Sampling Tutorial A tutorial for the designs from setrandvar.

* [Help chapter 1](index.html)
* [Tutorials](section_92e5b1d134f462b45362dd31534e5eb9.html)
* [Theory](nisp_theory.html)
* [Sampling Tutorial](setrandvar_tutorial.html)

[<< Theory](nisp_theory.html)

# Sampling Tutorial

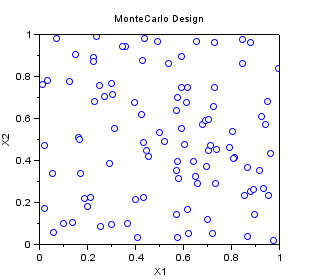
A tutorial for the designs from setrandvar.

### Monte-Carlo Sampling

In the following example, we create a "Monte-Carlo" sampling with 100 points in 2 dimensions, where each variable is uniform in [0,1].

srv=setrandvar\_new(2);  
setrandvar\_buildsample(srv,"MonteCarlo",100);  
sampling=setrandvar\_getsample(srv);  
scf();  
plot(sampling(:,1),sampling(:,2),"bo");  
xtitle("MonteCarlo Design","X1","X2");  
setrandvar\_destroy(srv)

The previous script produces the following output.



### Quasi-Monte-Carlo Sobol Sampling

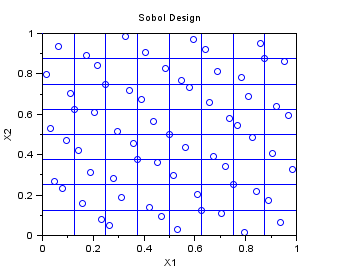
In the following example, we create a "QmcSobol" sampling with 63 points in 2 dimensions, where each variable is uniform in [0,1]. This is a Quasi-Monte-Carlo sequence, which can be used for multidimensionnal integration.

Notice that the first point, at (0,0) has been removed in the sample, so that there is no issue with the generation of samples from inversion of a cumulated distribution function. Indeed, if the point (0,0) has been kept, if normal variables were to be generated, for example, then the infinite point (-Inf,-Inf) would have been generated, which may cause a problem during the uncertainty propagation step, because the model would generate an infinite output.

We divide the unit square with squares with area 1/8 \* 1/8=1/64. We can see that there is exactly one point in each cell. This is because the Sobol sequence is a (t,s)-sequence in base 2 with t=O(slogs). The Sobol sequence use the same base b=2 for all dimensions. This is why we can divide the space with elementary boxes based on a power of 2, so that the number of points in each box is constant. For more details on this topic, please use the "lowdisc" module for Scilab.

srv=setrandvar\_new(2);  
np = 63;  
setrandvar\_buildsample(srv,"QmcSobol",np);  
sampling=setrandvar\_getsample(srv);  
scf();  
plot(sampling(:,1),sampling(:,2),"bo");  
xtitle("Sobol Design","X1","X2");  
// Add the cuts  
n=9  
cut = linspace ( 0 , 1,9);  
for i = 1 : n  
plot( [cut(i) cut(i)] , [0 1] , "-")  
end  
for i = 1 : n  
plot( [0 1] , [cut(i) cut(i)] , "-")  
end  
setrandvar\_destroy(srv)

The previous script produces the following output.



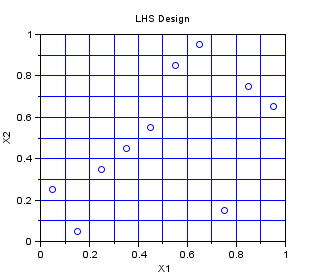
### Latin Hypercube Sampling

In the following example, we create a Centered Lhs sampling for 10 points in dimension 2. The Lhs design which is provided in this module is a centered Lhs (which is different from the classical): as we can see, the points are in the center of each cell.

This is a design which is appropriate for multidimensionnal integration. The one-dimensional projections are as good as can be, in the sense that the design guaranties that at leat one point is in each cell. If the code is monotone with respect to its arguments, then the LHS design improves the accuracy of the estimate of the mean, by reducing the variance of the estimator.

srv=setrandvar\_new(2);  
np = 10;  
setrandvar\_buildsample(srv,"Lhs",np);  
sampling=setrandvar\_getsample(srv);  
scf();  
plot(sampling(:,1),sampling(:,2),"bo");  
xtitle("LHS Design","X1","X2");  
// Add the cuts  
cut = linspace ( 0 , 1,np + 1);  
for i = 1 : np + 1  
plot( [cut(i) cut(i)] , [0 1] , "-")  
end  
for i = 1 : np + 1  
plot( [0 1] , [cut(i) cut(i)] , "-")  
end  
setrandvar\_destroy(srv)

The previous script produces the following output.



