University of Zurich[UZH]

Faculty of Business, Economics and Informatics

Statistical Foundations for Finance

Prof. Dr. Marco Paolella

Assignment 1 - 2

Authors:	Broennimann, Nicoletta	16-738-148
	Jezler, Linda	15-703-333
	Villiger, Cesare	17-920-208

In Partial Fulfillment of the Requirements for the Course "Statistical Foundations for Finance (Mathematical and Computational Statistics with a View Towards Finance)".

Date of Submission

October 18, 2022

Contents

1 Density of Stable Paretian Distribution

In question 1 we simulate random variates of a symmetric stable Paretian distribution. We then use a kernel estimate for the density and compare the result to the theoretical probability density function (pdf). The following is our code[1] to create random variates for an α-stable distribution and to compare the kernel density estimate to the theoretical density. The result can be seen in figure 1.

1.1 Code

```
%% ====================Exercise 1================================
% For stable parameter values alpha=1.7, beta=-0.4, scale 2 and
% location 0.3, generate a plot of the density that
% overlays TWO lines. The first line is a kernel density estimate
% based on 1e6 (one million) simulated IID stable variates with
% obviously the above mentioned parameters
%% ============================================================
%Creates random values based for a stable distribution with given
% parameters
clear
rng('default')

% Definition of parameters
alpha = 1.7; beta = -0.4; gamma =2; mu = 0.3;
x_lab = -30:.01:30;
nmbr_replications = 10^6;

% Creates stable random variates for given parameters
X = stblrnd(alpha, beta, gamma, mu, nmbr_replications, 1);

% Calculates the kernel density given the the random values from
% the above simulation
[f,xi] = ksdensity(X, x_lab);

% Calculates the theoretical density given the parameters
stab_theo = stblpdf(x_lab, alpha, beta, gamma, mu,'quick')


% Creates a figure given the two plots for the theoretical and
% the kernel density
```

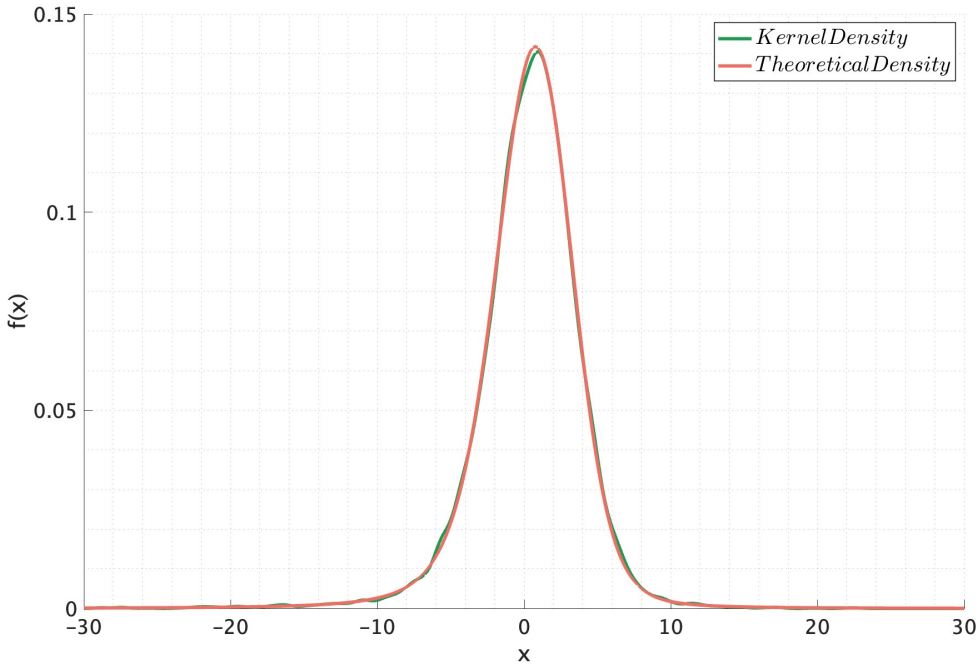
[1] Further documentation of the 'stblernd', 'stblepdf' and 'stblfit' can be found in the documentation for the 'STABLE' built-in package: https://www-mathworks-com.translate.goog/matlabcentral/fileexchange/37514-stbl-alpha-stable-distributions-for-matlab?_x_tr_sl=en_x_tr_tl=de_x_tr_hl=de_x_tr_pto=sc (09.10.2022).

```
PS = PLOT_STANDARDS();
figure(1);
fig1_comps.fig = gcf;
hold on
fig1_comps.p1 = plot(xi,f,'LineWidth',4);
fig1_comps.p2 = plot(x_lab, stab_theo,'LineWidth', 2);
grid minor % Create semitransparent area plots
set(fig1_comps.p1, 'Color', PS.Green3, 'LineWidth', 3);
set(fig1_comps.p2, 'Color', PS.Red1, 'LineWidth', 3);
%fig1_comps.plotTitle = title('Kernel Density vs Theoretical ...
    Density');
fig1_comps.plotXLabel = xlabel('x');
fig1_comps.plotYLabel = ylabel('f(x)');
fig1_comps.plotLegend = legend([fig1_comps.p1, fig1_comps.p2], ...
    '$$Kernel  Density$$', '$$Theoretical Density$$', ...
     '$$\frac{x}{\pi}$$', '\Big($$\frac{x}{\pi}\Big)^2$$', ...
     '$$exp\Big({\frac{x}{\pi}}\Big)$$', '$$log(1+x)$$', ...
     'Area\Big($$\frac{x}{\pi}\Big)^2$$', 'Interpreter', 'latex');
legendX0 = .71; legendY0 = .08; legendWidth = .3; legendHeight = .3;
set(fig1_comps.plotLegend, 'Location', PS.DefaultLegendLocation, ...
    'Box', 'on');
set(fig1_comps.plotLegend, 'FontSize', PS.LegendFontSize, ...
    'LineWidth', 0.75, ...
     'EdgeColor', PS.MyBlack);
set(gca, 'FontName', PS.PlotTextFont);
set([fig1_comps.plotXLabel, fig1_comps.plotYLabel], 'FontName', ...
    PS.PlotTextFont);
set(gca, 'FontSize', PS.AxisNumbersFontSize);
%set(fig1_comps.plotTitle, 'FontSize', PS.TitleFontSize, ...
    'FontWeight' , 'bold');
```

