Universität Zürich[UZH]

# Assignment 1, due 18.10.2022

## Statistical Foundations for Finance (Mathematical and Computational Statistics with a View Towards Finance and Risk Management)

Prof. Dr. Marc Paolella


presented by

**Dennis Arend: 21-742-135**

**Ernest Digore: 21-727-896**

**Kilian Sennrich: 18-051-060**



**Abstract**

The purpose of the assignment is to study the behaviour of the stable Paretian distribution, to implement certain methodologies of generating such distribution, and to compute tail risk measures such as Value at Risk and Expected Shortfall using the tools from (Stoyanov et al, 2006). We discuss the accuracy of the analytic approximations compared to the theoretical representation. Further, we take a look at the convolution of two stable distributions by employing such computation methods as Inversion and Convolution formulae. The work has been fulfilled using Matlab programming software with the detailed codes provided in each section.

# Contents

# 1 Generate Density for Stable Parameters

We begin our exploration of the stable Paretian distribution with a simple comparison of a stable random variables' probability density function, computed using both the theoretical probability density function (PDF), and a simulation approach in combination with a kernel density smoother.

The stable random variable examined is characterized by the following stable Paretian distribution parameter values: $\alpha = 1.7$, $\beta = -0.4$, $\sigma = 2$ and $\mu = 0.3$. The results are illustrated in Figure 1, with the red line showing the random variables' actual density, computed using the analytical approach of J.P. Nolan for general stable distributions. Nolan (in 1997) was able to derive expressions in the form of integrals, which were based on the characteristic function for standardized stable random variables. The blue dashed line in Figure 1 shows the kernel density estimate based on 1'000'000 replications. The code for Figure 1 can be found in Code-Section 1. One can see how the resulting PDF's match each other, and with help of the "tic" and "toc" functionality of Matlab we can see how the analytical method (in this case), is much faster than the simulation approach based on 1'000'000 replications (Analytical approach: 0.219 sec; Simulation approach: 0.688 sec).

