Generating Uniformly Random Numbers within a Hyper Shape (Super-Ring) in N-dimentional Space : A MATLAB Simulation

Ehsan Farzadnia

23 May 2018

M.Sc. of Secure Computing

ehsan_farzadnia@mut.ac.ir
e405504@gmail.com

Malek Ashtar University of Technology
Faculty of Electrical and Computer Engineering
Artificial Intelligence Laboratory

Report number:	MUTR3-D-98005
Date Performed:	May 8, 2018
Advisors:	Professor Hossein Shirazi
	Professor Alireza Nowroozi

1 Objective

In this report, we provided a source code for generating uniformly random numbers within a hyper shape in N-dimensional space, which is released in the world for the first time. It generates numbers in an n-dimensional space, but only inside a hyper shape (or super-ring that is a sphere in n-dimension) as uniformly random. Generating random numbers inside a hyper-shape relies on some user-defined parameter values that must be tuned simply. For simulating this idea in MATLAB, we inspired by [1] and its important strategy called "polar coordinations". Its general relation is as follows :

$$
\begin{gathered}
d'y_1 = dy_1 + p.cos(\theta_1) \\
d'y_2 = dy_2 + p.sin(\theta_1)cos(\theta_2) \\
... \\
d'y_{n-1} = dy_1 + p.sin(\theta_1)sin(\theta_2)...cos(\theta_{n-1}) \\
d'y_n = dy_n + p.sin(\theta_1)sin(\theta_2)...sin(\theta_{n-1})
\end{gathered}
$$

Where $d'y_1$, $d'y_2$, ..., $d'y_n$ is uniformly random generated candidate number within a super-ring with centroid dy_1, dy_2, ..., dy_n, and also raduis is p. The p is the $Polardiameter$, which is as the same as raduis of Super-Ring. It is variable in range of $[\sqrt[2]{n}/exp(t),\ \sqrt[2]{n}/exp(t-1)]$. The t is generation evolution. Whatever it goes to be higher, similar to variance, it causes the density of the distribution of generated random numbers inside the super-ring are up. Therefore, they are going to close to the center. On the other hand, it should be controled by user. In addition, θ_1, θ_2, ..., θ_n are random variables in [0,360]. In the next experiments , we determined its value properly.
In [1], the Polar coordinations strategy is applied to facilitate proliferating the uniformly random generated candidate detectors (Antibodies) in non-self N-dimensional problem space as well.
The Clone phase is actually to find the best position around the available detector, and then to migrate it to a new position vector with a new covered range, That of course, a newer one may be able to cover some holes. As a result, updating the position vectors of detectors around them means the clone. The Problem of the generation of uniformly random numbers just limited to the inside of an N-dimensional hyper-shape which is a Hard problem. This term is not being able to solve without using polar coordinations. We hope that this MATLAB source code enables can be useful for researchers.

2 Experimental Data

```
function Random_numbers_intoCircle
(radius,circle_center,numberOfrandomSamples,dimension,evolutionGeneration)
%% Created by Ehsan Farzadnia,
% M.Sc of Secure Computing,
% at Malek Ashtar University of Technology. 2018/05/16
% This Source Code produce a uniformly random generated n-dimentional
% Samples in Polar Coordination only within a super-ring shape.
% dimension must be bigger than 2 or equal.
%% the Polar Coordination of an n-dimensional Sample d'y1d'y2...d'y_n
% randomly generated within a Super-ring hyper shape with center of
```

```
% dy1dy2...dy_n and a specific raduis, p is the Polar diameter

%% My Supervisors : Dr. Hossein Shirazi and Alireza Nowroozi

% dis = 0;
s = 'cosinos';

rectangle('Position',[circle_center(1)-radius,circle_center(2) -
radius,2*radius,2*radius],'Curvature',[1,1]);
axis([-5 5 -5 5]);

% while(size(dis(dis >= radius),1) == 0)
       for nn = 1 : numberOfrandomSamples
           [distance] = distance_alc(radius,dimension,evolutionGeneration);
            for iTeta = 1 : dimension - 1
               [teta(iTeta)] = tetaCalc;
            end
                for d = 1 : dimension
                    if d == dimension
                        s = 'sinous';
                    end
                        if d == 1
                           nodes(d,nn) = circle_center(d) +
                           (distance  * cos(teta(d)));
                        else
                            i = 1;
                            prod = 1;
                            while(i <= d - 1)
                                prod = prod * sin(teta(i));
                                i = i + 1;
                            end
                                if strcmp(s,'sinous')
                                    nodes(d,nn) = circle_center(d) +
                                    (distance * prod);
                                else
                                    nodes(d,nn) = circle_center(d) +
                                    (distance * prod * cos(teta(i)));
                                end
                        end
                end
        end

        for i = 1 : numberOfrandomSamples
            dis(i,1) =
            pdist2(transpose(nodes(:,i)),circle_center,'euclidean');
        end
        disp(dis);
        disp(['The number of random samples are upper bound is : ',
        num2str(size(dis(dis >= radius),1))]);
        disp(dis(dis >= radius));

% end

%% plotting ...

    hold on
    plot(circle_center(1)+nodes(1,:),circle_center(2)+nodes(2,:),
```

```
    'rs','LineWidth',5,'MarkerSize',1.5);

end

function distance = distance_alc(radius,dimension,evolutionGeneration)
%      a = 0;
%      b = radius;
%      distance = a + ((b - a) * unifrnd(0,1));
%
 distance = unifrnd((sqrt(dimension)/exp(evolutionGeneration)),
 (sqrt(dimension)/exp(evolutionGeneration - 1)));

end


function teta = tetaCalc
    a = 0;
    b = 2 * pi;
    teta = a + ((b - a) * unifrnd(0,1));
end
```