1.2 Output

The two densities are plotted for a visual comparison in figure 1 for 10^6 simulated random variates. Figure 1 shows the kernel as well as the theoretical density and as was to be expected, the kernel estimate of the pdf and the theoretical pdf are very similar, confirming the validity of the kernel estimate.

Figure 1: Kernel Density vs. Theoretical Density.



Note: The figure shows the kernel density and the theoretical pdf with the parameters α=1.7, β=-0.4, γ=2, μ=0.3. The two densities look very much alike, as was to be expected.

2 Density of the Sum of Stable Paretian Distributions with Same Tail Indices α

In order to confirm that stable distributions are closed under addition (as long as the α-parameters are the same), we note that the sum of two random variables is the product of their characteristic functions (c.f.s). If stable distributions are indeed closed under addition, the tail index α in the resulting c.f. should be the same as in the two original stable distributions. This means that the sum of two stable distributions with the same α-parameter is again a stable distribution with said α. The other parameters will not remain the same and the formulae for the calculation are presented below. This is the c.f. for a stable distribution[2]:

$$\varphi_X(t) = \mathbb{E}(e^{it(X)}) = exp(it\mu - \gamma^\alpha |t|^a [1 + i\beta \text{sgn}(t)\upsilon_\alpha(t)]), \quad t \in \mathbb{R}$$

$$\upsilon_\alpha(t) = \begin{cases} \tan(\frac{\pi\alpha}{2}), & \alpha \neq 1 \\ \frac{2}{\pi}, & \alpha = 1 \end{cases},$$

where α is the tail index, β the skewness, μ the location and γ the scale parameter. If we have two random variables (r.v.) X and Y, then the parameters of the sum $S = X + Y$ can be computed via the product of the c.f.[3]:

$$\varphi_{X+Y}(t) = \mathbb{E}(e^{it(X+Y)}) = \varphi_X(t)\varphi_Y(t).$$

From this, it can be shown that S is distributed with tail index α, scale parameter $\gamma = (\gamma_1^\alpha + ... + \gamma_n^\alpha)^{\frac{1}{\alpha}}$ and location parameter $\mu = \sum_{i=1}^n \mu$. The skewness parameter β is given by:

$$\beta = \frac{\beta_1\gamma_1^\alpha + ... + \beta_n\gamma_n^\alpha}{\gamma_1^\alpha + ... + \gamma_n^\alpha}.$$

Using these results, we can simulate two stable distributions with different parameters (except for the α-parameter) and sum up the random variables. Then we use the kernel estimate for the density and compare it to the theoretical pdf with parameters calculated according to the formulae above. The result can be seen in figure 2.

[2] Derivation of c.f. and the parameters by Nolan can be found here: http://prac.im.pwr.wroc.pl/ burnecki/chap1.pdf (09.10.2022)

[3] The formulae are generalized for the sum of n random variables. In our case, $n = 2$.