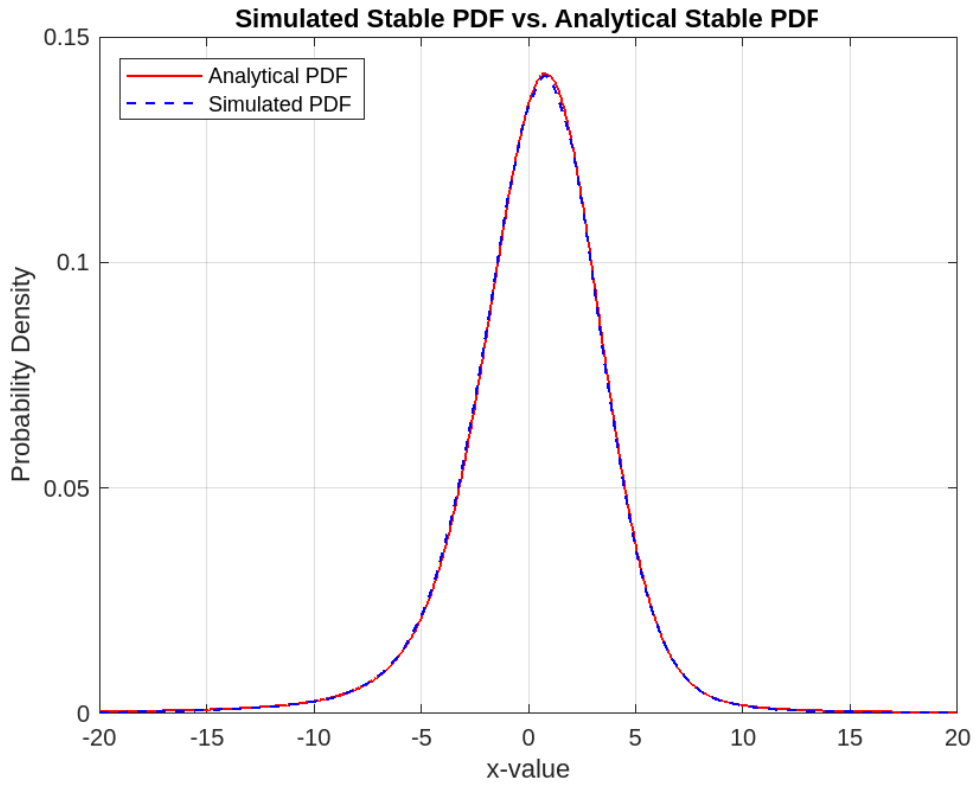


Figure 1: The Stable probability density function calculated once using the analytical method based on J.P. Nolan's method (red), and once using a simulation approach (blue).

```
% Parameters
a=1.7;b=-0.4;c=2;d=0.3;seed=2;nobs=1e6;
limit=20; steps = 1000; xvec = linspace(-limit,limit,steps);

% Simulation using Random Variates
tic; x = stabgen(nobs,a,b,c,d,seed);
[fsimulated,~] = ksdensity(x,xvec);
toc
% Analytical Method based on Nolan
tic; fanalytic = asymstab(xvec, a, b, c, d);
toc

figure
set(gca, 'fontsize',14), plot(xvec,fanalytic,'r','LineWidth',1), hold on
plot(xvec,fsimulated,'b--','LineWidth',1), grid on
title("Simulated Stable PDF vs. Analytical Stable PDF")
legend("Analytical PDF","Simulated PDF","Location","northwest"),
xlim([-limit,limit]), ylabel("Probability Density"), xlabel("x-value")

function x = stabgen(nobs,a,b,c,d,seed)
% Generates a random sample of size nobs from the S_a,b(d,c) distribution
if nargin<3, b=0; end, if nargin<4, c=1; end
if nargin<5, d=0; end, if nargin<6, seed=rand*1000; end
z=nobs; rng(seed,'twister'),V=unifrnd(-pi/2,pi/2,1,nobs);
rng(seed+42,'twister'),W=exprnd(1,1,nobs);

if a==1,   x=(2/pi)*(((pi/2)+b*V).*tan(V)-b*log((W.*cos(V))./((pi/2)+b*V))); x=c*x+d
    -(2/pi)*log(c)*c*b;
else     Cab=atan(b*tan(pi*a/2))/(a);    Sab=( 1 + b^2*(tan((pi*a)/2))^2 ) ^(1/(2*a));
    A=(sin(a*(V+Cab)))./((cos(V)).^(1/a));    B0=(cos(V-a*(V+Cab)))./W;
    B=(abs(B0)).^((1-a)/a); x=Sab*A.*(B.*sign(B0)); x=x*c+d;
end

function[f,F]=asymstab(xvec,a,b,c,d)
% PROGRAM LISTING A.3
bordertol = 1e-8;lo = bordertol;hi = 1-bordertol;tol = 1e-7;xl = length(xvec);
F = zeros(xl,1);f = F;
for loop = 1:length(xvec)
    x = xvec(loop);
    f(loop)=integral(@(uvec) fff(uvec,x,a,b,c,d,1), lo, hi)/pi;
    if nargout > 1,   F(loop) = 0.5-(1/pi)*integral(@(uvec) fff(uvec,x,a,b,c,d,0), lo, hi)/pi; end
end
end

function I = fff(uvec,x,a,b,c,d,dopdf)
I=zeros(size(uvec));
for ii=1:length(uvec)
    u=uvec(ii);    t=(1-u)/u;    cf=exp(-(c^a*(abs(t))^a)*(1-1i*b*sign(t)*tan(pi*a/2))+ 1i*d*t);
    z=exp(-1i*t*x).*cf;
    if dopdf==1, g=real(z); else    g=imag(z)./t; end
    I(ii)=g*u^(-2);
end
end
```

Code-Section 1: Matlab code for Figure 1.

## 2 Convolution of Two Stable Random Variables

As a second exercise, the goal is to confirm the properties of stable distributions, namely being closed under addition. Addition in this context refers to the convolution of two independent stable random variables which can differ in their parameters: alpha ($\alpha$), beta ($\beta$), location ($\mu$) and scale ($\sigma$ or also "c"). In case of identical $\alpha$ values between the two random variables, the Convolution Formula for two stable's (given identical alphas) is given by:

$$\text{Given } X_i \overset{\text{ind}}{\sim} S_{(\alpha,\beta_i)}(\mu_i, \sigma_i) \text{ and } S = \sum_{i=1}^{n} X_i \quad :$$

$$S \sim S_{(\alpha,\beta)}(\mu, \sigma), \quad \text{where} \quad \mu = \sum_{t=1}^{n} \mu_i \quad , \quad \sigma = \left( \sum_{t=1}^{n} \sigma_t^{\alpha} \right)^{1/\alpha} \quad , \quad \beta = \frac{\sum_{t=1}^{n} (\beta_t \times \sigma_t^{\alpha})}{\sum_{t=1}^{n} \sigma_t^{\alpha}}$$

This however only applies if both random variables have the same $\alpha$ parameters (Section 3 covers the case of differing tail risk parameters $\alpha$). For now, we work with the following parameters of the two independent stable random variables, allowing us to apply the above stated Convolution Formula:

$$\begin{aligned} X1: &\quad \alpha_1 = 1.7, \quad \beta_1 = -0.4, \quad \sigma_1 = 2.0, \quad \mu_1 = -0.5 \\ X2: &\quad \alpha_2 = 1.7, \quad \beta_2 = -0.5, \quad \sigma_2 = 2.2, \quad \mu_2 = -0.6 \end{aligned}$$

The result, shown in Figure 2, provides a graphic illustration of the property that the convolution of such two independent stable random variables in fact again admits a stable probability density. As done in Exercise 1, the result is illustrated for both, the analytical method based on the above formula, and the kernel density using a simulation over 1'000'000 iterations. The code for Figure 2 is given in Code-Section 2. Functions *stabgen* and *asymstab* are identical to Code-Section 1. Once again, we can verify that both methods lead to near identical results.

