Tools for sampling Multivariate Archimedean Copulas

Mario R. Melchiori CPA†
Universidad Nacional del Litoral
Santa Fe – Argentina
March 2006

Abstract

A hurdle for practical implementation of any multivariate Archimedean copula was the absence of an efficient method for generating them. The most frequently used approach named conditional distribution one, involves differentiation step for each dimension of the problem. For this reason, it is not feasible in higher dimension. Marshall and Olkin proposed an alternative method, which is computationally more straightforward than the conditional distribution approach. We present the tools necessary for understand it and use it. We introduce the Laplace Transform and its role in the generation of multivariate Archimedean copulas. In order to cover the gap between the theory and its practical implementation VBA code and R one are provided.‡

The prices do not follow models.
We invent models to describe prices.

Glyn Holton – www.contingencyanalysis.com

† BICA Coop. E.M.Ltda. 25 de Mayo 1774 – Santo Tomé – SANTA FE- Argentina - E-mail: Mario.Melchiori@gmail.com
The opinions expressed in this paper are those of the author and do not necessarily reflect views shared by BICA Coop. E.M.Ltda. or its staff.

‡ The codes are available from the author on request.
Introduction

A hurdle for practical implementation of any multivariate Archimedean copula was the absence of an efficient method for generating them. The most frequently used approach named conditional distribution one, involves differentiation step for each dimension of the problem. For this reason, it is not feasible in higher dimension. Marshall and Olkin proposed an alternative method, which is computationally more straightforward than the conditional distribution approach. A disadvantage is that it requires the generation of an additional variable. For bivariate applications, this means generating 50% more uniform random variables, but for higher dimension that drawback is negligible.

We describe now the algorithm to generate multivariate Archimedean copula. The $d$-dimensional Archimedean copulas may be written as:

$$C(u_1,\ldots,u_d) = \phi^{-1}(\phi(u_1) + \cdots + (u_d)),$$

Where $\phi$ is a decreasing function known as the generator of the copula and $\phi^{-1}$ denotes the inverse of the generator (Frees and Valdez; McNeil et al.). In addition, $\phi^{-1}$ is equal to the inverse of the Laplace transform of a distribution function $G$ on $\mathbb{R}^+$ satisfying $G(0) = 0$, the following algorithm can be used for simulating from the copula:

**Algorithm 1:**

1. Simulate $d$ independent uniform variable $u_i$ for $i = 1,\ldots,d$
2. Simulate a variable $Y$ with distribution function $G$ such that the Laplace transform of $G$ is the inverse of the generator.
3. Define $s_i = -\ln(u_i)/Y$ for $i = 1,\ldots,d$
4. Define $X_i = \phi^{-1}(s_i)$ for $i = 1,\ldots,d$

Then $X_1,\ldots,X_d$ have Archimedean copula dependence structure.

<table>
<thead>
<tr>
<th>Name</th>
<th>Clayton</th>
<th>Gumbel</th>
<th>Frank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generator</td>
<td>$\phi(t) = (t^{-\theta} - 1)$</td>
<td>$\phi(t) = (-\ln t)^\theta$</td>
<td>$\phi(t) = -\ln\left(\frac{e^{-\theta t}}{1 - e^{-\theta}}\right)$</td>
</tr>
<tr>
<td>Inverse Generator</td>
<td>$\phi^{-1}(s) = (1 + s)^{\frac{1}{\theta}}$</td>
<td>$\phi^{-1}(s) = \left(-e^s\right)^{\frac{1}{\theta}}$</td>
<td>$\phi^{-1}(s) = -\frac{1}{\theta}\ln\left[1 - e^{-s}\right](1 - e^{-s})$</td>
</tr>
<tr>
<td>Parameter</td>
<td>$\theta &gt; 0$</td>
<td>$\theta \geq 1$</td>
<td>$\theta \in \mathbb{R} \setminus {0}$</td>
</tr>
<tr>
<td>Y-Distribution</td>
<td>Gamma $\left(\frac{1}{\theta}, 1\right)$</td>
<td>Stable $\left(\alpha, \beta, \gamma, \delta\right)$</td>
<td>Logarithmic series on $\mathbb{N}^+$ with $\alpha = (1 - e^{-\theta})$</td>
</tr>
<tr>
<td>Density of $Y$</td>
<td>$\frac{1}{\Gamma\left(\frac{1}{\theta}\right)} e^{-\theta y^{1-\theta}/\theta}$</td>
<td>(No closed form is known)</td>
<td>$P[Y = k] = \frac{-1}{\ln(1 - \alpha)} \frac{\alpha^k}{k}$</td>
</tr>
</tbody>
</table>