2.1 Code

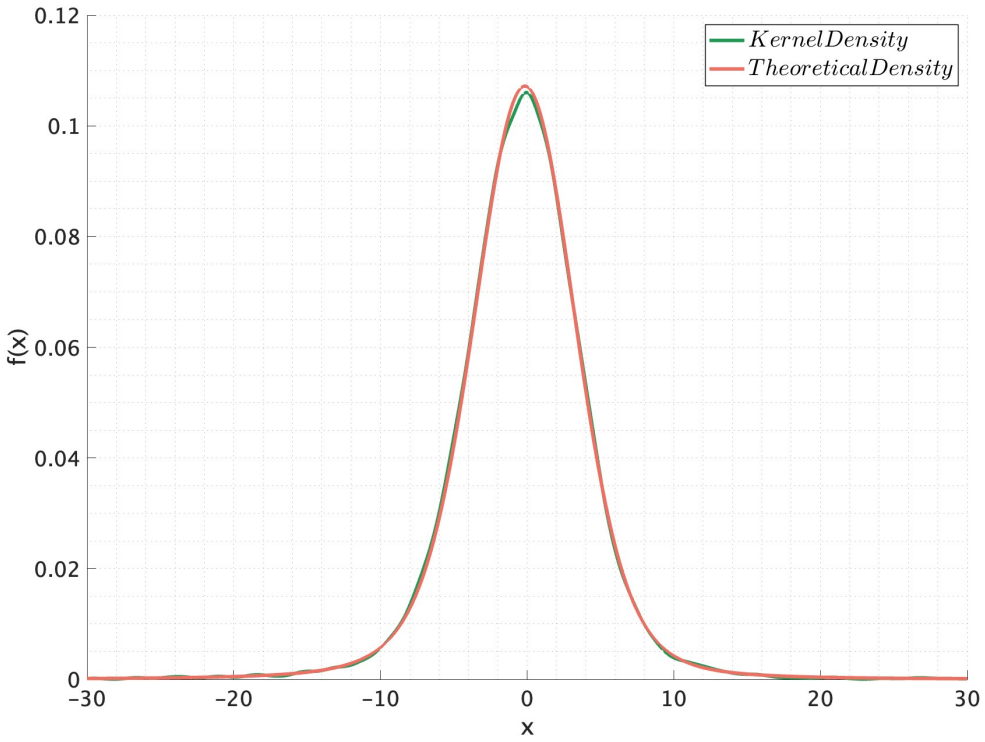
```
%% ====================Exercise 2==============================
% Confirm the convolution of two independent stable random ...
    variables with the same alpha* but possibly %different beta, ...
    location and scale, is itself again stable. Graph with the ...
    purported *theoretical* pdf of %S=X1+X2, which is stable ...
    with the parameters given as discussed in the textbooks. ...
    Take RV X1 to be stable %with alpha=1.7, beta=-0.4, scale=2, ...
    and location = -0.5; and independently, X2 is stable with ...
    the same %alpha, but with different values of beta, location ...
    and scale, that I let you choose. Compute the density %of a ...
    stable, but with the appropriate parameters based onthe ...
    convolution.
%% ============================================================
% Create random values for two stable distribution with different
% parameters (exept for alpaha) and the calculate the sum of the
% two for the convolution
clear
rng('default')

% Definitions of parameters
alpha=1.7;
beta1=-0.4; gam1=2; mu1=-0.5;
beta2=0.4; gam2=1.5; mu2=0.2;
x_lab = -30:.01:30;
nmbr_replications = 10^6;

% Create stable random variates for two different parameter vectors
X_1 = stblrnd(alpha, beta_1, gam_1, mu_1, nmbr_replications, 1);
X_2 = stblrnd(alpha, beta_2, gam_2, mu_2, nmbr_replications, 1);
S = X_1+X_2;

% Calculate the kernel density for the convolution
[f,x_lab] = ksdensity(S, x_lab);

% Parameters of the sum
a = 1.7; b = (beta_1*(gam_1^alpha)+beta_2*(gam_2^alpha))/ ...
    (gam_1^alpha+gam_2^alpha); gam = ...
    (gam_1^alpha+gam_2^alpha)^(1/alpha); mu = (mu_1+mu_2);

% Calculate the theoretical density given the calculated summed
% parameters
stab_theo_sum = stblpdf(x_lab, a, b, gam, mu, 'quick');

```

```
% Creates a figure given the two plots for the theoretical and
% the kernel density
PS = PLOT_STANDARDS();
figure(2);
fig2_comps.fig = gcf;
hold on
fig2_comps.p1 = plot(x_lab,f,'LineWidth',2)
fig2_comps.p2 = plot(x_lab, stab_theo_sum,'LineWidth',2);
grid minor % Create semitransparent area plots
set(fig2_comps.p1, 'Color', PS.Green3, 'LineWidth', 3);
set(fig2_comps.p2, 'Color', PS.Red1, 'LineWidth', 3);
%fig2_comps.plotTitle = title('Kernel Density vs Theoretical ...
    Density');
fig2_comps.plotXLabel = xlabel('x');
fig2_comps.plotYLabel = ylabel('f(x)');
fig2_comps.plotLegend = legend([fig2_comps.p1, fig2_comps.p2], ...
    '$$KernelDensity$$', '$$Density of S = X_1+X_2$$', ...
     '$$\frac{x}{\pi}$$', '\Big($$\frac{x}{\pi}\Big)^2$$', ...
     '$$exp\Big({\frac{x}{\pi}}\Big)$$', '$$log(1+x)$$', ...
     'Area\Big($$\frac{x}{\pi}\Big)^2$$', 'Interpreter', 'latex');
legendX0 = .71; legendY0 = .08; legendWidth = .3; legendHeight = .3;
set(fig2_comps.plotLegend, 'Location', PS.DefaultLegendLocation, ...
    'Box', 'on');
set(fig2_comps.plotLegend, 'FontSize', PS.LegendFontSize, ...
    'LineWidth', 0.75, ...
     'EdgeColor', PS.MyBlack);
set(gca, 'FontName', PS.PlotTextFont);
set([fig2_comps.plotXLabel, fig2_comps.plotYLabel], 'FontName', ...
    PS.PlotTextFont);
set(gca, 'FontSize', PS.AxisNumbersFontSize);
%set(fig2_comps.plotTitle, 'FontSize', PS.TitleFontSize, ...
    'FontWeight' , 'bold');
```

2.2 Output

Figure 2 shows the kernel estimate for the density and the theoretical pdf for 10^6 simulations. The lines, again, look very similar, confirming the correctness of the calculation of the parameters. The values for the parameters of the sum S can be seen in the note to figure 2.

Figure 2: Kernel Density vs. Theoretical Density.



Note: The figure shows the kernel density and the theoretical density of the sum of two stable random variables with resulting parameters: α=1.7, β=-0.096, γ=2.65, μ=-0.3. The two densities look very much alike, as was to be expected.