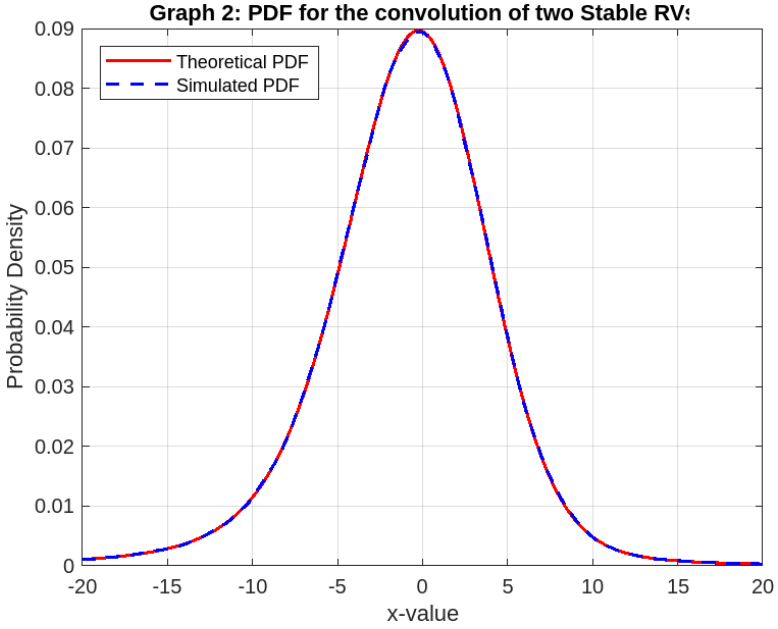


Figure 2: Probability Density Functions of two convoluted independent Stable random variables, once executed using the simulation approach (blue) and once the analytical approach based on Nolan (red).

```
%% 2. Confirm Convolution Property of stable distributions with same alpha
% Random Variable parameters
a1 = 1.7; b1 = -0.4; c1 = 2;    d1 = -0.5;
a2 = a1;  b2 = -0.5; c2 = 2.2; d2 = -0.6;
seed = 1;nobs = 1e6; limit = 20; steps = 500; xvec = linspace(-limit,limit,steps);

% Convolution Formula based on Slide 535 in notes
a = a1; b = (b1*(c1^a) + b2*(c2^a))/(c1^a + c2^a);
c = (c1^a + c2^a).^(1/a); d = d1 + d2;

% Analytical method using Program Listing A.3
fanalytic = asymstab(xvec, a, b, c, d);

% Simulation
x = stabgen(nobs,a,b,c,d,seed);
[fsimulated,~] = ksdensity(x,xvec);

figure
set(gca, 'fontsize',14)
plot(xvec,fanalytic,'k-',"LineWidth",1)
hold on
plot(xvec,fsimulated,'b--',"LineWidth",1)
grid on
title("Graph 2: PDF for the convolution of two Stable RVs")
legend("Theoretical PDF","Simulated PDF","Location","northwest")
xlim([-limit,limit])
ylabel("Probability Density")
xlabel("x-value")
```

Code-Section 2: Matlab code for Figure 2.

# 3 Convolution of Independent RV 2.0

As an extension of the convolution property shown in Section 2, we now move towards a case which is much closer to real-life: convoluting two independent stable random variables with *different* values of the tail index $\alpha$. Such cases often exist in the context of financial assets, which naturally have different tail risks.
The parameters chosen for the two independent stable random variables are:

$$\begin{aligned} X1: &\quad \alpha_1 = 1.6, \quad \beta_1 = 0, \quad \sigma_1 = 1, \quad \mu_1 = 0 \\ X2: &\quad \alpha_2 = 1.8, \quad \beta_2 = 0, \quad \sigma_2 = 1, \quad \mu_2 = 0 \end{aligned}$$

In the following, we will try and produce the results using four different methods, namely:

1. **Inversion Formula**
2. **Simulation & Kernel Density**
3. **Convolution Formula**
4. **Convolution Formula (Matlab)**

The fourth method is identical to the third, with the only difference being that the fourth method utilizes a predefined Matlab function "*conv*".

**1. Inversion Formula**
Let $X$ be a univariate discrete random variable with support $\{x_j\}_{j=1}^{\infty}$, probability mass function $f_X$, and characteristic function $\varphi_X$. The inversion formula states that, for $j \geq 1$:

$$f_X(x_j) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \mathrm{e}^{-itx_j} \varphi_X(t) \mathrm{d}t$$

For numeric work one can use the counterpart:

$$f_X(x_j) = \frac{1}{\pi} \int_{0}^{\pi} \mathrm{Re}\left[\mathrm{e}^{-itx_j} \varphi_X(t)\right] \mathrm{d}t$$

The characteristic function of $X \sim S_\alpha(\mu, \sigma)$ is given by

$$\varphi_X(t; \alpha) = \exp\{|t|^\alpha\}$$

Using these results, we can define the convolution of two independent variables $X_1 \sim S_{\alpha_1,\beta_1}(\mu_1, \sigma_1)$ and $X_2 \sim S_{\alpha_2,\beta_2}(\mu_2, \sigma_2)$ as:

$$f_X(x_j) = \frac{1}{\pi} \int_{0}^{\pi} \mathrm{Re}\left[\mathrm{e}^{-itx_j} \exp\{|t|^\alpha\}\right] \mathrm{d}t$$

This formula represents the analytical approach to the task, thus should lead to the most accurate result (see Figure 3, black plot) and will serve as the benchmark to which we compare the alternative methods in Figure 3.