An example

In this section, we try to test the proposed function through an example in 2D problem space. For a better representation, all test results include 2D-plots. Assume that we need 50 numbers must be generated uniformly randomly in 2D space inside of a ring. In this case, the hyper shape is a circle (or ring) with a specific radius and centroid that is [0,0] for example. Therefore, we tested the proposed function four times by determining different parameter values (radius = 1; circle center = [0; 0]; $numberOfrandomSamples = 50$), and then the evolution generation parameter value is tunned from zero to ten during the tests. Final results have been shown as follows :

Table 1. Threshold rates of Evolution Generation

Raduis	Dimension	Number of Outbounds	Evolution Generation
2	2	34	0
2	2	25	0.25
2	2	21	0.34
2	2	2	0.6
2	**2**	**0**	**0.67**
3	2	12	0
3	2	2	0.2
3	2	0	0.25
2	3	50	0
2	3	42	0.2
2	3	20	0.75
2	3	2	1
2	**3**	**0**	**1.2**
3	3	37	0
3	3	26	0.2
3	3	6	0.5
3	3	1	0.7
3	3	0	0.78
2	**12**	**50**	**0**
2	12	50	0.5
2	12	44	0.9

2	12	16	1.5
2	12	2	2
2	**12**	**0**	**2.2**
3	12	50	0
3	12	44	0.5
3	12	27	0.9
3	12	2	1.5
3	12	0	1.84

Run 1 :
Random_numbers_inside_Circles(2,[0,0],50,2,0)

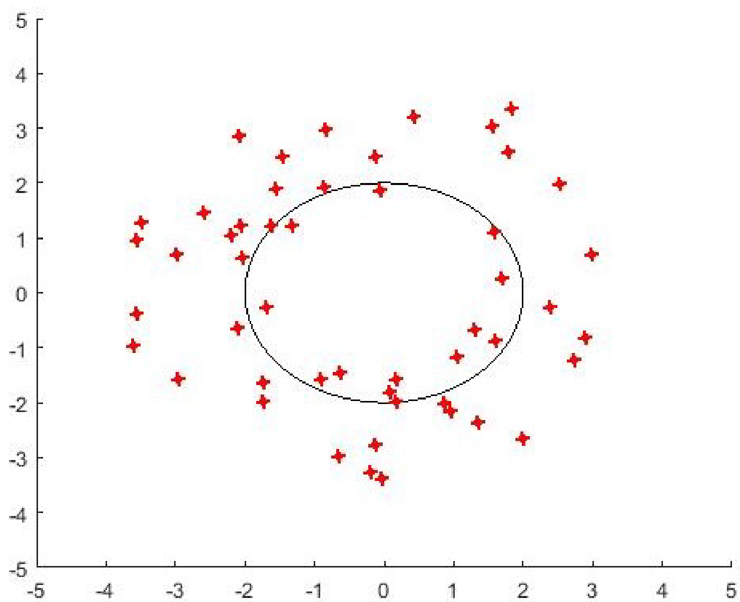

Fig 1. evolution Generation is Zero

According to Table.1, we do not have any random numbers to be outbounded when t value is 0.67 for $n = 2$ and $raduis = 2$. As a result, the number of outbound is going to be close to zero when raising the radius. We tested this function for different dimensions, from 2D to 50D to gain a threshold ratio for $EvolutionGeneration$ parameter. Concerning these results, the threshold value is regulatable.

Note: Provided source code has been tested in MATLAB 2017a. It works on all of dimensions perfectly. Hence, you can try it for bigger values of n. If you have any questions, do not hesitate to contact me.