3 Density of the Sum of Stable Paretian Distributions with Different Tail Indices α

In question 3 we convolute two stable random variables with different values of tail index α in four different ways. First we use the simple integration formula (we will refer to this as convolution formula) to get the pdf of the sum of the two stable random variables with different values of tail index α:

$$f_s(s) = \int_{-\infty}^{\infty} f_{X,Y}(x, s-x)\ dx,$$

where $f_{X,Y}$ is the density of the jointly distributed continuous random variables.
We then compare the resulting pdf with the pdf calculated via the inversion formula, applied to the characteristic function of the sum of X_1 and X_2. Since the characteristic function of the linear combination is continuous, the pdf of the two characteristic functions is calculated by the following formula:

$$f_X(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-itx} \varphi_{X+Y}(t)\ dt.$$

Thirdly, we compare the two pdfs generated by the convolution formula and the inversion formula with the kernel density, estimated from the sum of 10^6 simulations of random variates for two sets of parameters.
In a last step, we use the *conv*()-function that is built into Matlab. We use the bin size as a factor (the "magic number"), since the convolution function does not consider the bin size in the multiplication of the elements in the vectors that are convoluted. The results can be seen in figures 3 and 4 for the respective α-parameters. The code is set up such that all variables can be defined at the beginning. This allows us to easily change parameters or other factors, such as bin size or the number of replications. The presented code uses $\alpha_1 = 1.6$ and $\alpha_2 = 1.8$. For the second step, i.e. different α-parameters, we simply change these values to $\alpha_1 = 1.5$ and $\alpha_2 = 1.9$.

3.1 Code

```
%======================Exercise 3=================================
% Convolute the densities of two stable parameters with ...
    different values of tail index alpha (alpha1=1.6, ...
    alpha2=1.8), beta=0, scale 1 and location 0, in four ...
    different ways. First based on simulation, second with the ...
    inversion formula,third with the convolution formula and ...
    last with the conv() function. Repeat the above, but for ...
    alpha1=1.5, and alpha2=1.9.
%
%=========================================================================
% Create random values for two stable distribution with different
% parameters (exept for alpaha) and the calculate the sum of the ...
    two for
% the convolution
clear
rng('default')
bin_size = .2;
x_min = -10; x_max = 10;
nmbr_replications = 10^6;

% Define parameters: alpha_1 and alpha_2 need to be changed for ...
    second run
alpha_1 = 1.6; alpha_2 = 1.8; beta = 0; gam = 1; mu = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%a) kernel estimate

%simulate stable random variates for two different alphas
X_1 = stblrnd(alpha_1, beta, gam, mu, nmbr_replications, 1);
X_2 = stblrnd(alpha_2, beta, gam, mu, nmbr_replications, 1);

% Sum the two simulations of stable random variates
S = X_1 + X_2;

% Calculate the kernel density for the convolution
x1 = x_min:bin_size:x_max;
[y1, x1] = ksdensity(S, x1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%b) inversion formula

% Create RV S = X_1 + X_2 from two alpha-stable distributions ...
    with two
```

```
% different alpha parameters. We do that by integrating over the ...
    product of
% the two characteristic function. First we create the c.f.:

cf_stable_1  = @(t) exp(sqrt(-1)*mu*t - ...
    gam*abs(t).^alpha_1.*(1+sqrt(-1)*beta*(t/abs(t))*-(2/pi)* ...
    log(abs(t))));
cf_stable_2  = @(t) exp(sqrt(-1)*mu*t - ...
    gam*abs(t).^alpha_2.*(1+sqrt(-1)*beta*(t/abs(t))*-(2/pi)* ...
    log(abs(t))));


% C.f. of the sum of two random variables is the product of the two
% c.f.:
c    = [1 1];
cf_stable_Z = @(t) cf_stable_1(c(1)*t) .* cf_stable_2(c(2)*t);


% Calculate inversion formula (integral) of product of c.f.
fx =  @(x) real(1/(2.*pi) .* integral(@(t)exp(-i.*t.*x).* ...
    cf_stable_Z(t),-inf,inf));

% Loop of the x-values to get pdf
x2=x_min:bin_size:x_max;
for i=1:size(x2,2)
y2(i)=fx(x2(i));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%c) convolution formula

%% Create two stable random variables
fx= @(x) stblpdf(x, alpha_1, beta, gam, mu,'quick');
fy= @(y) stblpdf(y, alpha_2, beta, gam, mu,'quick');

% Calculate convolution via formula (integral)
fs = @(s) (integral(@(x) fx(x).*fy(s-x),-inf,inf));

% Loop over the x-values to get the pdf
x3=x_min:bin_size:x_max;
for j=1:size(x3,2)
y3(j)=fs(x3(j));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%d) conv() function in matlab

