon, France

ABSTRACT

In this paper, we consider the problem of generating efficiently random points uniformly distributed in hyperellipsoid. Such kind of problem arises frequently in Monte Carlo simulations for the performance evaluation of multitarget tracking algorithm in cluttered environment since the false alarms are usually assumed to be uniformly distributed in the (ellipsoidal) validation gate. We present here a more efficient algorithm for solving this problem which outperforms all existing methods both in term of reduction of computational loads and in term of quality of uniformity obtained. The efficiency and the feasability of our new method is demonstrated by several simulation results and comparisons. Moreover, the Matlab™ source code of algorithm is also provided for convenience.

1. INTRODUCTION

In the Monte Carlo simulations for the study and design of multitarget tracking algorithms [2,3], one needs frequently to generate false alarms (FA) in target validation gates defined by hyperellipsoids in measurement space computed from predicted target measurement and covariance of measurement innovation. False alarms are usually supposed to be independent and uniformly distributed in validation gates. During many years, the only inefficient method for generating such random points [1] was to generate points in the minimal hypercube containing hyperellipsoid, and then sort and keep points which have been drawn in the hyperellipsoid based on a Mahalanobis distance test. This method which can be used whenever measurement space dimension and spatial density of false alarms are low, become very inefficient with the growth of FA spatial density and measurement space dimension because of the exponential rejection ratio which will be presented in section 2.3.

To overcome this major drawback, X.R. Li has been the first one (to the knowledge of the authors) to propose in 1992 [12] a new algorithm, for generating points uniformly distributed in hyperellipsoids. In 1999, T.J. Ho and M. Farooq have however pointed out in [10] an obstacle in the practical use of Li's approach. They have then proposed an improved approach (referred here as HF algorithm; HF standing for initials of authors) based on the orthogonal factorization of covariance matrix $\mathbf{S}$ which avoids the indefinite number of iterations occuring within Li's algorithm. It is worthwhile to note that both approaches are based on computation of eigenvalues of matrix $\mathbf{S}$. This requirement is time consuming (high computation burden) when measurement space dimension becomes high.

In recent tracking developments, authors have tested intensively the HF algorithm and have discovered the poor performances of this algorithm in term of spatial uniformity of random points generated in validation gates. A presentation of HF algorithm results will be detailed in the sequel. To overcome this major drawback, we propose a new fast, efficient and reliable algorithm for generating directly random points really uniformly distributed in hyperellipsoid which has the following two important properties : its complexity is $O(n^3)$ (n being the measurement space dimension), and it does not require the computation of eigenvalues of matrix $\mathbf{S}^{-1}$ and $\mathbf{S}^{-1}$ itself as in previous existing methods. The new method proposed in this paper follows exactly the same assumptions as in [12,10]: 1) the number of false measurements to be generated can be described by a suitable Poisson model; 2) the false measurements are uniformly distributed in validation gate and are independent from scan to scan.

Authors can be contacted at following E-mail addresses: Jean.Dezert@onera.fr or Christian.Musso@onera.fr
Matlab™ is a trademark of MathWorks, Inc.

2. PRELIMINARY

2.1. Validation of measurements

In target tracking, a validation gate V is used for eliminating sensor measurements which have small probability to belong to target. The measurements falling in the gate are said to be validated. Let $\hat{\mathbf{x}}(k|k \Leftrightarrow 1)$ be the one step predicted state vector of a given target at time k and $\mathbf{P}(k|k \Leftrightarrow 1)$ the corresponding one step prediction covariance matrix of prediction error $\mathbf{x}(k) \Leftrightarrow \hat{\mathbf{x}}(k|k \Leftrightarrow 1)$. $\mathbf{x}(k)$ is the true (unknown) state vector of target at time k with dimension n_x. Given all information about the target up to k, we assume the probability density function (pdf) $p(\mathbf{x})$ to be Gaussian with mean $\hat{\mathbf{x}}(k|k \Leftrightarrow 1)$ and covariance $\mathbf{P}(k|k \Leftrightarrow 1)$, that is $p(\mathbf{x}(k)) = \mathcal{N}(\mathbf{x}(k); \hat{\mathbf{x}}(k|k \Leftrightarrow 1), \mathbf{P}(k|k \Leftrightarrow 1))$. If the observation model $\mathbf{z}(k) = \mathbf{h}[k, \mathbf{x}(k), \mathbf{w}(k)]$ is linear with additive zero-mean white Gaussian noise $\mathbf{w}(k)$ with covariance $\mathbf{R}(k)$ (i.e. $\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{w}(k)$), then the innovation $\tilde{\mathbf{z}}(k)$ (i.e. difference between measurement $\mathbf{z}(k)$ and its prediction $\hat{\mathbf{z}}(k|k \Leftrightarrow 1) = \mathbf{H}(k)\hat{\mathbf{x}}(k|k \Leftrightarrow 1)$) is Gaussian with zero mean and covariance $\mathbf{S}(k) = \mathbf{H}(k)\mathbf{P}(k|k \Leftrightarrow 1)\mathbf{H}'(k) + \mathbf{R}(k)$ where superscript $'$ denotes the transposition [2]. Therefore, the pdf of true target measurement $\mathbf{z}(k)$ is given by [14]

$$p(\mathbf{z}(k)) = \mathcal{N}(\mathbf{z}(k); \hat{\mathbf{z}}(k|k \Leftrightarrow 1), \mathbf{S}(k)) = |2\pi\mathbf{S}(k)|^{-1/2} e^{-\frac{1}{2}[\mathbf{z}(k)-\hat{\mathbf{z}}(k|k-1)]'\mathbf{S}^{-1}(k)[\mathbf{z}(k)-\hat{\mathbf{z}}(k|k-1)]} \tag{1}$$

or equivalently

$$p(\tilde{\mathbf{z}}(k)) = \mathcal{N}(\tilde{\mathbf{z}}(k); \mathbf{0}, \mathbf{S}(k)) = |2\pi\mathbf{S}(k)|^{-1/2} e^{-\frac{1}{2}\tilde{\mathbf{z}}'(k)\mathbf{S}^{-1}(k)\tilde{\mathbf{z}}(k)} \tag{2}$$

where $\mathbf{z}(k)$ and $\tilde{\mathbf{z}}(k)$ are vectors of dimension n_z, $\mathbf{S}(k)$ is a real symmetric and positive definite matrix of size $n_z \times n_z$ and $\mathbf{0}$ is the null vector ($[0, \dots, 0]'$) of size n_z. For notation convenience and brevity, the time index k is from now omitted in the following.