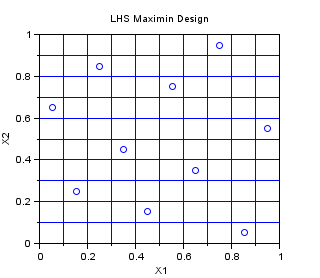
### LHS Maximin

In the following example, we create a Centered Lhs-Maximin sampling for 10 points in dimension 2. We select the best sampling in a set of 1000 samplings, so that the minimum distance between two points in the sampling is maximized. The Lhs design which is provided in this module is a centered Lhs (which is different from the classical): as we can see, the points are in the center of each cell.

This is a design which is appropriate for multidimensionnal integration. It has all the good properties of a LHS design and has, potentially, a better space-filling property. Indeed, the figure below has points which are more regularily distributed in the 2D space than that previous LHS design.

srv=setrandvar\_new(2);  
np = 10;  
setrandvar\_buildsample(srv,"LhsMaxMin",np,1000);  
sampling=setrandvar\_getsample(srv);  
scf();  
plot(sampling(:,1),sampling(:,2),"bo");  
xtitle("LHS Maximin Design","X1","X2");  
// Add the cuts  
cut = linspace ( 0 , 1,np+1);  
for i = 1 : np + 1  
plot( [cut(i) cut(i)] , [0 1] , "-")  
end  
for i = 1 : np + 1  
plot( [0 1] , [cut(i) cut(i)] , "-")  
end  
setrandvar\_destroy(srv)

The previous script produces the following output.

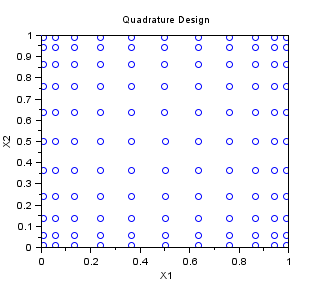


### Quadrature

In the following example, we create a "Quadrature" sampling for np=10 in dimension 2. The one-dimensionnal Gauss quadrature rule has np+1 points. It is tensorised, so that the final 2D sampling has (np+1)^2 points.

srv=setrandvar\_new(2);  
np=10;  
setrandvar\_buildsample(srv,"Quadrature",np);  
sampling=setrandvar\_getsample(srv);  
scf();  
plot(sampling(:,1),sampling(:,2),"bo");  
xtitle("Quadrature Design","X1","X2");  
setrandvar\_destroy(srv)

The previous script produces the following output.

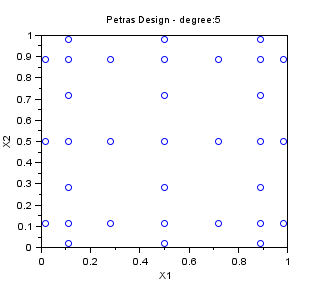


### Petras

In the following example, we create a "Petras" sampling for maxdegree=5 in dimension 2. The Smolyak algorithm is so that the PC decomposition is exact for a polynomial with degree maxdegree. Notice that the number of points is greatly reduced with respect to a "Quadrature" sampling.

srv=setrandvar\_new(2);  
maxdegree=5;  
setrandvar\_buildsample(srv,"Petras",maxdegree);  
sampling=setrandvar\_getsample(srv);  
scf();  
plot(sampling(:,1),sampling(:,2),"bo");  
xtitle("Petras Design - degree:5","X1","X2");  
setrandvar\_destroy(srv)

The previous script produces the following output.