## 2. Simulation & Kernel Density

First, we individually simulate the two stable distributions using 1'000'000 iterations each. Further, we perform a simple sum of the samples generated for the two vectors $X_1$ and $X_2$, and compute the kernel density of this sum, which should accurately represent the convolution of the two generated stable distributions. One can verify this by implementing a regression analysis to obtain the parameter estimates. Intuitively, alpha should be close to 1.7, beta to 0, sigma to 2 and mu to 0, which is actually the case.

## 3. Convolution Formula

The Convolution Formula for independent stable random variables with *differing* tail risk parameters is defined as the integral of the product of the two functions (densities) after one is reflected about the y-axis and shifted. The formulas given in Section 2 do not apply anymore.

$$f_S(s) = \int_{-\infty}^{\infty} f_X(x)\, f_Y(s-x)\, \mathrm{d}x$$

We implement this via a simple function represented in the Code-Section 3, once using Matlab's built-in *"conv"* - function, and once using a manual implementation of the above formula.
The results using the four methods are illustrated in Figure 3, with the Inversion Formula result shown in black, the Simulation result shown in blue, the result based on the convolution formula in red and finally the result using Matlab's built-in *"conv"* - function in green dotted. The code used to generate Figure 4 is shown in Code-Section 3.

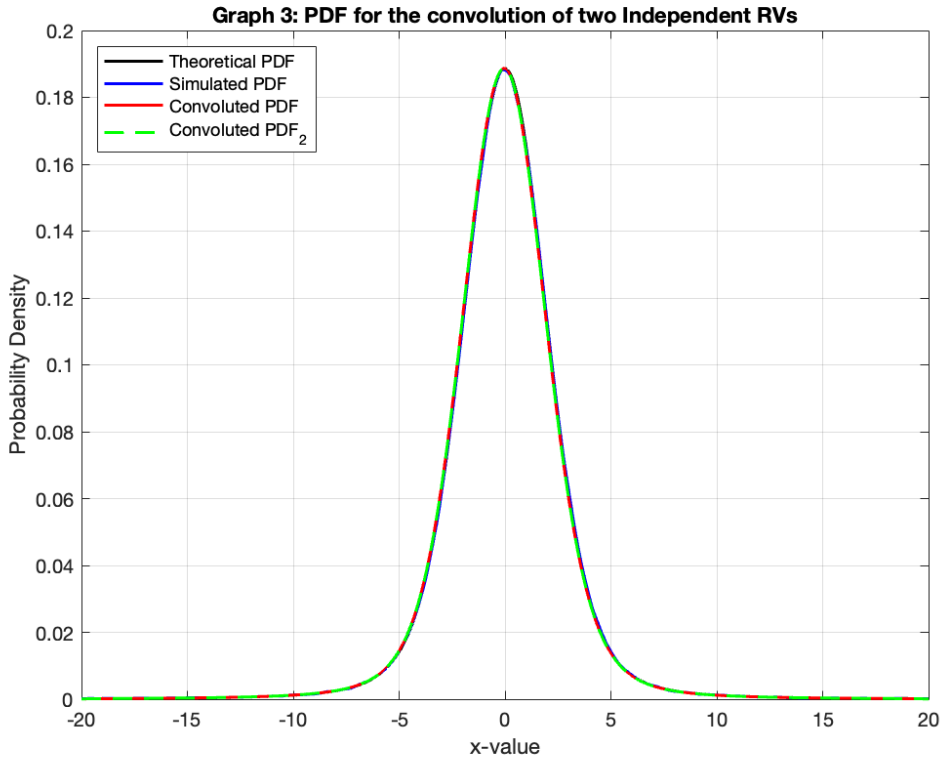


Figure 3: Comparison of the stable probability density functions of two convoluted independent stable random variables with different tail risks $\alpha_1 = 1.6$ and $\alpha_2 = 1.8$, computed using four different methods.

In addition to the above, we reproduce the results for a variation of alpha values, while once again utilizing all four available methods. The alphas are now chosen as $\alpha_1 = 1.5$ and $\alpha_2 = 1.9$, with the results shown in Figure 4. As there is no change in the necessary methods, the code remains the same as in Code-Section 3. Albeit the graphs are crowded with four lines, the main takeaway is that they almost perfectly fit each other, which is what we want to see from the four computation methods.