Though the density of a $\alpha - \text{stable}$ distribution’s closed form is not known Nolan proposed the following simulation algorithm for generating random variables $\alpha - \text{stable}$ distributed:
Algorithm 2:
1. Simulate an uniform variable $\Theta \sim U\left(\frac{-\pi}{2}, \frac{\pi}{2}\right)$
2. Simulate an exponentially distributed variable $W$ with mean 1 independently of $\Theta$.
3. Set $\theta_0 = \arctan(\beta \tan(\pi \alpha / 2)) / \alpha$.
4. Compute $Z \sim St\left(\alpha, \beta, 1, 0\right)$

\[
Z = \frac{\sin \alpha (\theta_0 + \Theta)}{(\cos \alpha \theta_0 \cos \Theta)^{1/\alpha}} \left[\cos \left(\alpha \theta_0 + (\alpha - 1)\Theta\right) / W\right]
\]
\[
Z = \frac{2}{\pi} \left[\frac{2}{\pi} + \beta \Theta\right] \tan \Theta - \beta \ln \left(\frac{\pi W \cos \Theta}{\pi \beta + \beta \Theta}\right)
\]

5. Compute $X \sim St\left(\alpha, \beta, \gamma, \delta\right)$:

\[
X = \gamma Z + \delta \quad \alpha = 1
\]
\[
X = \gamma Z + \left(\delta + \frac{\beta^2}{2} \gamma \ln(\gamma)\right) \quad \alpha = 1
\]


Algorithm 3:

Set $c = \ln(1 - \alpha)$

Simulate an uniform variable $V \in [0, 1]$.

IF $V \geq c$ then set $X = 1$
ELSE

Simulate an uniform variable $U \in [0, 1]$.

Set $q = 1 - e^{cU}$

CASE

$V \leq q^2$ : set $X = \text{int} \left[1 + \frac{\ln(V)}{\ln(q)}\right]$

$q^2 < V \leq q$ : set $X = 1$.

$V > q$ : set $X = 2$.

$X$ is Logarithmic Series distributed.

No other algorithm is necessary because the most of software have a built-in function that generates random variables Gamma distributed, such as GammaInv function on Excel and rgamma on R.

Laplace Transform

The Laplace Transform of a function $f(t)$ is denoted by $\mathcal{L}[f(t)]$ and defined by:

\[
\mathcal{L}[f(t)] = \int_0^\infty f(t)e^{-st}dt
\]

The Laplace transform $\mathcal{L}[f(t)]$ of $f(t)$ is, therefore, a function of the variable $s$ and is commonly denoted by $f(s)$.

The inverse Laplace transform is written as:
\[ \mathcal{L}^{-1}[f(s)] = f(t). \]

Table of Laplace Transform pairs, which are of interest for our work:

<table>
<thead>
<tr>
<th>Function ( f(t) )</th>
<th>Laplace Transform ( f(s) = \int_0^\infty f(t)e^{-st}dt )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 for ( t &gt; 0 )</td>
<td>( \frac{1}{s} )</td>
</tr>
<tr>
<td>( e^{st} )</td>
<td>( \frac{1}{s-g} ) for ( s &gt; g )</td>
</tr>
<tr>
<td>( \frac{1}{\Gamma(\frac{1}{\theta})} e^{-t/(\theta-\theta)} )</td>
<td>( (1+s)^{-\frac{1}{\theta}} ) for ( \theta &gt; 0 )</td>
</tr>
<tr>
<td>( P[T = t] = \frac{-1}{\ln(1-\alpha)} \frac{\alpha^t}{t} )</td>
<td>( \frac{-1}{\theta} \ln[1 - e^{-s(1 - e^{-\theta})}] ) for ( \alpha = (1 - e^{-\theta}) )</td>
</tr>
</tbody>
</table>

(No closed form is known) \( e^{-s^{\frac{1}{\theta}}} \) for \( \theta \geq 1 \)