```

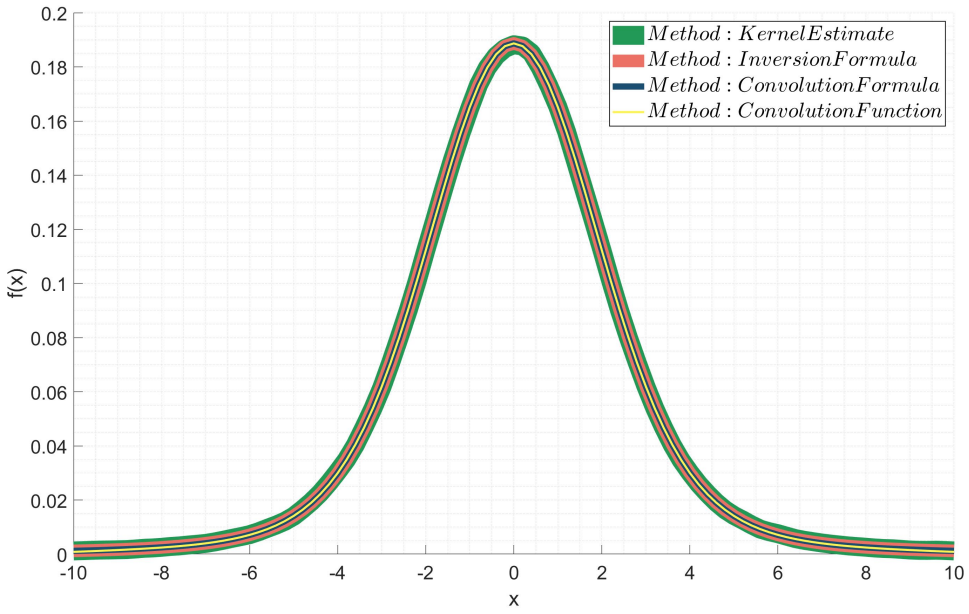
```
% Via conv-function, as per extra last mail
% create linspace for x-axis in plot
x_4 = x_min:bin_size:x_max;

% Create two stable densities
X_1_pdf = stblpdf(x_4, alpha_1, beta, gam, mu, 'quick');
X_2_pdf = stblpdf(x_4, alpha_2, beta, gam, mu, 'quick');

% Convolute the stable random variates and multiply convolution ... by the
% bin size, as conv() does not consider bin size in the ... multiplication
%of the elements in the vector.
conv = conv(X_1_pdf, X_2_pdf, 'same')*bin_size;

% Create x-axis for plot
x_lab_conv = linspace(x_min, x_max, length(conv));

%figure
PS = PLOT_STANDARDS();
figure(3);
fig3_comps.fig = gcf;
hold on
fig3_comps.p1 = plot(x1, y1, 'LineWidth',2) %kernel
fig3_comps.p2 = plot(x2, y2,'LineWidth',2) %inversion
fig3_comps.p3 = plot(x3, y3,'LineWidth',2) %convolution
fig3_comps.p4 = plot(x_lab_conv, conv,'LineWidth',2) %conv()
grid minor % Create semitransparent area plots
set(fig3_comps.p1, 'Color', PS.Green3, 'LineWidth', 18);
set(fig3_comps.p2, 'Color', PS.Red1, 'LineWidth', 12);
set(fig3_comps.p3, 'Color', PS.Blue5, 'LineWidth', 6);
set(fig3_comps.p4, 'Color', PS.Yellow1, 'LineWidth', 2);
fig3_comps.plotXLabel = xlabel('x');
fig3_comps.plotYLabel = ylabel('f(x)');
fig3_comps.plotLegend = legend([fig3_comps.p1, fig3_comps.p2, ... fig3_comps.p3, fig3_comps.p4], '$$Method Kernel Estimate$$', ... '$$Method: Inversion Formula$$','$$Method: Convolution ... Formula$$', '$$Method: Convolution Function$$',  ...
     '$$\frac{x}{\pi}$$', '\Big($$\frac{x}{\pi}\Big)^2$$', ...
     '$$exp\Big({\frac{x}{\pi}}\Big)$$', '$$log(1+x)$$', ...
     'Area\Big($$\frac{x}{\pi}\Big)^2$$', 'Interpreter', 'latex');
legendX0 = .71; legendY0 = .08; legendWidth = .3; legendHeight = .3;
set(fig3_comps.plotLegend, 'Location', PS.DefaultLegendLocation, ... 'Box', 'on');
set(fig3_comps.plotLegend, 'FontSize', PS.LegendFontSize, ... 'LineWidth', 0.75, ...
     'EdgeColor', PS.MyBlack);
```

```
116 set(gca, 'FontName', PS.PlotTextFont);
117 %set([fig3_comps.plotXLabel, fig3_comps.plotYLabel], 'FontName', ...
        PS.PlotTextFont);
118 set(gca, 'FontSize', PS.AxisNumbersFontSize);
```

3.2 Output

Figure 3 and 4 plot the lines from the explanations above for two different sets of parameters for 10^6 simulation per set of parameters. As can be seen in figure 3, all four lines are nearly identical, as was to be expected. Also, the two graphs look very similar. The differences are minimal and are only really visible, if you switch between the two graphs in Matlab. This is because the parameters β, γ and μ are the same and the convolution of the two random variables gives very similar estimates for α, as can be seen in table 2 and 3.[4]