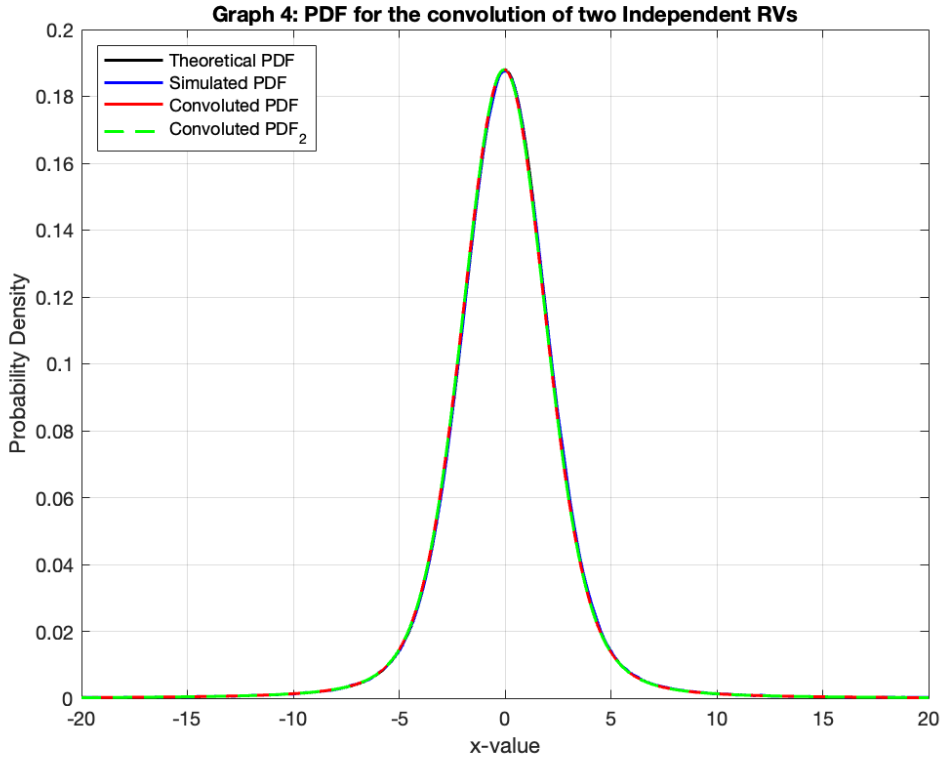


Figure 4: Comparison of the stable probability density functions of two convoluted independent stable random variables with $\alpha_1 = 1.5$ and $\alpha_2 = 1.9$, computed using four different methods.

```
%% 3. Convolution of Independent RV 2.0
% Random Variable parameters
a = 1.6; b = 0; c = 1;    d = 0; a1 = 1.8;   b1 = 0; c1 = 1; d1 = 0;
seed = 1;nobs = 1e6; limit = 20; steps = 500; xvec = linspace(-limit,limit,steps); dx = xvec(2)-xvec(1);

% Via Inversion Formula
% Where asymstab has been altered to compute the convolution between two stables by multiplying the
    characteristic functions of the two (see Appendix A.1)
fanalytic_c = asymstab3(xvec, a, b, c, d,a1,b1,c1,d1);

% Simulation
x1 = stablernd(a,b,c,d,nobs,1); % See Appendix A.2 for the function as Written by Rafal Weron
    (2010.04.26)
x2= stablernd(a1,b1,c1,d1,nobs,1);
x_c = (x1+x2);
[fsimulated_c,~] = ksdensity(x_c,xvec);

% Based on convolution formula A.146
pd1 = asymstab(xvec, a, b, c, d);
pd2 = asymstab(xvec, a1, b1, c1, d1);
x = pd1; h = pd2;
N = length(x); M = length(h);
Ny = N + M -1;
y = zeros(1,Ny);
for i = 1:N
    for k = 1:M
        y(i+k-1) = y(i+k-1) + h(k)*x(i);
    end
end
fconv= y*dx;   %adjusting the component with the mesh size
fconv=fconv(251:750);  %trimming the extra zeros created for the integration

% Based on "conv" command
fconv2 =conv(pd1,pd2,"same")*dx;

figure
set(gca, 'fontsize',14)
plot(xvec,fanalytic_c,'k-',"LineWidth",1.5)
hold on
plot(xvec,fsimulated_c,'b-',"LineWidth",1.5)
hold on
plot(xvec,fconv,'r-',"LineWidth",1.5)
hold on
plot(xvec,fconv2,'g--',"LineWidth",1.5)
grid on
title("Graph 3: PDF for the convolution of two Independent RVs")
legend("Theoretical PDF","Simulated PDF","Convoluted PDF","Convoluted
    PDF_2","Location","northwest")
xlim([-limit,limit])
ylabel("Probability Density")
xlabel("x-value")
```

Code-Section 3: Matlab code for Figure 3 and Figure 4.

# 4 Expected Shortfall

We now turn to the "Expected Shortfall" (ES). Using the Stoyanov et al. result (see Equation 1), the theoretical ES is computed as a function of the tail probability $\xi$, using the stable parameters $\alpha = 1.7$, $\beta = 0$, $\sigma = 1$ and $\mu = 0$.