Table II – Laplace Transform pairs

From an economic point of view we immediately recognize the Laplace transform as the present value of a stream of returns \( f(t) \) at the interest rate \( s \). The present value of cash flows of \$1, \( f(t) = 1 \), paid annually and perpetually at the continuous interest rate \( s \) is \( \mathcal{L}[f(s)] = \int_0^\infty e^{-st}dt = \frac{1}{s} \). On the other hand, the present discounted value at the continuous interest rate \( s \) of a cash flow growing at the rate \( g \), \( f(t) = e^{gt} \), is \( \mathcal{L}[f(s)] = \int_0^\infty e^{gt}e^{-st}dt = \frac{1}{s-g} \). If \( f(t) \) is continuous and differentiable at all \( t \geq 0 \), then it is possible to recover \( f(t) \) from \( \mathcal{L}[f(s)] \) through the inverse transformation:

\[ \mathcal{L}^{-1}[f(s)] = f(t) = \lim_{b \to \infty} \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} e^{st} \mathcal{L}[f(s)]ds, a \geq \alpha. \]

This has the economic meaning that if we know the present discounted value of a stream of returns at every interest rate, we can recover the whole pattern of the stream of returns.

**Experiment I**

We conduct an experiment for seeing practically in what manner, by knowing the Laplace transform of a function, we can recover it. We will do it here, but you can reproduce it on your own computer. For doing this, we need an approach that implements numerically the inverse transformation \( \mathcal{L}^{-1}[f(s)] \). We will use the Euler\(^3\) method. Readers interested in deepening this subject should consult the following link:

[http://www.columbia.edu/~ww2040/LaplaceInversionJoC95.pdf](http://www.columbia.edu/~ww2040/LaplaceInversionJoC95.pdf)

\(^3\) In Appendix A, we provide a VBA code that implements numerically the inverse Laplace transform by employing the Euler method. The Laplace transform is specified by the variable \( Fs \) in the function \( Rf \). Inserting different transforms here can solve different problems.
The experiment consists of computing the flow of returns by knowing its net present value using the inverse Laplace transform. As we already saw, the net present value of a stream of returns that grows to the continuous rate $g$, represented by the function $f(t) = e^{gt}$, discounted to the continuous interest rate $s$ to be $\frac{1}{s - g}$. Also $\frac{1}{s - g}$ is the Laplace Transform of the function $f(t) = e^{gt}$ as well. So that, we can calculate the stream of returns that grows at rate $g^2$ discounted to the continuous interest rate $s$ using $e^{gt}$ or employing the Inverse Laplace Transform of $\frac{1}{s - g}$.

In the VBA alluded to the Laplace transform $\frac{1}{s - g}$, is specified by the variable $Fs$ in the function $Rf$. The VBA should look as follow:

```vba
Function Rf(ByVal X, ByVal Y) As Double
    s = imsum(X, improduct("i", Y))
    g = 0.02
    Fs = imdiv(1, (imsum(s, - g)))
    Rfs = imreal(Fs)
    Rf = Rfs
End Function
```

So in Excel’s language $e^{gt}$ is represented as $=\text{Exp}(0.02*A1)$ for $t = 1$, $=\text{Exp}(0.02*A2)$ for $t = 2$ and so on. On the other hand, the Inverse Laplace Transform of $\frac{1}{s - g}$ as $=\text{InverseLaplace}(A1)$ for $t = 1$, $=\text{InverseLaplace}(A2)$ for $t = 2$ and so on.
Experiment II

The previous experiment was a warm-up for this one, which has more relation with our research. The point 2 of the Algorithm 1 requires simulating a variable \( Y \) with distribution function \( G \) such that the Laplace transform of \( G \) is the inverse of the generator. In Table I, we can see in the Clayton case that the probability density function \( G \) is \( \text{Gamma}(\frac{1}{\theta}, 1) \). So that, the inverse Laplace transform of the inverse generator \( (1 + s)^{-\frac{1}{\theta}} \) for \( \theta > 0 \) equals to the Probability Density Function of a \( \text{Gamma}(\frac{1}{\theta}, 1) \) distribution. For doing this in the VBA alluded to, the Laplace transform \( (1 + s)^{-\frac{1}{\theta}} \), is specified by the variable \( Fs \) in the function \( Rf \) in complex notation. The VBA should look as follow:

```vba
Function Rf(ByVal X, ByVal Y) As Double
    s = imsum(X, improduct("i", Y))
    theta = 1.84
    Fs = impower(imsum(1, s), -1 / theta)
    Rfs = imreal(Fs)
    Rf = Rfs
End Function
```