The density function (1) is constant whenever the quadratic form $\epsilon \triangleq [\mathbf{z} \Leftrightarrow \hat{\mathbf{z}}]'\mathbf{S}^{-1}[\mathbf{z} \Leftrightarrow \hat{\mathbf{z}}]$ in the exponent is, so that it is constant on the ellipsoid (called hyperellipsoid if $n_z > 3$) defined by

$$[\mathbf{z} \Leftrightarrow \hat{\mathbf{z}}]'\mathbf{S}^{-1}[\mathbf{z} \Leftrightarrow \hat{\mathbf{z}}] = \gamma \tag{3}$$

in $\mathbb{R}^{n_z}$ for every $\gamma > 0$. ϵ is called the Mahalanobis distance (or statistical distance) of the measurement $\mathbf{z}$ with respect to its prediction $\hat{\mathbf{z}}$ and is also referred as the NIS (Normalized Innovation Squared) in [2]. This ellipsoid has center $\hat{\mathbf{z}}$, while $\mathbf{S}$ determines its shape and orientation. Since innovation $\tilde{\mathbf{z}}$ is a zero-mean Gaussian random variable with dimension n_z, ϵ is a $\chi^2_{n_z}$ random variable (see theorem 1.4.1 of [14] for proof). The pdf of ϵ is then given by [16] (p. 187)

$$p(\epsilon) = \begin{cases} 0 & \text{for } \epsilon < 0 \\ \frac{1}{2^{n_z/2}\Gamma(n_z/2)} \epsilon^{\frac{1}{2}n_z - 1} e^{-\frac{1}{2}\epsilon} & \text{for } \epsilon \geq 0 \end{cases} \tag{4}$$

where $\Gamma(.)$ is the Gamma function defined for $n > 0$ by $\Gamma(n) = \int_0^\infty t^{n-1}e^{-t}dt$ which follows the well known recurrence formulae $\Gamma(n+1) = n\Gamma(n)$ and $\Gamma(n+1) = n!$ if $n = 0, 1, 2, \dots$. One has also $\Gamma(2) = \Gamma(1) = 1$ and $\Gamma(\frac{1}{2}) = \sqrt{\pi}$ and the following recurrence formula $\Gamma(n) = \Gamma(n+1)/n$ holds when $n < 0$.

The validation (gating) of sensor measurements is obtained by choosing the threshold γ in such a way that the probability of true measurement falling in the validation gate $\mathcal{V}(\gamma)$, defined by

$$\mathcal{V}^{n_z}(\gamma) \triangleq \{\mathbf{z} : [\mathbf{z} \Leftrightarrow \hat{\mathbf{z}}]'\mathbf{S}^{-1}[\mathbf{z} \Leftrightarrow \hat{\mathbf{z}}] \leq \gamma\} \tag{5}$$

corresponds to a given gating probability P_g. The gating threshold γ and P_g are related through the following relationship

$$P_g = Pr\{\mathbf{z} \in \mathcal{V}^{n_z}(\gamma)\} = Pr\{\chi^2_{n_z} \leq \gamma\} = \int_0^\gamma p(\epsilon)d\epsilon = \frac{1}{2^{n_z/2}\Gamma(n_z/2)} \int_0^\gamma \epsilon^{(n_z/2)-1} e^{-\epsilon/2} d\epsilon \tag{6}$$

Under Matlab programming environment (with statistics toolbox), the threshold γ can be easily computed using the command `gamma_threshold=chi2inv(Pg,nz)`. The square root $g = \sqrt{\gamma}$ is usually called the ''number of sigmas" (standard deviations) of the gate [1]. The semi-axis of ellipsoid $\mathcal{V}^{n_z}(\gamma)$ are the square roots of diagonal terms of $\gamma\mathbf{S}$. In summary the validation test $T(\mathbf{z})$ is formally defined by

$$T(\mathbf{z}) = \begin{cases} 1 & \text{if } \tilde{\mathbf{z}}'\mathbf{S}^{-1}\tilde{\mathbf{z}} \leq \gamma \quad \Rightarrow \quad \mathbf{z} \text{ is validated} \\ 0 & \text{if } \tilde{\mathbf{z}}'\mathbf{S}^{-1}\tilde{\mathbf{z}} > \gamma \quad \Rightarrow \quad \mathbf{z} \text{ is discarded} \end{cases} \tag{7}$$

In most of tracking applications, the observation of the targets is quite often difficult because of small target detection probabilities, bad conditions of observations due to cluttered environment and the limited quality of sensors of tracking system. In many practical tracking problems, one has therefore to take into account the presence of false measurements in the validation gate. For performance evaluation of realistic tracking algorithms based on Monte Carlo simulations, we are then frequently faced to the problem of generation of false alarms in validation gates. The usual assumption made is to consider the false alarms uniformly distributed in validation and independent from scan to scan. The development of our new algorithm for generating random points uniformly distributed in an hyperellipsoid allows herefater to efficiently solve this problem with a minimal computational burden.

2.2. Volume of an hyperellipsoid

The volume $V^{n_z}(\gamma)$ of an hyperellipsoid $\mathcal{V}^{n_z}(\gamma)$ is defined by

$$V^{n_z}(\gamma) = \int_{\tilde{\mathbf{z}}'\mathbf{S}^{-1}\tilde{\mathbf{z}} \leq \gamma} d\tilde{\mathbf{z}} \tag{8}$$

Since $\mathbf{S}$ is a real symmetric positive definite matrix, there exists [13] a non singular linear transformation $\mathbf{T}$ such that $\mathbf{S} = \mathbf{T}\mathbf{T}'$. $\mathbf{T}$ is called square root of $\mathbf{S}$. Such decomposition is not unique but the Cholesky factorization allows to get easily an useful solution (in $O(n^3)$ complexity) for $\mathbf{S}^{1/2} = \mathbf{T}$. Details about implementation of Cholesky factorization can be found in [2] and [5]. From this factorization, one has

$$\mathbf{S}^{-1} = (\mathbf{T}\mathbf{T}')^{-1} = \mathbf{T}'^{-1}\mathbf{T}^{-1} \Leftrightarrow \mathbf{T}'\mathbf{S}^{-1}\mathbf{T} = \mathbf{I} \tag{9}$$

In order to compute $V^{n_z}(\gamma)$, one has to introduce the following variable transformation $\mathbf{y} = \mathbf{T}^{-1}\tilde{\mathbf{z}}$. Then

$$\tilde{\mathbf{z}}'\mathbf{S}^{-1}\tilde{\mathbf{z}} = (\mathbf{T}\mathbf{y})'\mathbf{S}^{-1}(\mathbf{T}\mathbf{y}) = \mathbf{y}(\mathbf{T}'\mathbf{S}^{-1}\mathbf{T})\mathbf{y} = \mathbf{y}'\mathbf{y} \tag{10}$$