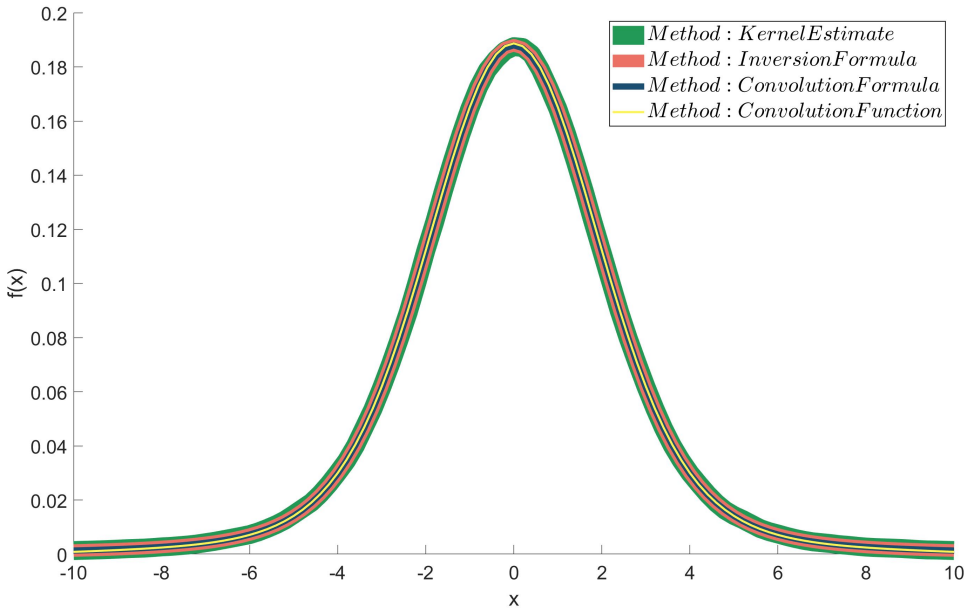
Figure 3: Kernel Estimate vs. Inversion vs. Convolution Formulae I.



Note: The figure shows the kernel density, the theoretical pdf and the pdf through convolution (calculated via the integral and Matlab function) with the parameters α_1=1.6, α_2=1.8, β=0, γ=1, $\mu = 0$. The four densities look very much alike, as was to be expected.

[4]Since all four lines in Figure 3 and 4 are very similar, we use four different line widths, to see them all.

Figure 4: Kernel Estimate vs. Inversion vs. Convolution Formulae II.



Note: The figure shows the kernel density, the theoretical pdf and the pdf through convolution (calculated via the integral and Matlab function) with the parameters α_1=1.5, α_2=1.9, β=0, γ=1, μ=0. The four densities look very much alike, as was to be expected.

3.3 (Bonus) Sums of Three Random Variables using the *conv*() Function

We can compute the theoretical pdf of the sum of three stable r.v.s with the same tail index α analogously as we did in section 2 (obviously, now $n = 3$). We could then compare a stable distribution with the calculated parameters to the pdf calculated via the *conv*() function.

To avoid being repetitive, we look at the sum of three Gaussian r.v.s by comparing their theoretical pdf to the pdf calculated by the *conv*() function. We simulate three Gaussian r.v.s and use the *conv*() function to get the pdf of the sum. This, we can compare to the theoretical pdf of the sum of three Gaussian r.v.s, as the parameters of the convolution can readily be calculated via the product of the c.f. Given the independence of the r.v.s, the parameters are simply the sum of the original parameters, such that: $\mu_{sum} = \mu_1 + \mu_2 + \mu_3$ and equivalently for σ_{sum}. As can be seen in figure 5, the lines are nearly identical, confirming the validity of the calculations by the *conv*() function.

