BAUDIN Michael

De:	trac@openturns.org
Envoyé:	jeudi 27 octobre 2016 18:12
Cc:	bugs@openturns.org
Objet:	[OpenTURNS] #863: Mixed Normal copula

#863: Mixed Normal copula

```
------------------------------------+-----------------------------
      Reporter:  michael.baudin@...  |       Owner:  mrbugs
          Type:  enhancement        |      Status:  new
      Priority:  unspecified        |   Milestone:  openturns-1.8
     Component:  unknown            |     Version:  1.8rc1
      Keywords:                     |  Architecture:  All
Operating system:  All              |     Symptom:  None
------------------------------------+-----------------------------
```

The starting point is the case of a couple of variables with a gaussian copula. All I need to have is a sample of this distribution.

```
{{{
# Case 1 : OK
R = ot.CorrelationMatrix(2, [1., 0.5, 0.5, 1.])  copula = ot.NormalCopula(R)  marginals = [ot.Normal(), ot.Uniform()]
mydist=ot.ComposedDistribution(marginals)
X=mydist.getSample(10)
}}}
```

In my practical test, I obviously will have more than two variables.
Moreover, there are two possible situations :
* either two variables are equal, e.g. X1=X2,
* or two variables are correlated with a Gaussian copula.

The second case works fine, but not the first where I have either
corr(X1,X2)=+1 or corr(X1,X2)=-1. Consider for example the case where corr(X1,X2)=+1. The script :

```
{{{
R = ot.CorrelationMatrix(2, [1., 1., 1., 1.])  copula = ot.NormalCopula(R)  }}}  produces :
{{{
TypeError: InvalidArgumentException : The correlation matrix must be  definite positive  }}}
```

Actually, the error message is correct, since the matrix R has two eigenvalues : 0 and 2. However, I guess that it could be expected that the NormalCopula takes that degenerate case into account and manages the copula the same way my workaround does :

```
{{{
import openturns as ot
import numpy as np
N=10
X1=marginals[0].getSample(N)
p1=marginals[0].computeCDF(X1)
pp1=np.array(p1)
X2=marginals[1].computeQuantile(pp1[:,0])
X=ot.NumericalSample(N,2)
X[:,0]=X1
X[:,1]=X2
}}}
```

But the previous script is not satisfactory for two reasons.
* It is not clear for me that this is actually a rigorous Gaussian copula.
* The correlation between X1 and X2 could be +1 (as in the workaround) or negative. Modifying the workaround to take that into accound is easy : if
corr(Xi,Xj)=-1 instead of +1, just consider 1-p1 instead of p1.
* If there are more than 2 variables, searching for the dependent variables is not trivial anymore. For example :
{{{
>>> R = ot.CorrelationMatrix(3, [1., 1., 0., 1., 1., 0., 0., 0., 1.]) >>> print(R) [[1 1 0]
 [1 1 0]
 [0 0 1]]
}}}
In this cas, the subset of variables associated with a -1 or +1 correlation should be removed, so that the remaining regular gaussian variables can be simulated. Once done, the same workaround as previously could be used. This decomposition could be based on the SVD, for example (with a threshold, if required).

So :
* either the NormalCopula could be extended to take that into account,
* or a new "NormalCopulaWithFullCorrelations" could be created.

--
Ticket URL: <http://trac.openturns.org/ticket/863>
OpenTURNS <http://trac.openturns.org>
OpenTURNS is an Open source initiative to Treat Uncertainties, Risks'N Statistics in a structured industrial approach