It follows

$$V^{n_z}(\gamma) = \int_{\mathbf{y}'\mathbf{y} \leq \gamma} J d\mathbf{y} \tag{11}$$

where $J = \mid \frac{\partial \tilde{\mathbf{z}}}{\partial \mathbf{y}} \mid = \mid \mathbf{T} \mid$ is the Jacobian of the transformation from the $\tilde{\mathbf{z}}$ variable to the $\mathbf{y}$ variable. Since $\mathbf{S}^{-1} = \mathbf{T}'^{-1}\mathbf{T}^{-1}$, then $\mid \mathbf{S}^{-1} \mid = \mid \mathbf{T} \mid^{-2}$ and therefore $J = \mid \mathbf{T} \mid = 1/\sqrt{|\mathbf{S}^{-1}|} = \sqrt{|\mathbf{S}(k)|}$. By using the generalized spherical coordinate change of variable [13], one has

$$V^{n_z}(\gamma) = \sqrt{|\mathbf{S}|} \int_0^{\sqrt{\gamma}} \int_0^{2\pi} \underbrace{\int_0^{\pi} \ldots \int_0^{\pi}}_{n_z-2} r^{n_z-1} \Big(\prod_{k=1}^{n_z-2} \sin^{n_z-1-k} \Phi_k \Big) dr d\theta d\Phi_1 \ldots d\Phi_{n_z-2} \tag{12}$$

which can be written as

$$V^{n_z}(\gamma) = \sqrt{|\mathbf{S}|}\Big(\int_0^{\sqrt{\gamma}} r^{n_z-1}dr\Big)\Big(\int_0^{2\pi} d\theta\Big)\prod_{k=1}^{n_z-2}\int_0^{\pi} \sin^{n_z-1-k}\Phi_k d\Phi_k \tag{13}$$

But

$$\int_0^{\pi} \sin^{n_z-1-k}\Phi_k d\Phi_k = B((n_z \Leftrightarrow k)/2, 1/2) = \frac{\Gamma(\frac{1}{2}(n_z \Leftrightarrow k))\Gamma(\frac{1}{2})}{\Gamma(\frac{1}{2}(n_z \Leftrightarrow k+1))} \tag{14}$$

where $B(.,.)$ is the Beta function and $\Gamma(.)$ the Gamma function and hence

$$\prod_{k=1}^{n_z-2}\int_0^{\pi} \sin^{n_z-1-k}\Phi_k d\Phi_k = \frac{\Gamma^{n_z-2}(\frac{1}{2})}{\Gamma(\frac{n_z}{2})} = \frac{\pi^{(n_z-1)/2}}{\Gamma(\frac{n_z}{2})} \tag{15}$$

By reporting previous expressions into (13), one gets

$$V^{n_z}(\gamma) = \sqrt{|\mathbf{S}|} \times \frac{\gamma^{\frac{n_z}{2}}}{n_z} \times 2\pi \times \frac{\pi^{(n_z-1)/2}}{\Gamma(\frac{n_z}{2})} \tag{16}$$

which can be finally expressed as

$$V^{n_z}(\gamma) = \frac{(\pi\gamma)^{\frac{n_z}{2}}\sqrt{|\mathbf{S}|}}{\Gamma(\frac{n_z}{2}+1)} = c_{n_z}\sqrt{|\mathbf{S}|}\gamma^{n_z/2} \tag{17}$$

where coefficient c_{n_z} is given by

$$c_{n_z} = \frac{\pi^{n_z/2}}{\Gamma(\frac{n_z}{2}+1)} = \begin{cases} \frac{\pi^{n_z/2}}{(n_z/2)!} & \text{for } n_z \text{ even} \\ \frac{2^{n_z+1}(n_z+1/2)!}{(n_z+1)!}\pi^{(n_z-1)/2} & \text{for } n_z \text{ odd} \end{cases} \tag{18}$$

c_{n_z} can be easily obtained under Matlab by using the command `cnz=pi^(nz/2)/gamma(1+nz/2)`.

The volume $V_s^{n_z}(\gamma)$ of an n_z-dimensional hypersphere of radius $\sqrt{\gamma}$ is therefore obtained by choosing $\mathbf{S} = \mathbf{I}$ (i.e. the identity matrix of size $n_z \times n_z$). One gets directly from (17)

$$V_s^{n_z}(\gamma) = \frac{(\pi\gamma)^{\frac{n_z}{2}}}{\Gamma(\frac{n_z}{2}+1)} \tag{19}$$

The volume $V_c^{n_z}(\gamma)$ of minimal hypercube containing this hypersphere is given by

$$V_c^{n_z}(\gamma) = (2\sqrt{\gamma})^{n_z} \tag{20}$$

Hence, the ratio $r = V_c^{n_z}(\gamma)/V_s^{n_z}(\gamma)$ is equal to $(4/\pi)^{n_z/2}\Gamma(\frac{n_z}{2}+1)$. By using Stirling development of $\Gamma(\frac{n_z}{2}+1)$, one can show for n_z sufficiently large that r is actually proportional to $c = (4/\pi)^{n_z/2} \times \sqrt{\pi n_z}(n_z/2)^{n_z/2}e^{-n_z/2}$. With elementary algebraic manipulation, the factor c can be expressed as $c = \sqrt{\pi n_z}e^{n_z/2[\epsilon+\ln n_z]}$ with $\epsilon = \ln(2) \Leftrightarrow \ln(\pi) \Leftrightarrow 1$. This remark shows clearly the exponential increase of r with n_z as reported in following section.

2.3. Evolution of $V_c^{n_z}/V_s^{n_z}$ with n_z

As already stated, during many years the generation of FA uniformly distributed in hyperellipsoid was based on the generation of FA uniformly distributed in the minimal hyperparallelepiped containing the validation gate. This method is still frequently used in many tracking simulators. When the dimension n_z of measurement space is low ($n_z \leq 3$), this method is acceptable since the overcharge of computations is low. However, whenever $n_z > 3$, such method must really be bannished because of its strong overcharge of needless computations involved due to the exponential growth of the ratio r of hyperparallelepiped volume over hyperellipsoid volume with dimension.

We have plotted on figure 1 the growth of $r = V_c^{n_z}/V_s^{n_z}$ with n_z. We can see the exponential growth of r which renders this method very inefficient for Monte-Carlo simulations since most of the time the method generates FA outside the hyperellipsoid rather than inside. For example for $n_z = 7$, if one wants to generate on average 100 FA in a given hyperellipsoid, the method requires to generate and to test on average 2700 FA in hyperparallelepiped. This is the major limitation of this method for Monte Carlo simulations. The new algorithm presented in this paper does not suffer of such limitation as it will be shown.

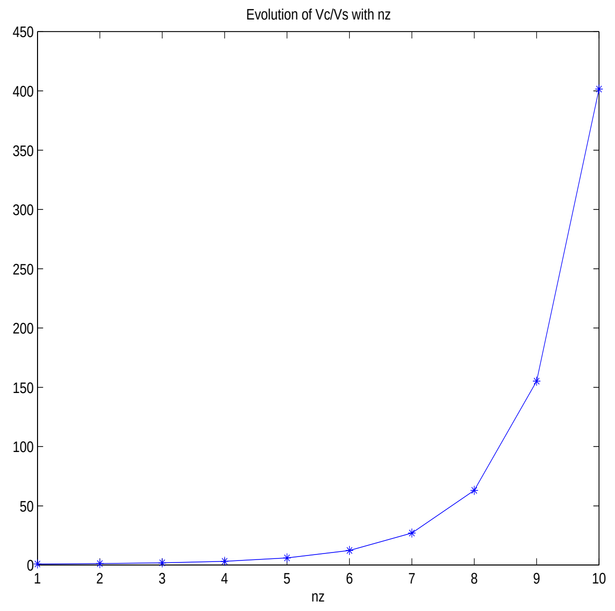


Figure 1. Evolution of $V_c^{n_z}/V_s^{n_z}$ with n_z

3. LIMITATIONS OF HF ALGORITHM FOR SIMULATIONS

In this section we recall the HF algorithm proposed recently in [10] to generate random points uniformly distributed in validation gate. We point out some problems arising in simulations with this algorithm and show its practical limitations.