For $\alpha > 1, \quad S \sim S_\alpha(0,1), \quad 0 < \xi < 1, \text{ and } \quad q_{S,\xi} = F_S^{-1}(\xi)$:

$$\mathrm{ES}(S;\xi) = \frac{1}{\xi}\,\mathrm{Stoy}\left(q_{S,\xi},\alpha\right) \tag{1}$$

where the tail component $\int_{-\infty}^{\sigma} x f_S(x;\alpha)\mathrm{d}x$ is

$$\mathrm{Stoy}(\sigma,\alpha) = \frac{\alpha}{\alpha-1}\frac{|\sigma|}{\pi}\int_0^{\pi/2} g(\theta)\exp\left(-|\sigma|^{\alpha/(\alpha-1)}v(\theta)\right)\mathrm{d}\theta,$$

and

$$g(\theta) = \frac{\sin(\alpha-2)\theta}{\sin\alpha\theta} - \frac{\alpha\cos^2\theta}{\sin^2\alpha\theta}, \quad v(\theta) = \left(\frac{\cos\theta}{\sin\alpha\theta}\right)^{\alpha/(\alpha-1)}\frac{\cos(\alpha-1)\theta}{\cos\theta}.$$

The first result, based on a tail probability of $\xi = 0.01$, is compared with the empirical ES achieved using the simulation method based on 1'000'000 stable variates. The Matlab-code used for these results can be found in Code-Section 4. As can be seen in Table 1, the simulated ES becomes more accurate with an increase in stable variates, with the difference decreasing from -0.5833 (1e4 variates) to 0.0313 (1e8 variates).

| **Iterations** | **1e4** | **1e5** | **1e6** | **1e7** | **1e8** |
|---|---|---|---|---|---|
| **Empirical ES** | -12.0546 | -11.6859 | -11.4263 | -11.2926 | -11.4400 |
| **Analytical ES** | -11.4713 | -11.4713 | -11.4713 | -11.4713 | -11.4713 |

Table 1: Expected Shortfall calculated based on Stoyanov et al. (Analytical ES), compared to the Empirical ES based on simulations using 1e4 up to 1e8 stable variates.

```
%% 4. Expected Shortfall

a=1.7;b=0;c=1;d=0; xi = 0.01; iter = 1e8;

% Analytical ES using Stoyanov et al
ES = asymstableES(xi,a,b,d,c,1);
X = ['Expected Shortfall by Stoyanov et al:',num2str(ES)];
disp(X)

% Simulation using method from p.445
x = stabgen(iter,a,b,c,d,1);
q = quantile(x,xi);
I = x(x<q);
ES_sim= mean(I);
X = ['Expected Shortfall by simulation:',num2str(ES_sim)];
disp(X)
```

Code-Section 4: Matlab code for the computation of the ES for a given tail probability $\xi$.

# 5 Expected Shortfall 2.0

Once again we take a look at the case of two convoluted independent stables, such that their tail index values, $\alpha_1 = 1.6$ and $\alpha_2 = 1.8$, are different, with the other parameters being chosen arbitrarily. We choose our two stable random variables to be characterized with the following parameters:

$$\begin{aligned} X1: &\quad \alpha_1 = 1.6, \quad \beta_1 = 0, \quad \sigma_1 = 1, \quad \mu_1 = 0 \\ X2: &\quad \alpha_2 = 1.8, \quad \beta_2 = 0, \quad \sigma_2 = 1, \quad \mu_2 = 0 \end{aligned}$$

Using these parameters, we simulate the empirical ES of their convolution for multiple tail probabilities: $\xi = 0.01, \quad 0.025$ and $0.05$ based on 1e8 (1'000'000'000) random variates each. The results, presented in Table 2, show how for an increase in tail probability $\xi$, the ES increases as well. In a second step, we estimate the *parameters* of a stable Paretian distribution, this time using a slightly smaller set of random variates, namely 1e7. As expected, the tail risk parameter of the convoluted random variables is roughly the mean of its two values (see Table 3), as we chose $\beta$ and $\mu$ to be 0, while keeping the scale $\sigma = 1$. Finally, we once again use the parameter estimates to perform the analytical calculation of the Expected Shortfall based on Stoyanov et al., giving the results shown in Table 4. The Matlab code used for these calculations is presented in Code-Section 5. A simple "for"-loop enables us to swiftly perform the same calculations once again for a different choice of $\alpha$, namely $\alpha_1 = 1.5, \alpha_2 = 1.9$. The results are listed in Table 2,3 and 4 in the second rows.

**Empirical ES**

| **$\xi$-Value** | **0.010** | **0.025** | **0.050** |
|---|---|---|---|
| $\alpha_1 = 1.6$, $\alpha_2 = 1.8$ | -23.9603 | -14.5695 | -10.2971 |
| $\alpha_1 = 1.5$, $\alpha_2 = 1.9$ | -27.5453 | -16.1800 | -11.1752 |

Table 2: Empirical ES (using 1e8 variates) of two convoluted stable random variables with different tail risk parameters $\alpha_1 = 1.6$ and $\alpha_2 = 1.8$, as well as $\alpha_1 = 1.5$ and $\alpha_2 = 1.9$.