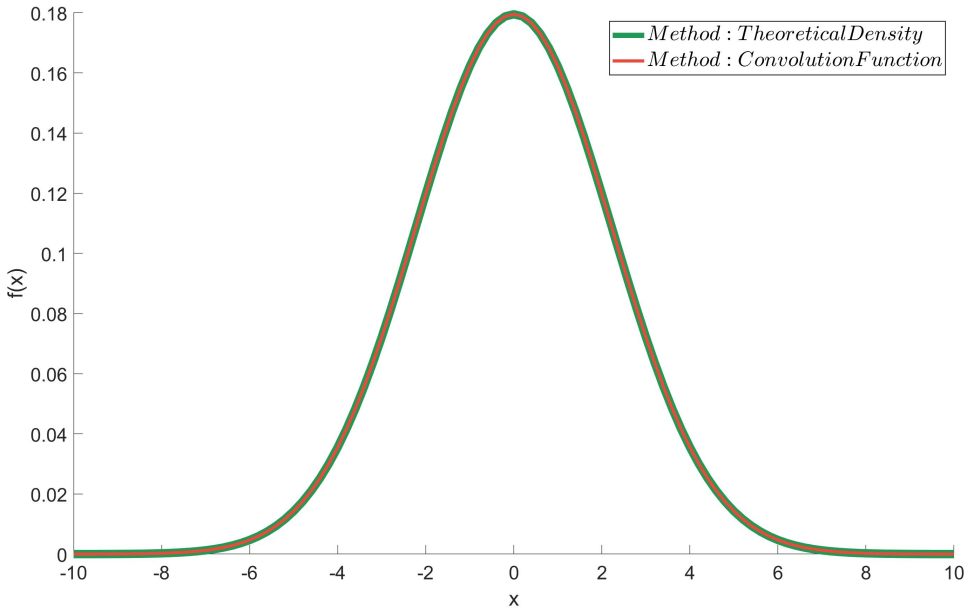
3.3.1 Code

```
%======================Exercise 3 (Bonus)==========================
%Convolute the densities of three normal/gaussian random ...
    variables with the conv() function. Compare it to the ...
    theoretical distribution of the sum of three gaussian random ...
    variables.
%======================================================================
%%create convolution of three normal/gaussian random variables ...
    (same bonus
%%question, but there is no theoretical distribution for three ...
    stbale
%%random variables to compare to. for the normal case, we can ...
    make a
%%convolution and compare it to the theoretical one. Please make ...
    graph that
%%shows theoretical vs. simulated sum of three rv.)
clear
rng('default')
bin_size = 0.2;
x_lab = -10:bin_size:10;
length_x_lab = length(x_lab);

% Create three gaussian random variables using built-in function ...
    (where
% sigma is the standard deviation)
% Creates pdfs for three Gaussian random variables with ...
    different paramters
mu_1 = 0; mu_2 = 0; mu_3 = 0; sigma_1 = 1; sigma_2 = 1.5; ...
    sigma_3 = 1.3;
X_1_norm = normpdf(x_lab, mu_1, sigma_1);
X_2_norm = normpdf(x_lab, mu_2, sigma_2);
X_3_norm = normpdf(x_lab, mu_3, sigma_3);

% Create sum of all random variates
Z = X_1_norm + X_2_norm + X_3_norm;

% Convolution of three gaussian rv.
conv_norm_3 = conv(conv(X_1_norm, X_2_norm, 'same'), X_3_norm, ...
    'same')*bin_size^2;

% for x-values
x_lab_conf = linspace(-10,10, length(conv_norm_3));

%theoretical pdf for convolution of three gaussian rv. if the three
```

```
%gaussian random variables are independent, the sum has is also ...
    normally
%distributed with mu and sigma as the sum of the single parameters.

% Calculate parameters mu and sigma for convoluted gaussian
mu_sum = mu_1 + mu_2 + mu_3;
sigma_sum = sqrt(sigma_1^2 + sigma_2^2 + sigma_3^2);

% Normal density with theoretical parmater values for ...
    convolution of three Gaussian random variables
X_sum = normpdf(x_lab, mu_sum, sigma_sum);


% Plot theoretical vs. convolution line in the same plot
PS = PLOT_STANDARDS();
figure(2);
fig3_comps.fig = gcf;
hold on
fig3_comps.p1 = plot(x_lab, X_sum, 'LineWidth',2) %theoretical
fig3_comps.p2 = plot(x_lab_conf, conv_norm_3,'LineWidth',2) ...
    %convolution
grid minor % Create semitransparent area plots
set(fig3_comps.p1, 'Color', PS.Green3, 'LineWidth', 5);
set(fig3_comps.p2, 'Color', PS.Red2, 'LineWidth', 3);
fig3_comps.plotXLabel = xlabel('x');
fig3_comps.plotYLabel = ylabel('f(x)');
fig3_comps.plotLegend = legend([fig3_comps.p1, fig3_comps.p2], ...
    '$$Method: Theoretical Density$$', '$$Method: Convolution ...
    Function$$',  ...
     '$$\frac{x}{\pi}$$', '\Big($$\frac{x}{\pi}\Big)^2$$', ...
     '$$exp\Big({\frac{x}{\pi}}\Big)$$', '$$log(1+x)$$', ...
     'Area\Big($$\frac{x}{\pi}\Big)^2$$', 'Interpreter', 'latex');
legendX0 = .71; legendY0 = .08; legendWidth = .3; legendHeight = .3;
set(fig3_comps.plotLegend, 'Location', PS.DefaultLegendLocation, ...
    'Box', 'on');
set(fig3_comps.plotLegend, 'FontSize', PS.LegendFontSize, ...
    'LineWidth', 0.75, ...
     'EdgeColor', PS.MyBlack);
set(gca, 'FontName', PS.PlotTextFont);
%set([fig3_comps.plotXLabel, fig3_comps.plotYLabel], 'FontName', ...
    PS.PlotTextFont);
set(gca, 'FontSize', PS.AxisNumbersFontSize);
```

3.3.2 Output

Figure 5: Theoretical vs. Convolution Density of Sum of Three Gaussian Random Variables.



Note: The figure shows the theoretical pdf and the pdf generated by the convolution of three Gaussian random variables, with $\mu_1 = 0$, $\mu_2 = 0$, $\mu_3 = 0$, $\sigma_1 = 1$, $\sigma_2 = 1.5$, $\sigma_3 = 1.3$.

4 Theoretical vs. Empirical Expected Shortfall of a Stable Paretian Distribution

In question 4 we turn to the expected shortfall. The expected shortfall (ES) is an example of tail risk measures used in empirical finance and quantitative (financial) risk management.[5] The expected shortfall is defined as follows:

$$ES_\theta(R) = \mathbb{E}[R|R \leq q_{R,\theta}] = \frac{1}{\theta}\int_{-\infty}^{q_{R,\theta}} r f_R(r)\, dr,$$

where R is a future period financial return and $q_{R,}$ is the θ-quantile such that $\mathbb{P}(R \leq q_{R,\theta}) = \theta$, where θ[6] is the tail probability.[7]
To compute the theoretical ES for the stable parameters (α=1.7, β=0, γ=1, μ=0) and θ=0.01, we use the results from Stoyanov et al.[8], which can be seen in our code. We then compare the theoretical ES with the empirical ES by simulating 10^6 stable variates. As can be seen in the table 1, they are almost the same up to some small simulation error.

Table 1: Theoretical vs. Empirical ES.