The HF algorithm consists of two stages. The first stage generates the Poisson-distributed number m_{FA} of false validated measurements in the hyperellipsoid under consideration. T. Ho and M. Farooq use the Poisson Random Generator (PRG) proposed in [6]. This is only one issue possible among many other PRG available in the literature [17,8]. We will not discuss here about the quality of PRG used in stage I. Under Matlab, m_{FA} can be easily generated by using the simple instruction `m_FA=poissrnd(Lfa*V)` where `V` is the volume of hyperellipsoid given by (17) and `Lfa` is the spatial density of FA. The stage II generates false measurements supposed to be uniformly distributed in hyperellipsoid. This is accomplished as follows:

1. **Stage I**: Poisson Random Generator (PRG) to generate m_{FA}
2. **Stage II**: Generation of m_{FA} random points uniformly distributed in gate
   - Obtain the orthogonal matrix $\mathbf{L}$ such that

     $$\mathbf{L}^{-1}\mathbf{S}^{-1}\mathbf{L} = \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & \lambda_{n_z-1} & 0 \\ 0 & \dots & 0 & 0 & \lambda_{n_z} \end{bmatrix}$$

     where each λ_i, $1 \leq i \leq n_z$, is an eigenvalue of the matrix $\mathbf{S}^{-1}$ and $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n_z}$.
   - $l = 1$
   - Repeat until $l > m_{FA}$
     - Form the vector $\mathbf{x} = [x_1 \ldots x_{n_z}]$ where
     - $x_1 \sim \mathcal{U}[\Leftrightarrow\sqrt{\gamma/\lambda_1}, \sqrt{\gamma/\lambda_1}]$ and
     - for $2 \leq i \leq n_z$, $x_i \sim \mathcal{U}[\Leftrightarrow\sqrt{\tau_i/\lambda_i}, \sqrt{\tau_i/\lambda_i}]$
     - with $\tau_i = \gamma \Leftrightarrow \lambda_1 x_1 \Leftrightarrow \ldots \Leftrightarrow \lambda_{i-1} x_{i-1}^2$
     - $\tilde{\mathbf{z}}(l) = \mathbf{L}\mathbf{x}$ (or equivalently $\mathbf{z}(l) = \mathbf{L}\mathbf{x} + \hat{\mathbf{z}}$)
     - $l = l + 1$

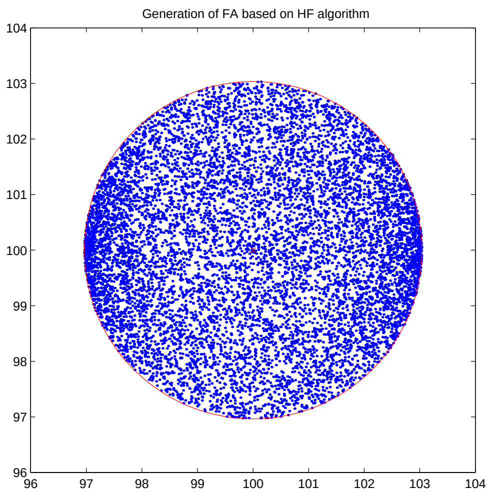
where γ is the gating threshold and $x \sim \mathcal{U}[a, b]$ means that x is a real random variable uniformly distributed in the interval $[a, b]$. $\tilde{\mathbf{z}}(l)$ is the l-th innovation generated in the validation gate by the algorithm. The l-th false measurement is obtained by adding the center of the gate $\hat{\mathbf{z}}$ to $\tilde{\mathbf{z}}(l)$; i.e. $\mathbf{z}(l) = \tilde{\mathbf{z}}(l) + \hat{\mathbf{z}}$.

This algorithm outperforms Li's algorithm [12] in term of computation cost because it does not require an indefinite number of iterations since not rejection test is necessary. The first drawback of this algorithm is its necessity to compute $\mathbf{S}^{-1}$ and sort all eigenvalues of $\mathbf{S}^{-1}$. This first step of stage II can become actually very difficult to achieve with good precision as already reported in [12]. Usually, this requires a lot of computations when dimension of measurement space becomes high. The second and most important drawback of HF algorithm is its reliability. Actually, the random points generated by HF algorithm appear to be not uniformly distributed in the gate (see following examples). All results reported here have been obtained with the generic Matlab routine (HFalgorithm.m) given in the appendix to convince the reader about these concluding remarks and results.

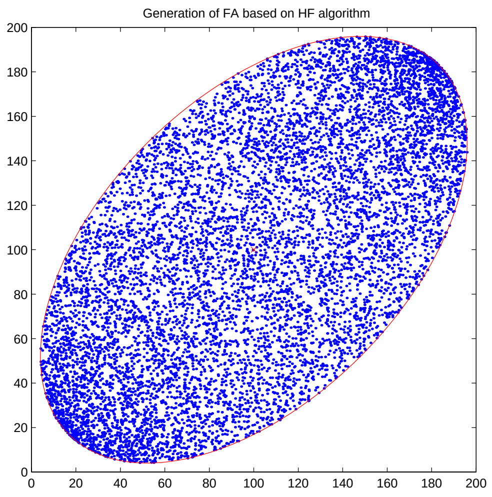
3.1. Simulation results of random points generated by HF algorithm

We present here three results of random points generation obtained by HF algorithm in 2D measurement space ($n_z = 2$). The gating probability P_g has been set to 0.99 which imposes the following gating threshold $\gamma \approx 9.2103$. The number of points generated in each validation gate has been arbitrary chosen to $m_{FA} = 10000$. The center $\hat{\mathbf{z}}$ of gates has been taken at $\hat{\mathbf{z}} = [100\ 100]'$. The simulation results presented on figure 2 correspond to the three choices of covariance matrices for $\mathbf{S}$

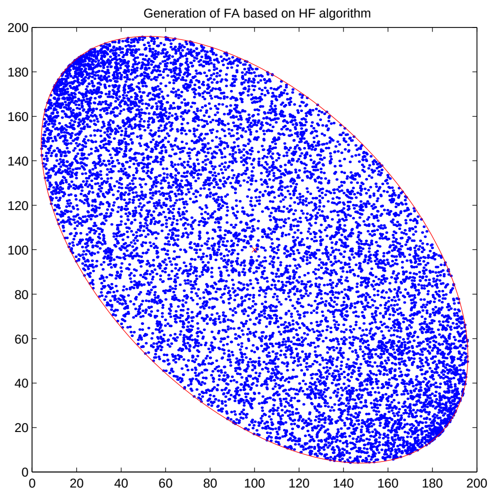
$$\mathbf{S}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{S}_2 = \begin{bmatrix} 1000 & 500 \\ 500 & 1000 \end{bmatrix} \qquad \mathbf{S}_3 = \begin{bmatrix} 1000 & \Leftrightarrow 500 \\ \Leftrightarrow 500 & 1000 \end{bmatrix}$$