So that the Laplace transform \( (1 + s)^{-\frac{1}{\theta}} \) in Excel’s language is represented by the function \( \text{=InverseLaplace}(A1) \) for \( t = 1 \), \( \text{=InverseLaplace}(A2) \) for \( t = 2 \) and so on. Also, the Probability Density Function of a \( \text{Gamma}(\frac{1}{\theta}, 1) \) distribution equals to \( \frac{1}{\Gamma(\frac{1}{\theta})} e^{-\frac{1}{\theta}} t^{-\frac{1}{\theta}} \) or in Excel’s language \( =1/(\text{Exp(Gamma.Ln(1/1.84))}*\text{Exp}(-A1)*A1^((1-A1)*A1^((1-A1)*A1^((1-A1)*A1^((1-A1)/1.84))) or =GammaDist(A1,1/1.84,1,0)) \) for \( t = 1 \) and, \( =1/(\text{Exp(Gamma.Ln(1/1.84))}*\text{Exp}(-A2)*A2^((1-A2)/1.84)) or =GammaDist(A2,1/1.84,1,0)) \) for \( t = 2 \) and so on.

---

2 In this example the variable \( \theta \) equal to 1.84.
We invite readers to conduct a third experiment that computes the probability density function in the Frank case using both the function $P[Y = k] = \frac{-1}{\ln(1 - \alpha)} \frac{\alpha^k}{k}$ and the inverse Laplace Transform of 

$$-\frac{1}{\theta} \ln[1 - e^{-s}(1 - e^{-\theta})]$$ for $\alpha = (1 - e^{-\theta}).$

### Conclusions

There is clear evidence that equity returns have unconditional fat tails, to wit, the extreme events are more probable than anticipated by normal distribution, not only in marginal but also in higher dimensions. This is important both for market risk models as credit risk one, where equity returns are used as a proxy for asset returns that follow a multivariate normal distribution, and, therefore, default times have a multivariate normal dependence structure as well. Other than normal distribution should be used both in marginal as joint distributions. To overcome these pitfalls, the concept of copula emerges. A hurdle for practical implementation of any multivariate Archimedean copula was the absence of an efficient method for generating them. The most frequently used approach named conditional distribution one, involves differentiation step for each dimension of the problem. For this reason, it is not feasible in higher dimension. Marshall and Olkin proposed an alternative method, which is computationally more straightforward than the conditional distribution approach. We present the tools necessary for understand it and use it. We introduce the Laplace Transform and its role in the generation of multivariate Archimedean copulas. In order to cover the gap between the theory and its practical implementation VBA code and R one are provided.

### References


Schönbucher, Philipp J. (August 2002) *Taken to the Limit: Simple and Not-so-simple Loan Loss Distributions* – Bonn University (http://www.deaultrisk.com/pdf_files/Taken_To_the_Limit~Smpl_n_Nt~s~mpl_Ln_Lss_Dstribns.pdf)
Appendix A

(http://www.columbia.edu/~ww2040/LaplaceInversionJoC95.pdf)

Function InverseLaplace(t As Double) As Double

Const Pi = 3.14159265358979
Dim SU(13), C(12)
m = 11

For n = 0 To m
    C(n + 1) = Application.WorksheetFunction.Combin(m, n)
Next

A = 18.4
Ntr = 15
u = Exp(A / 2) / t
X = A / (2 * t)
h = Pi / t

Sum = Rf(X, 0) / 2
    For n = 1 To Ntr
        Y = n * h
        Sum = Sum + (-1) ^ n * Rf(X, Y)
    Next

SU(1) = Sum
    For k = 1 To 12
        n = Ntr + k
        Y = n * h
        SU(k + 1) = SU(k) + (-1) ^ n * Rf(X, Y)
    Next

Avgsu = 0
Avgsu1 = 0
    For j = 1 To 12
        Avgsu = Avgsu + C(j) * SU(j)
        Avgsu1 = Avgsu1 + C(j) * SU(j + 1)
    Next
Fun = u * Avgsu / 2048
Fun1 = u * Avgsu1 / 2048

InverseLaplace = Fun1
errt = Abs(Fun - Fun1) / 2

End Function

Function Rf(ByVal X, ByVal Y) As Double
s = imsum(X, improduct("i", Y))
Fs = imdiv(1, (imsum(s, -0.02)))
Rfs = imreal(Fs)
Rf = Rfs
End Function