Theoretical ES	-11.5516
Empirical ES	-11.2422

[5] (Paolella, M. S. (2018). Fundamental Statistical Inference: A Computational Approach (Vol. 216). John Wiley Sons. Page 437)
[6] In the assignment x_i corresponds to θ
[7] (Paolella, M.S. (2022). Intermediate Probability: A Computational Approach [lecture notes]. Slide 459)
[8] (Stoyanov, S. V., Samorodnitsky, G., Rachev, S., Ortobelli Lozza, S. (2006). Computing the portfolio conditional value-at-risk in the alpha-stable case. Probability and Mathematical Statistics, 26(1), 1-22)

4.1 Code

```
%=========================Exercise 4==============================
%Compute the theoretical and the empirical ES for given stable ...
    parameters.
%=================================================================
% Define variables
alpha=1.7; beta=0; gam=1; mu=0;
P=stblrnd(alpha,beta,gam,mu,10^6,1);
xi=0.01; %xi is the equivalent to theta in the text -> quantile

% Theoretical ES
q=quantile(P, xi);
ES_t=(gam*Stoy(q,alpha,beta)/xi)+mu

% Empirical ES
q=quantile(P, xi); Z=stblrnd(alpha,beta,gam,mu,10^6,1); I=(Z<q);
ES_e=mean(Z.*I)/xi
```

5 Expected Shortfall of Sums of Stable Paretian Distributions

In question 5 we simulate 10^6 random variates for two α-stable r.v.s with different α-parameters. We sum them up in $S = X_1 + X_2$ and then estimate the parameters of the resulting distribution. Unfortunately, the link that was provided for the black-box program did not work for us, so we resorted to the built-in 'stblfit' function (see footnote 1) for the parameter estimation. The resulting parameters are:

Table 2: Parameter Estimates for Sum of Stable Random Variable with $\alpha_1 = 1.6$ and $\alpha_2 = 1.8$.

α	β	γ	μ
1.6882	-0.0013	1.5030	2.8656e-04

Table 3: Parameter Estimates for Sum of Stable Random Variable with $\alpha_1 = 1.5$ and $\alpha_2 = 1.9$.

α	β	γ	μ
1.6516	-0.0012	1.5033	3.0386e-04

The parameter estimates in tables 2 and 3 satisfy common sense, as the estimates for the α-parameters lie between the two α-parameters used in the convolution. Also,

as we noticed already when looking at the output of section 3.2, the outputs of figures 3 and 4 look very similar, given the almost identical estimates for α and β.

Using the estimated parameters, we calculate the theoretical ES and compare it to the empirical ES in tables 4 and 5. In order to relate the theoretical to the empirical value, we apply a simple measure of accuracy: $\frac{TheoreticalES}{EmpiricalES} - 1$. Obviously, the smaller this measure, the closer are the two numbers to each other, and the more accurate is the empirical ES. Table 4 and 5 confirm the assumption made in the problem assignment that the accuracy increases with smaller θ. We also note that the accuracy in table 5 is lower than in table 4, because the α-values used in table 5 and farther apart from each other. We suspect that this results in a less accurate estimation of the convolution parameters. Followingly, the accuracy is lower.

Table 4: Theoretical vs. Empirical ES for $\alpha_1 = 1.6$ and $\alpha_2 = 1.8$.

θ	0.01	0.025	0.05
Theoretical ES	-13.9655	-7.8310	-5.1914
Empirical ES	-15.9858	-10.1918	-6.9851
Accuracy	-0.1264	-0.2316	-0.2568

Table 5: Theoretical vs. Empirical ES $\alpha_1 = 1.5$ and $\alpha_2 = 1.9$.

θ	0.01	0.025	0.05
Theoretical ES	-14.7123	-8.2809	-5.4265
Empirical ES	-21.9055	-12.6124	-8.6426
Accuracy	-0.3284	-0.3434	-0.3721

5.1 Code

```
%=========================Exercise 5==============================
% In exercise 5 we sum two stable random variables and calculate ...
    the empirical ES to the theoretical ES calucluated via the ...
    parameters obtained from fitting the model with 'stblfit'.
%=================================================================
% Create random values for two stable distribution with different
% parameters (exept for alpaha) and the calculate the sum of the ...
    two for the convolution
clear
rng('default')
nmbr_replications = 10^6;

```

```
% Define parameters: alpha_1 and alpha_2 need to be changed for ...
    second run
alpha_1=1.6; alpha_2=1.8; beta=0; gam=1; mu=0;

% Create stable random variates
X_1 = stblrnd(alpha_1,beta,gam,mu,nmbr_replications,1);
X_2 = stblrnd(alpha_2,beta,gam,mu,nmbr_replications,1);
S = X_1 + X_2;

% Estimate the parameters. The link in the assignment to the ...
    black-box function does not exist anymore, so we use ...
    'stblfit' to estimate the parameters.
p = stblfit(S,'ecf',statset('Display','iter'));

% Fitted parameters
alpha = p(1); beta = p(2); gam = p(3); mu = p(4);

% Define the three quantiles
xi1=0.01; xi2=0.025; xi3=0.05;
P=stblrnd(alpha, beta, gam, mu, 10^6, 1);

q1=quantile(P, xi1); q2=quantile(P, xi2); q3=quantile(P, xi3);

% Theoretical ES based on the estimates parameters of the ...
    convolution
ES_t1=(gam*Stoy(q1,alpha,beta)/xi1)+mu
ES_t2=(gam*Stoy(q2,alpha,beta)/xi2)+mu
ES_t3=(gam*Stoy(q3,alpha,beta)/xi3)+mu

% For the empirical ES we use the sum of the two stable rv (S).
I1=(S<q1); I2=(S<q2); I3=(S<q3);

ES_e1=mean(S.*I1)/xi1
ES_e2=mean(S.*I2)/xi2
ES_e3=mean(S.*I3)/xi3

% Simple measure of accuracy
acc_x1 = (ES_t1/ES_e1)-1
acc_x2 = (ES_t2/ES_e2)-1
acc_x3 = (ES_t3/ES_e3)-1
```