2.1: Gate 1: $\mathbf{S} = \mathbf{S}_1$



2.2: Gate 2: $\mathbf{S} = \mathbf{S}_2$



2.3: Gate 3: $\mathbf{S} = \mathbf{S}_3$

Figure 2. Simulation results of HF algorithm ($n_z = 2$, $P_g = 0.99$ and $m_{FA} = 10000$)

As we can easily observe, random points generated by HF algorithm cover the entire validation gates. However, simulation results show also that the false alarms are actually not exactly uniformly distributed in gates since there are two regions (darker areas on figures) in each gate which have a higher spatial density. This can be observed at left and right side of x-axis for gate 1 and at extremities of major axis of gates 2 and 3. This clearly indicates that practical use of HF algorithm is questionable. To overcome this drawback, we propose a new efficient algorithm which is more reliable both in term of uniformity, in term of computation burden reduction and which does not require inversion of $\mathbf{S}$.

4. A NEW EFFICIENT ALGORITHM

4.1. Theoretical development of the new algorithm

As in previous algorithms, our new algorithm consists of two stages. The first stage generates the Poisson-distributed number m_{FA} of false validated measurements in the hyperellipsoid under consideration with some existing PRG algorithms [17,6,8]. In our Matlab simulations, we simply use the `poissrnd` function of Matlab statistics toolbox for stage I. The stage II, which generates m_{FA} false measurements uniformly distributed in hyperellipsoid, is now presented.

Consider the hyperellipsoid in $\mathbb{R}^{n_z}$ defined by $\mathcal{V}^{n_z}(\gamma) \triangleq \{\tilde{\mathbf{z}} \in \mathbb{R}^{n_z} : \tilde{\mathbf{z}}'\mathbf{S}^{-1}\tilde{\mathbf{z}} \leq \gamma\}$ where $\mathbf{S}$ is a real symmetric positive definite matrix. This ellipsoid is equivalent, by denoting $\mathbf{x} \triangleq \tilde{\mathbf{z}}/\sqrt{\gamma}$ to "unit" ellipsoid $\mathcal{V}^{n_z}(1) \triangleq \{\mathbf{x} \in \mathbb{R}^{n_z} : \mathbf{x}'\mathbf{S}^{-1}\mathbf{x} \leq 1\}$. As already recalled in section 2.2, since $\mathbf{S}$ is a real symmetric positive definite matrix, there exists a square matrix $\mathbf{T}$ such that $\mathbf{S} = \mathbf{T}\mathbf{T}' \Leftrightarrow \mathbf{S}^{-1} = \mathbf{T}'^{-1}\mathbf{T}^{-1}$ Using the following linear transformation $\mathbf{y} = \mathbf{T}^{-1}\mathbf{x}$, one has $\mathbf{x}'\mathbf{S}^{-1}\mathbf{x} = \mathbf{y}'\mathbf{y}$. Consequently, if $\mathbf{y}$ is uniformly distributed in unit hypersphere $\mathcal{V}_s^{n_z}(1) = \{\mathbf{y} \in \mathbb{R}^{n_z} : \mathbf{y}'\mathbf{y} \leq 1\}$, then $\mathbf{x} = \mathbf{T}\mathbf{y}$ will be uniformly distributed in "unit" ellipsoid $\mathcal{V}^{n_z}(1)$ because of linear mapping between $\mathbf{x}$ and $\mathbf{y}$ and therefore $\tilde{\mathbf{z}} = \sqrt{\gamma}\mathbf{x}$ will be uniformly distributed in validation gate $\mathcal{V}^{n_z}(\gamma)$ which is what we are looking for. Hence, our problem is mathematically equivalent to the problem of generation of random points uniformly distributed in hypersphere $\mathcal{V}_s^{n_z}(1)$. The solution of this problem is however well established in the milestone book of L. Devroye [8] (Chapter V, section 4) and we present now the algorithm for generating points in $\mathcal{V}_s^{n_z}(1)$.

We first recall basic definitions and theorems about radially symmetric random variables in $\mathbb{R}^{n_z}$. A random vector $\mathbf{u} \in \mathbb{R}^{n_z}$ is radially symmetric if $\mathbf{A}\mathbf{u}$ is distributed as $\mathbf{u}$ for all orthonormal (rotation) $n_z \times n_z$ matrices $\mathbf{A}$. If moreover $Pr\{\mathbf{u} = \mathbf{0}\} = 0$, then $\mathbf{u}$ is said to be strictly radially symmetric. $\mathbf{u}$ is uniformly distributed on unit hypersphere $\mathcal{V}_s^{n_z}(1)$ when $\mathbf{u}$ is radially symmetric with $\|\mathbf{u}\|= 1$ ($\|.\|$ being the standard L_2 norm). The density $p(\mathbf{u})$ of any radially symmetric random variable $\mathbf{u}$ is necessarly of the form $g(\|\mathbf{u}\|)$ such that $\int_0^\infty n_z V_s^{n_z} r^{n_z-1} g(r) dr = 1$ where $V_s^{n_z} = (\pi)^{\frac{n_z}{2}}/\Gamma(\frac{n_z}{2}+1)$ is the volume of unit hypersphere derived in (19). $g(.)$ is called the defining function of radial density $p(\mathbf{u})$.

The generation of random points uniformly distributed *on* $\mathcal{V}_s^{n_z}(1)$ can be easily obtained via normal random variates as follows [8]: each random point $\mathbf{u}_i$ ($i = 1, \ldots, m_{FA}$) is generated by drawing n_z iid normal random variates $u_1, \ldots, u_{n_z}$, computing $s = (u_1^2 + \ldots + u_{n_z}^2)^{1/2}$ and returning $\mathbf{u}_i = [u_1/s, \ldots, u_{n_z}/s]'$. The radial transformation theorem [8], states that:

a) if $\mathbf{u}$ is strictly radially symmetric *in* $\mathbb{R}^{n_z}$ with a defining function $g(.)$, then $r = \|\mathbf{u}\|$ has density $p(r) = n_z V_s^{n_z} r^{n_z-1} g(r)$.

b) if $\mathbf{u}$ is uniformly distributed *on* $\mathcal{V}_s^{n_z}(1)$ and r is independent of $\mathbf{u}$ and has pdf $p(r)$ above, then $r\mathbf{u}$ is strictly radially symmetric *in* $\mathbb{R}^{n_z}$ with defining function $g(r)$.

A random vector is uniformly distributed *in* $\mathcal{V}_s^{n_z}(1)$ when it is radially symmetric with defining function $g(r) = 1/V_s^{n_z}(1)$ for $0 \leq r \leq 1$ and $g(r) = 0$ for $r > 1$.