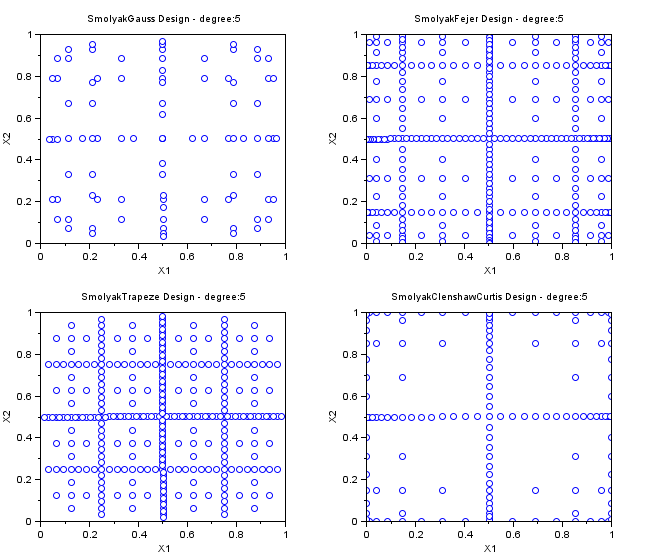
### Other Smolyak Designs

In the followings example, we create various Smolyak samplings for maxdegree=5 in dimension 2. These SmolyakGauss algorithms are so that the PC decomposition is exact for a polynomial with degree maxdegree. Notice that the number of points is greatly reduced with respect to a "Quadrature" sampling.

* The "SmolyakGauss" sampling computes a multi-dimensional sparse grid based on a Gauss one-dimensional formula.
* The "SmolyakTrapeze" sampling computes a multi-dimensional sparse grid based on a Trapezoidal one-dimensional formula.
* The "SmolyakFejer" sampling computes a multi-dimensional sparse grid based on a Fejer one-dimensional formula.
* The "SmolyakClenshawCurtis" sampling computes a multi-dimensional sparse grid based on a Clenshaw-Curtis one-dimensional formula.
* This one-dimensional formula uses absissas based on the roots of Chebyshev polynomials.

names=[  
"SmolyakGauss"  
"SmolyakFejer"  
"SmolyakFejer"  
"SmolyakClenshawCurtis"  
];  
maxdegree=5;  
scf();  
for i=1:4  
subplot(2,2,i);  
srv=setrandvar\_new(2);  
setrandvar\_buildsample(srv,names(i),maxdegree);  
sampling=setrandvar\_getsample(srv);  
plot(sampling(:,1),sampling(:,2),"bo");  
xtitle(names(i)+" Design - degree:"+string(maxdegree),..  
"X1","X2");  
setrandvar\_destroy(srv);  
end

The previous script produces the following output.



### Number of experiments in quadrature samplings

In the followings example, we compute the number of experiments in various quadrature-type samplings.

names=[  
"Quadrature"  
"Petras"  
"SmolyakGauss"  
"SmolyakFejer"  
"SmolyakTrapeze"  
"SmolyakClenshawCurtis"  
];  
for maxdegree=1:9  
mprintf("maxdegree=%d\n",maxdegree)  
for i=1:6  
srv=setrandvar\_new(2);  
setrandvar\_buildsample(srv,names(i),maxdegree);  
sampling=setrandvar\_getsample(srv);  
np=size(sampling,"r");  
setrandvar\_destroy(srv);  
mprintf(" %s, np=%d\n",names(i),np)  
end  
end

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| maxdegree | Quadrature | Petras | SmolyakGauss | SmolyakFejer | SmolyakTrapeze | SmolyakClenshawCurtis |
| 1 | 4 | 5 | 5 | 5 | 5 | 5 |
| 2 | 9 | 9 | 14 | 17 | 17 | 13 |
| 3 | 16 | 17 | 30 | 49 | 49 | 29 |
| 4 | 25 | 33 | 55 | 129 | 129 | 65 |
| 5 | 36 | 33 | 91 | 321 | 321 | 145 |
| 6 | 49 | 65 | 140 | 769 | 769 | 321 |
| 7 | 64 | 97 | 204 | 1793 | 1793 | 705 |
| 8 | 81 | 97 | 285 | 4097 | 4097 | 1537 |
| 9 | 100 | 161 | 385 | 9217 | 9217 | 3329 |
| 10 | 121 | 161 | 506 | 20481 | 20481 | 7169 |

[<< Theory](nisp_theory.html)

scilab-5.5.2

[Scilab Enterprises](http://www.scilab.org/)  
Copyright (c) 2011-2014 (Scilab Enterprises)  
Copyright (c) 1989-2012 (INRIA)  
Copyright (c) 1989-2007 (ENPC)  
with [contributors](https://www.scilab.org/development/contribute/contributors)

Last updated:  
Tue Jan 01 15:54:19 CET 2019