**Parameter Estimates**

| | $\alpha$ | $\beta$ | $\mu$ (Location) | $\sigma$ (Scale) |
|---|---|---|---|---|
| $\alpha_1 = 1.6$, $\alpha_2 = 1.8$ | 1.696 | 0.002 | 2.009 | 0.001 |
| $\alpha_1 = 1.5$, $\alpha_2 = 1.9$ | 1.679 | 0.002 | 2.036 | 0.001 |

Table 3: Parameter estimates of the stable distribution underlying two convoluted stable random variables with different tail risk parameters $\alpha$.

**Analytical ES**

| **$\xi$-Value** | **0.010** | **0.025** | **0.050** |
|---|---|---|---|
| $\alpha_1 = 1.6$, $\alpha_2 = 1.8$ | -23.3539 | -14.3228 | -10.1742 |
| $\alpha_1 = 1.5$, $\alpha_2 = 1.9$ | -24.9659 | -15.1610 | -10.6752 |

Table 4: Analytical ES based on the previously estimates parameters of two convoluted stable random variables with different tail risk parameters $\alpha$.

```
%% 5. Expected Shortfall 2.0
% All alpha values
a= [1.6  1.8  ;  1.5   1.9];

for  l=1:2
    a1 = a(l,1); a2 = a(l,2);
     fprintf('\n\n ES results based on alpha1 = %.2f and alpha2 = %.2f:\n', a1,a2);

    % RV parameters and other Parameters
    % alpha values are  different ,  other  parameters are chosen to be  identical
    b = -0.4; c = 2;    d = -0.5;
    seed = 1;iter  = 1e6; limit  = 20; steps  = 500; xvec = linspace(-limit,limit,steps);

    xi  = [  0.01;0.025;    0.05];
    loops = size(xi,1);
    ES_stoyanov = zeros(size(xi));
    % Simulate X1+X2, and calculate its Expected shortfall for the given values  of  xi
    X1 = stabgen(iter, a1,  b,  c,  d,  seed);
    X2 = stabgen(iter, a2,  b,  c,  d,  seed);
    x_sum = X1 + X2;
    q = quantile(x_sum, xi);
    for  i  = 1:loops
        I  = x_sum(x_sum < q(i));
        ES_stoyanov(i) = mean(I);
         fprintf('ES via simulation for  tail  probability  xi=%.3f: %.4f\n', xi(i),ES_stoyanov(i));
    end


    % Using 1e7, we approximate the parameters of the sum's underlying stable
    % distribution
    iter  = 1e7;
    X1 = stabgen(iter, a1,  b,  c,  d,  seed);
    X2 = stabgen(iter, a2,  b,  c,  d,  seed);
    x_sum = X1 + X2;

    [a_roof,b_roof,c_roof,d_roof]  = stablereg(x_sum);
    fprintf("\n Estimated stable dist.  parameters are given by: " + ...
        "alpha=%.3f, beta=%.3f, scale=%.3f, location=%.3f\n\n",a_roof,b_roof,c_roof,d_roof);

    % Finally,  we use the parameter estimates to calculate  once again the
    % Expected Shortfall, this  time using Stoyanov et al.  method
    ES_stoyanov = zeros(size(xi));
    for  i  = 1:loops
        ES_stoyanov(i,1) = asymstableES(xi(i,1), a_roof,  b_roof,  d_roof,  c_roof  ,1);
         fprintf('ES via Stoyanov et al.  for  tail  probability  xi=%.3f: %.4f\n', xi(i,1),ES_stoyanov(i,1));
    end
end
```

Code-Section 5: Matlab code for the computation of the Expected Shortfall over two convoluted stable random variables using both the analytical approach from Stoyanov, and the simuation approach in calculating the ES.

# 6 Appendix

```
function [f,F] = asymstab3(xvec,a,b,c,d,a1,b1,c1,d1)
% [f,F] = asymstab(xvec,a,b)

% use this for plotting the integrand
%x=xvec; dopdf=1; uvec=0.001:0.001:0.999; I=fff(uvec,x,a,dopdf); plot(uvec,I)
if nargin<3, b=0; end
bordertol=1e-8; lo=bordertol; hi=1-bordertol; tol=1e-7;
xl=length(xvec); F=zeros(xl,1); f=F;
for loop=1:length(xvec)
  x=xvec(loop); dopdf=1;
  f(loop)= quadl(@fff,lo,hi,tol,[], x,a,b,c,d,a1,b1,c1,d1,1) / pi;
  if nargout>1
    F(loop)=0.5-(1/pi)* quadl(@fff,lo,hi,tol,[], x,a,b,c,d,a1,b1,c1,d1,0);
  end
end


function I=fff(uvec,x,a,b,c,d,a1,b1,c1,d1,dopdf);
subs=1;
for ii=1:length(uvec)
  u=uvec(ii);
  if subs==1, t =(1-u)/u; else, t =u/(1-u); end
  if a==1
    cf = exp( -abs(t)*c*( 1 + i*b*(2/pi)*sign(t) * log(t) ) +i*d*t )*exp( -abs(t)*c1*( 1 +
        i*b1*(2/pi)*sign(t) * log(t) ) +i*d1*t );
  else
    cf = exp( - ((abs(t))^a)*c^a *( 1 - i*b*sign(t) * tan(pi*a/2) ) +i*d*t)* exp( - ((abs(t))^a1)*c^a1
        *( 1 - i*b1*sign(t) * tan(pi*a1/2) ) +i*d1*t);
  end
  z = exp(-i*t*x) .* cf;
  if dopdf==1, g=real(z); else g=imag(z)./t; end
  if subs==1, I(ii)=g*u^(-2); else, I(ii)=g*(1-u)^(-2); end
end
```