We give here the proof of statement b) not provided in [8]. If we consider a random vector $\mathbf{u}$ uniformly distributed *over* $\mathcal{V}_s^{n_z}(1)$ and a random variable r uniformly distributed *in* $\mathcal{V}_s^{n_z}(1)$ with $p(r) = n_z r^{n_z-1}$, then we want to prove that $\mathbf{z} = r\mathbf{u}$ is uniformly distributed *in* $\mathcal{V}_s^{n_z}(1)$ which is equivalent to prove $p(z) = \frac{1}{V_s^{n_z}(1)}\mathbf{1}_{\mathcal{V}_s^{n_z}(1)}$ (where $\mathbf{1}_a$ denotes the indicator function on set a). Consider now the following pdf $p(\mathbf{u}) = \frac{1}{T(\epsilon)}\mathbf{1}_{\mathcal{T}(\epsilon)}$ defined in hypertorus

$\mathcal{T}(\epsilon) \triangleq \{\mathbf{u} \in \mathbb{R}^{n_z} : 1 \Leftrightarrow \epsilon \leq \|\mathbf{u}\| \leq 1 + \epsilon\}$ having volume (by setting $\mathbf{S} = \mathbf{I}$ and $\gamma^{1/2} = 1 + \epsilon$ in (17)) $T(\epsilon) = c_{n_z}[(1+\epsilon)^{n_z} \Leftrightarrow (1 \Leftrightarrow \epsilon)^{n_z}] \simeq 2\epsilon n_z c_{n_z}$ when $\epsilon \to 0$. Now, consider the pdf of $\mathbf{z}$ which can be expressed as $p(\mathbf{z}) = \int_0^1 p(\mathbf{z} = r\mathbf{u})p(r)dr$. By taking into account previous expressions for $p(\mathbf{u})$ and $p(r)$, $p(\mathbf{z})$ can equivalently be expressed as

$$p(\mathbf{z}) = \lim_{\epsilon \to 0} \frac{1}{T(\epsilon)} \int_0^1 \frac{n_z}{r^{n_z}} \mathbf{1}_{\mathcal{T}(\epsilon)} r^{n_z - 1} dr = \lim_{\epsilon \to 0} \frac{n_z}{T(\epsilon)} \int_0^1 \mathbf{1}_{1-\epsilon \leq \|\mathbf{u}\| \leq 1+\epsilon} \frac{1}{r} dr = \lim_{\epsilon \to 0} \frac{n_z}{T(\epsilon)} \int_{\frac{\|\mathbf{z}\|}{1+\epsilon}}^{\frac{\|\mathbf{z}\|}{1-\epsilon}} \frac{1}{r} dr \times \mathbf{1}_{\|\mathbf{u}\| \leq 1}$$

by integration, one gets

$$p(\mathbf{z}) = \lim_{\epsilon \to 0} \frac{n_z}{T(\epsilon)} \log[\frac{1+\epsilon}{1 \Leftrightarrow \epsilon}] \times \mathbf{1}_{\|\mathbf{u}\| \leq 1} \simeq \frac{n_z}{2\epsilon n_z c_{n_z}} 2\epsilon \times \mathbf{1}_{\|\mathbf{u}\| \leq 1} = \frac{1}{c_{n_z}} \mathbf{1}_{\|\mathbf{u}\| \leq 1} \equiv \frac{1}{V_s^{n_z}(1)} \mathbf{1}_{V_s^{n_z}(1)}$$

which completes the proof.

From the previous theorem, the following steps allow to generate random point $\mathbf{y}$ uniformly distributed *in* $\mathcal{V}_s^{n_z}(1)$:

1. generate a random vector $\mathbf{u}$ uniformly distributed on $\mathcal{V}_s^{n_z}(1)$
2. generate a scalar random variate r with density $p(r) = n_z r^{n_z - 1}$
3. return $\mathbf{y} = r\mathbf{u}$

To generate r ($0 \leq r \leq 1$) following pdf $n_z r^{n_z-1}$ in previous step 2, we use the standard inverse method [17,16] as follows. The repartition function associated with $p(r)$ is

$$v \triangleq F(t) = Pr\{r \leq t\} = n_z \int_0^t r^{n_z - 1} dr = t^{n_z}$$

and its inverse is equal to $F^{-1}(v) = v^{1/nz}$. Hence, the generation of $r \sim p(r)$ is easily obtained by generating $F^{-1}(v)$ with $v \sim \mathcal{U}([0;1])$.

4.2. Summary

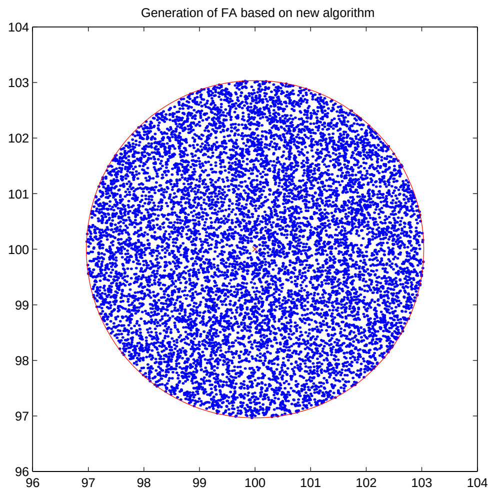
We give here the summary of our new algorithm for generating random points uniformly distributed in hyperellipsoid $\mathcal{V}^{n_z}(\gamma)$.

1. **Stage I**: Poisson Random Generator (PRG) to generate m_{FA}
2. **Stage II**: Generation of m_{FA} random points uniformly distributed in gate as follows
   - generate iid points $\mathbf{u}_i$ ($i = 1, \ldots, m_{FA}$) uniformly distributed on unit hypersphere $\mathcal{V}_s^{n_z}(1)$. Each point $\mathbf{u}_i$ is generated by drawing n_z iid normal random variates $u_1, \ldots, u_{n_z}$, computing $s = \sqrt{u_1^2 + \ldots + u_{n_z}^2}$ and returning $\mathbf{u}_i = [u_1/s, \ldots, u_{n_z}/s]$.
   - generate, independently of $\mathbf{u}_i$, scalar$r_i = v^{1/nz}$ ($i = 1, \ldots, m_{FA}$) with $v \sim \mathcal{U}([0;1])$.
   - compute $\mathbf{y}_i = r\mathbf{u}_i$ ($i = 1, \ldots, m_{FA}$). $\mathbf{y}_i$ is uniformly distributed in $\mathcal{V}_s^{n_z}(1)$.
   - compute square root matrix $\mathbf{T}$ of $\mathbf{S}$ using Cholesky factorization ($\mathbf{S} = \mathbf{T}\mathbf{T}'$).
   - compute $\mathbf{x}_i = \mathbf{T}\mathbf{y}_i$ ($i = 1, \ldots, m_{FA}$).
   - return (false alarms) $\mathbf{z}_i = \sqrt{\gamma}\mathbf{x}_i + \hat{\mathbf{z}}$ ($i = 1, \ldots, m_{FA}$).