Run 2 :
Random_numbers_inside_Circle(2,[0,0],50,2,0.5)

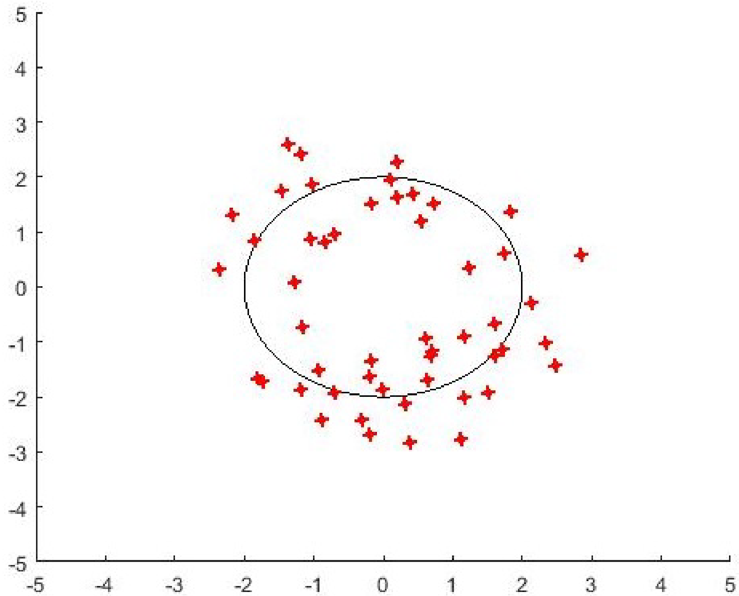

Fig 2. evolution Generation is 0.25

References

[1] Xiao, Xin., Li T. and Zhang, R., 2015. An immune optimization based real-valued negative selection algorithm. Applied Intelligence 42(2), p.289-302. doi:10.1007/s10489-014-0599-9.

Run 3:
Random_numbers_inside_Circle(2,[0,0],50,2,0.5)

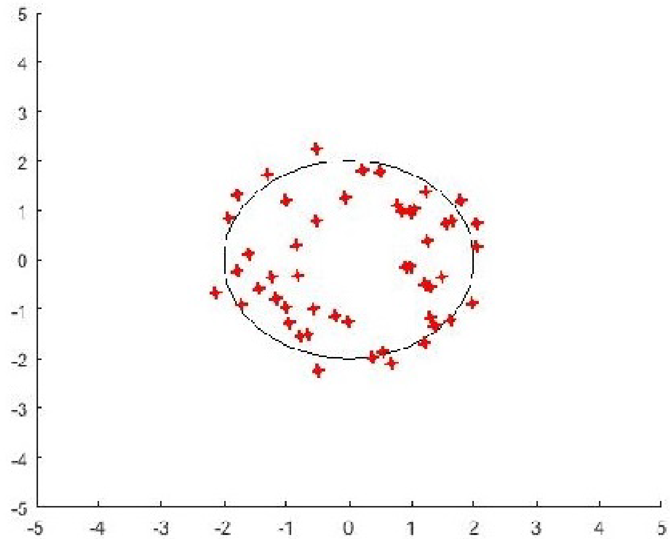

Fig 3.evolution Generation is 0.5

Run 4:
Random_numbers_inside_Cirlce(2,[0,0],50,2,0.75)

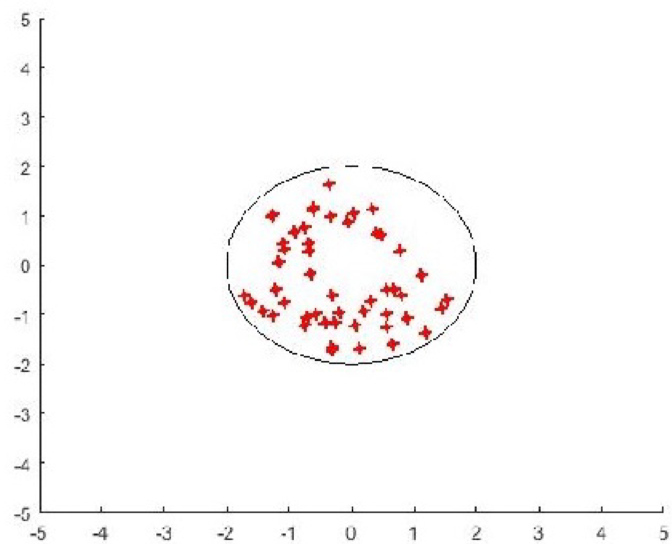

Fig 4.evolution Generation is 0.75