Code-Section Appendix A.1: Alteration of asymstab.m to include the product of two characteristic functions with different parameters

```
function r = stablernd(alpha,beta,sigma,mu,m,n,par);

if nargin<2,
    error('Requires at least two input arguments.');
end
if nargin<3, sigma = 1; end
if nargin<4, mu = 0; end
if nargin<5, m = 1; end
if nargin<6, n = 1; end
if nargin<7, par = 0; end

% Initialize r to zero.
r = zeros(m,n);

% Run the Chambers-Mallows-Stuck algorithm
U = pi.*(rand(size(r)) - 0.5);
W = -log(rand(size(r)));
piby2 = pi/2;
if alpha~=1,
    zeta = -beta.*tan(piby2.*alpha);
    xi = atan(-zeta)./alpha;
    S_ab = (1+zeta.^2).^(0.5./alpha);
    r = S_ab.*sin(alpha.*(U+xi))./(cos(U).^(1./alpha)).*(cos(U-alpha.*(U+xi))./W).^((1-alpha)./alpha);
else % alpha==0
    r = ((piby2 + beta.*U).*tan(U) - beta.*log(piby2*W.*cos(U)./(piby2 + beta.*U)))./piby2;
end

% Add scale and location
r = sigma.*r + mu;
if par==0, % Correct for alpha<>1 in the S0 parametrization
    if alpha~=1,
        r = r - sigma*beta*tan(alpha*piby2);
    end
else % Correct for alpha==1 in the S (or S1) parametrization
    if alpha==1,
        r = r + sigma*beta*log(sigma)/piby2;
    end
end
```

Code-Section Appendix A.2: stablernd.m Copyright (c) 2010 by Rafal Weron. The function generates random variables according to the input parameters of a stable distribution

```
function [ES, VaR] = asymstableES(xi , a, b, d, c , method)
    if nargin < 3, b=0; end, if nargin < 4, d=0; end
    if nargin < 5, c = 1; end, if nargin < 6, method = 1; end

    % Get q, the quantile from the S(0 ,1) distribution
    opt=optimset('Display', 'off', 'TolX', 1e-6);    q=fzero(@stabcdfroot , -6, opt , xi , a, b);
    VaR=d+c*q;

    if (q==0)
        t0 = (1 / a) * atan(b * tan(pi * a/2));
        ES = ((2 * gamma((a-1) / a)) / (pi - 2*t0)) * (cos(t0) / cos(a*t0)^(1 / a));
        return ;
    end
    ES=(c*Stoy(q,a,b)/xi)+d;
end

function diff = stabcdfroot(x, xi, a, b)
    [~, F] = asymstab_ab(x, a, b);
    diff =F-xi;
end

function [ f ,F] = asymstab_ab( xvec , a , b )
bordertol=1e-8; lo=bordertol ; hi=1-bordertol ; tol =1e-6;
xl=length(xvec ); F=zeros(xl ,1 ); f=F;
    for loop=1:length(xvec)
        x=xvec(loop);
        f(loop)= quadl(@fff_ab,lo,hi,tol ,[], x,a,b,1) / pi;
        if nargout>1
            F(loop)=0.5-(1/pi)* quadl(@fff_ab,lo,hi,tol ,[], x,a,b,0);
        end
    end
end

function S=Stoy(cut,a,b)
if nargin<3, b=0;end

cut=-cut; % we use a different sign convention
bbar=-sign(cut)*b;
t0bar=(1/a)*atan(bbar*tan(pi*a/2));

%'beta==0'=>'bbar==0'=>'t0bar==0'
small=1e-8;tol=1e-8;abscut=abs(cut);display=0;
integ=quadl(@stoyint,-t0bar+small,pi/2-small,tol,display,abscut,a,t0bar);

S=a/(a-1)/pi *abscut*integ;

function I=stoyint(t,cut,a,t0bar)
s=t0bar+t;
g=sin(a*s-2*t)./sin(a*s)-a*cos(t).^2./sin(a*s).^2;
v=(cos(a*t0bar)).^(1/(a-1)).*(cos(t)./sin(a*s)).^(a/(a-1)).*cos(a*s-t)./cos(t);
term=-(abs(cut)^(a/(a-1)));
I=g.*exp(term.*v);
```

Code-Section Appendix A.3: Functions *asymstableES*,*stabcdfroot*, *asymstab-ab* and *Stoy*, all used for Code-Section 4.

# References

[1] Marc S. Paolella (2018). Fundamental Statistical Inference: A Computational Approach, *New York: John Wiley Sons*.

[2] Marc S. Paolella (2007). Intermediate Probability: A Computational Approach, *New York: John Wiley Sons*.