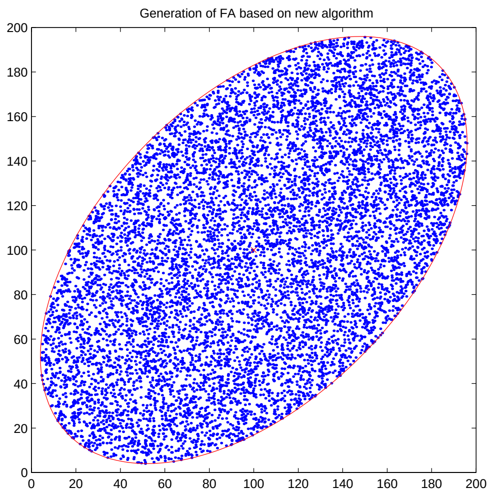
4.3. Simulation results of random points generated by the new algorithm

We present on figure 3 the results of random points generation in 2D measurement space obtained with our new algorithm (provided in appendix for convenience) with same parameters as before ($n_z = 2$,$P_g = 0.99$, $m_{FA} = 10000$, $\hat{\mathbf{z}} = [100\ 100]'$) with

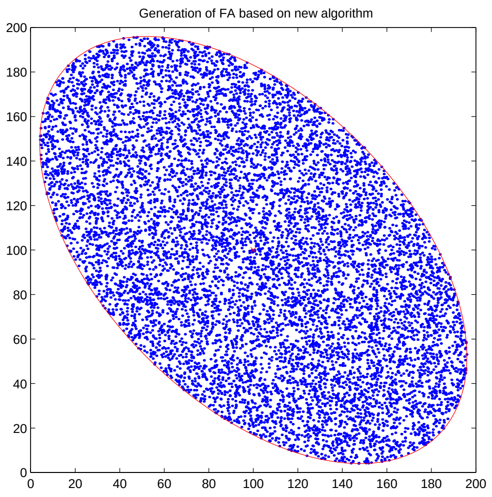
$$\mathbf{S}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{S}_2 = \begin{bmatrix} 1000 & 500 \\ 500 & 1000 \end{bmatrix} \qquad \mathbf{S}_3 = \begin{bmatrix} 1000 & \Leftrightarrow 500 \\ \Leftrightarrow 500 & 1000 \end{bmatrix}$$



3.1: Gate 1: $\mathbf{S} = \mathbf{S}_1$



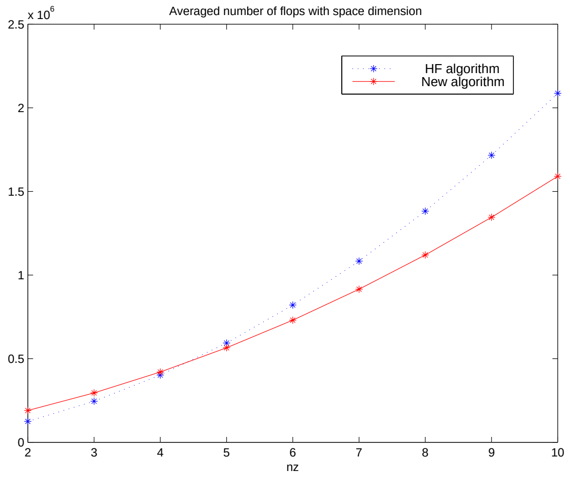
3.2: Gate 2: $\mathbf{S} = \mathbf{S}_2$



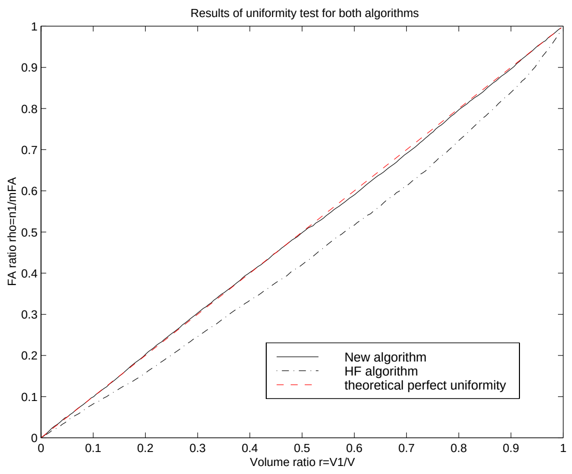
3.3: Gate 3: $\mathbf{S} = \mathbf{S}_3$

Figure 3. Simulation results of new algorithm ($n_z = 2$, $P_g = 0.99$ and $m_{FA} = 10000$)

Simulations results on figure 3 show the better quality of spatial uniformity of random points generated by our new algorithm with respect to the uniformity obtained by HF algorithm on figure 2. This ”visual” conclusion is reinforced by uniformity test results presented on next figure 4.2.



4.1: Complexity of both algorithms



4.2: Uniformity quality of both algorithms

Figure 4. Performance comparison of the new algorithm vs. HF algorithm

The comparison of the averaged number of Matlab flops (floating point operations) of the two algorithms with variation of measurement space dimension n_z is plotted on figure 4.1. These results are based on 10 Monte Carlo runs for each value of n_z. Each run consists in random generation of covariance matrix $\mathbf{S}$ with $\dim(\mathbf{s}) = n_z \times n_z$ and generation of $m_{FA} = 5000$ false alarms per gate. Results indicate the $O(n^3/3)$ complexity of our new algorithm with measurement space dimension. The charge of computation is mainly due to Cholesky factorization step involved in our algorithm which requires $O(n^3/3)$ arithmetic operations. All other steps of our algorithm require only $O(n)$ operations. For small values of measurement space dimension ($n_z \leq 3$) the HF algorithm seems to require less amount of flops than our new algorithm. The difference of computation load between two algorithms is however not that much. When the dimension n_z increases, our algorithm however outperforms drastically HF algorithm in term of computation loads. We point out the fact that our algorithm does not require the inversion of matrix $\mathbf{S}$ but only Cholesky factorization of $\mathbf{S}$. The number of flops for matrix inversion have not been taken into account for complexity evaluation of HF algorithm. If this had been done, its complexity would become greater than the complexity of our algorithm even for small measurement space dimensions. On figure 4.2, we present the results of the following uniformity test applied to both algorithms (with parameters $\mathbf{S} = \mathbf{S}_3$, $m_{FA} = 10000$ and $P_g = 0.99$). For any given real symmetric definite positive matrix $\mathbf{S}$ and positive threshold γ, we consider the full gate volume $V^{n_z}(\gamma)$ and any enclosing gate $V^{n_z}(\gamma_1) < V^{n_z}(\gamma)$ with $\gamma_1 = r\gamma$, $(0 \leq r \leq 1)$. The ratio of two volumes $V^{n_z}(\gamma_1)/V^{n_z}(\gamma)$ is then exactly equal to $r^{n_z/2}$. If the random points are exactly uniformly distributed in $V^{n_z}(\gamma)$, all of them included in any $V^{n_z}(\gamma_1) < V^{n_z}(\gamma)$ must be necessary uniformly distributed in $V^{n_z}(\gamma_1)$ and therefore the ratio $\hat{\rho}$ of number of points n_1 in $V^{n_z}(\gamma_1)$ over the total number m_{FA} of points generated in $V^{n_z}(\gamma)$ must be theoretically equal to $r^{n_z/2}$. In 2D measurement space, if the algorithms are well designed, one should get the straight line $\hat{\rho} \simeq r$ for r varying in $[0;1]$. Simulation results clearly indicate the poor performance obtained by HF algorithm by using such empirical uniformity test. This confirm our previous ”visual” conclusion about HF algorithm given in section 3.1. On the contrary, the new algorithm provides uniformity performances which appear to be very close to optimality.

5. CONCLUSION

We have presented in this paper a new efficient algorithm for generating directly random points uniformly distributed in hyperellipsoid defined by $[\mathbf{z} \Leftrightarrow \hat{\mathbf{z}}]'\mathbf{S}^{-1}[\mathbf{z} \Leftrightarrow \hat{\mathbf{z}}] \leq \gamma$. This algorithm outperforms all previous existing methods in term of computation savings (since computation of $\mathbf{S}^{-1}$ and computation of eigenvalues of $\mathbf{S}^{-1}$ is not required), in term of quality of uniformity obtained and in term of complexity ($O(n^3/3)$). The choice of this new method is highly recommended specially in multitarget tracking research area for running Monte Carlo simulations requiring an efficient and fast way to generate false measurements in validation gates.

6. APPENDICES

We provide here only stage II of the HF and new algorithm. The stage I can be easily accomplished by using `poissrnd` function of statistics toolbox of Matlab (if available) or by implementing one of PRG described in [17,6,8].

6.1. Generic Matlab implementation of HF algorithm

```
%*****************************************************************
function [ z_fa ]=HFalgorithm(Gamma_Threshold,S_inv,z_hat,m_FA)
%*****************************************************************
% This routine implements the T.J . HO and M. Farooq algorithm for
% generating random points uniformly distributed in hyperellipsoid .
% Inputs : Gamma_Threshold = Gating threshold (>0)
%          S_inv = inverse of covariance matrix S (dim(S)=nzxnz)
%          z_hat = center of the gate (dim(z_hat)=nzx1)
%          m_FA = number of false alarms to generate in the gate
% Output: Z_fa = [ z (1),... z(mFA)] set of FA generated by HF algorithm
%*****************************************************************
[V,D]=eig(S_inv );        % Decomposition inv(V)*S_inv*V=Diag(eigenvalues)
[Y,I]=sort(diag(D ));     % Sorting of eigenvalues by ascending order
A=diag(Y,0);              % Diagonal matrix of sorted eigenvalues
L=V(:,I );                % Permutation of eigenvectors corresponding to eigenvalues
z_fa =[]; nz=size( z_hat ,1);
for l=1:m_FA
    x(1)=sqrt(Gamma_Threshold/A(1,1))*(2*rand-1);
    for i=2:nz
        Tau_i=0;
        for j=1:i-1, Tau_i=Tau_i+A(j,j)*(x(j)^2); end
        Tau_i=Gamma_Threshold-Tau_i;
        x(i)=sqrt(Tau_i/A(i,i))*(2*rand-1);
    end
    z_fa=[z_fa (L*x'+z_hat)];
end
```

6.2. Generic Matlab implementation of the new algorithm

```
%*********************************************************************
function [ z_fa ]=New_Algorithm(Gamma_Threshold,S,z_hat,m_FA)
%*********************************************************************
% This routine implements the new algorithm for generating random
% points uniformly distributed in hyperellipsoid for nz>=2.
% Inputs : Gamma_Threshold = Gating threshold (>0)
%          S = Covariance matrix S ( dim(S)=nzxnz)
%          z_hat = center of the gate ( dim(z_hat)=nzx1)
%          m_FA = number of false alarms to generate in the gate
% Output: Z_fa = [ z (1),... z(mFA)] set of FA generated by new algorithm
%*********************************************************************
nz=length(S );
X_Cnz=randn(nz,m_FA);
X_Cnz=X_Cnz./kron(ones(nz,1),sqrt(sum(X_Cnz.^2)));       % Points uniformly distributed on hypersphere
R=ones(nz,1)*(rand(1,m_FA).^(1/nz ));                    % Points with pdf nz*r^(nz-1); 0<r<1
unif_sph=R.*X_Cnz;                                       % m_FA points in the hypersphere
T=chol(S );                                              % Cholesky factorization of S => S=T'T
unif_ell =T'*unif_sph ;                                  % Hypersphere to hyperellipsoid mapping
z_fa=( unif_ell *sqrt(Gamma_Threshold)+(z_hat*ones(1,m_FA))); % Translation around gate center
```

REFERENCES

1. Y. Bar-Shalom, T. E. Fortmann, *Tracking and Data Association*, Academic Press, New York, 1988.
2. Y. Bar-Shalom, X. R. Li, *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, New York, 1993.
3. Y. Bar-Shalom, X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*, (3rd printing), YBS Publishing, Storrs, CT, 1995.
4. R. Bellman, *Introduction to Matrix Analysis*, McGraw-Hill, New York, 1960.
5. G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic Press, New York, 1977.
6. P. Bratley and al., *A guide to Simulation*, Springer-Verlag, New York, 1983.
7. M. A. Chmielewski, *Elliptically Symmetric Distributions: A review and Bibliography*, International Statistical Review, 49:67-74,1981.
8. L. Devroye, *Non-Uniform Random variate Generation*, Springer-Verlag, New York, 1986.
9. K. T. Fang, S. Kotz, K. W. Ng, *Symmetric Multivariate and related Distributions*, Chapman and Hall, New York, 1990.
10. T. -J. Ho, M. Farooq, *An Efficient Method for Uniformly Generating Poisson-Distributed Number of Measurements in a Validation Gate*, Proc. of 2nd Int. Conf. on Inf. Fusion (Fusion'99), Sunnyvale, CA, Vol 2., pp 749-754 , July 6-8, 1999.
11. M. E. Johnson, J. S. Ramberg, *Elliptically Symmetric Distributions: Characterizations and Random variate Generation*, A.S.A. Proc. Statist. Comp. Sect., pp. 262-265, 1977.
12. X. R. Li, *Generation of Random Points Uniformly Distributed in Hyperellipsoids*, Proc. of 1st IEEE Conf. on Control Applications, pp. 654-658, Dayton, OH, Sept., 1992.
13. K. S. Miller, *Multidimensional Gaussian Distributions*, John Wiley and Sons, Inc., New York, 1964.
14. R. J. Muirhead, *Aspects of Multivariate Statistical Theory*, Springer-Verlag, New York, 1986.
15. M. E. Muller, *A Note on a Method for Generating Points Uniformly on n-Dimensional Spheres*, Communications of the ACM, 2(4), pp. 19-20, 1959.
16. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill Book Co - Singapore, International Editions, Electrical Engineering Series, 6th Printing, 1989.
17. R. Y. Rubinstein, *Simulation and the Monte Carlo Method*, John Wiley and Sons, New York, 1981.
18. M. Sibuya, *A Method for Generating Uniformly Distributed Points on n-Dimensional Spheres*, Ann. Inst. Statist. Math., 14:,81-85, 1962.