Computing call option values through the Normal Inverse Gauss model of Barndorff-Nielsen.

For the NIG model the probability density is known in terms of Bessel functions. Hence the option value is given through a Black-Scholes style formula by numerical integration. For that one needs a 'risk neutral' density and this adjustment for the rates is done here through the Esscher transform (since it is also known explicitely). Alternatively one could have choosen a mean correcting adjustment.

Computational time can be improved by using an external library for numerics (as Maple seems to care too much for symbolics for special functions which is not need here).

References: Wim Schoutens, Levy Processes in Finance, Wiley 2003
and http://www.wilmott.co.uk/messageview.cfm?catid=8&threadid=14313

```
> restart;
  Digits:=10; # give Maple the chance to use hardware

  # parameter restrictions are
  # assume(mu::real):
  # assume(0<delta):
  # assume(0<=alpha):
  # assume(beta::real); additionally(abs(beta)<alpha);
```

$$\text{Digits} := 10$$

Set up the pdf for the NIG model

```
> alpha*delta/Pi*exp(delta*(alpha^2-beta^2)^(1/2)+beta*(x-mu))*
    BesselK(1,alpha*(delta^2+(x-mu)^2)^(1/2))/(delta^2+(x-mu)^2)^(1/2):
  subs(mu=0,%):
  pdfNIG:= unapply(%, x, alpha, beta, delta);
```

$$\text{pdfNIG} := (x, \alpha, \beta, \delta) \to \frac{\alpha\, \delta\, \mathbf{e}^{(\delta \sqrt{\alpha^2-\beta^2}+\beta\, x)}\, \text{BesselK}(1, \alpha \sqrt{\delta^2+x^2})}{\pi \sqrt{\delta^2+x^2}}$$

Now write down the Black-Scholes style pricing formula using numerical integration, restrict integration to a finite intervall by cutting off (see below for the integrands) and choose an adaptive Gaussian quadrature method within Maple:

```
> cutOff:=4.0;
  c:=ln(K/S0);
  I1:= (S0,K,t,  alpha,beta,delta) ->
    Int(pdfNIG(x,alpha, 1+beta,delta*t),x=ln(K/S0)..cutOff,method = _Gquad);
  I2:= (S0,K,t,  alpha,beta,delta) ->
    Int(pdfNIG(x,alpha,   beta,delta*t),x=ln(K/S0)..cutOff,method = _Gquad);
  ``;
  C:= (S0,K,t,r, alpha,beta,delta) ->
    S0*I1(S0,K,t, alpha,beta,delta) - exp(-r*t)*K*I2(S0,K,t,
  alpha,beta,delta);
```

$$\text{cutOff} := 4.0$$

$$c := \ln\left(\frac{K}{S0}\right)$$

$$I1 := (S0, K, t, \alpha, \beta, \delta) \to \text{Int}\left(\text{pdfNIG}(x, \alpha, 1+\beta, \delta\, t), x = \ln\left(\frac{K}{S0}\right) .. \text{cutOff}, \text{method} = \_Gquad\right)$$

$$I2 := (S0, K, t, \alpha, \beta, \delta) \to \text{Int}\left(\text{pdfNIG}(x, \alpha, \beta, \delta\, t), x = \ln\left(\frac{K}{S0}\right) .. \text{cutOff}, \text{method} = \_Gquad\right)$$

$$C := (S0, K, t, r, \alpha, \beta, \delta) \to S0\ I1(S0, K, t, \alpha, \beta, \delta) - e^{(-r\,t)}\ K\ I2(S0, K, t, \alpha, \beta, \delta)$$

Adjust the whole thing for rates:
The Esscher transform for the NIG model means to solve the following equation for theta
$r - \mu = \delta\,(\sqrt{\alpha^2 - (\beta + \theta)^2} - \sqrt{\alpha^2 - (\beta + \theta + 1)^2})$ and use a new beta as $\beta + \theta$

This is done numerical after choosing some test data (do not forget to compute a new beta if you change the data).

```
> tstData:=[alpha=6.1882, beta=-3.8941, delta=0.1622, mu=0, r=0.019,
  S0=1124.47]; ``;
```

$$tstData := [\alpha = 6.1882, \beta = -3.8941, \delta = 0.1622, \mu = 0, r = 0.019, S0 = 1124.47]$$

```
> r-mu =
  delta*((alpha^2-(beta+theta)^2)^(1/2)-(alpha^2-(beta+(theta+1))^2)^(1/2)):
  eval(%,tstData):
  theTheta:='fsolve(%,theta,complex)';

  'newBeta'='beta+theTheta';
  newBeta:= eval(beta+theTheta,tstData);
```

$$theTheta := fsolve(0.019 = 0.1622\sqrt{38.29381924 - (-3.8941 + \theta)^2} - 0.1622\sqrt{38.29381924 - (-2.8941 + \theta)^2}, \theta, complex)$$

$$newBeta = \beta + theTheta$$

$$newBeta := 0.217572637$$

Example: strike = 900 and expiry in 1.708 years:

```
> tstTime:=1.708;
  tstK:=900.0;
  st:=time():
  ``;
  C(1124.47,tstK,tstTime, 0.019, 6.1882,newBeta,0.1622):
  `price`=evalf(%);
  ``;
  `computational time`=time()-st;
```

$$tstTime := 1.708$$

$$tstK := 900.0$$

$$price = 264.4620921$$

$$computational\ time = 4.174$$

on my PC (WinME, 900 Mhz and 256 MB) it takes almost 2 seconds to get the result

Instead of purely working within Maple use the enclosed C library for the numerics,
a DLL called NIG_tiny.dll - it is assumed to be in the same directory as this sheet

```
> currentdir(): NIGDLL:=cat(%,`\\NIG_tiny.dll`);
  #NIGDLL:="C:/temp/NIG_tiny.dll";
```

NIGDLL := "C:_Work\Maple_Work\Finance\TransformMethods\NIG\NIG_tiny.dll"

access the pricing function which is coded in the DLL (restricting to 15 digits)

```
> evalF:=proc(expr)
```

```
    return ( evalf(expr,15) );
  end proc:

  NIG_Call := evalF@define_external( 'NIG_Call_BS_style',AUTO,
    'S'::float[8],
    'K'::float[8],
    't'::float[8],
    'r'::float[8],
    'a'::float[8],
    'b'::float[8],
    'd'::float[8],
    'mu'::float[8],
    'cutOff'::float[8],
    'RETURN'::float[8] ,
    LIB=NIGDLL):
```

compare the option value with the results given by Maple

```
> `price`=NIG_Call(1124.47, 900, 1.708, 0.019, 6.1882, 0.217572637, 0.1622, 0.0
  , 8.0);
```

price = 264.462091644507

compare the speed (400 strikes, from 800 to 1200)

```
> st:=time():
  seq(NIG_Call(1124.47, 800+i, 1.708, 0.019, 6.1882, 0.217572637, 0.1622, 0.0,
  8.0),
    i=1..400):
  (time()-st)/400:
  `computational time each option value [seconds] `=evalf(%,2);
```

computational time each option value [seconds] = 0.00062

So exactness is similar and speed improved by a factor of 1000 (on my PC it is 0.6 msec) compared to the direct method (may be because of the Bessel functions in Maple).

As integration was cut off it is a good thing to look at the integrands (ok, that's not a sound justification ...). That is also coded inside the DLL.

```
> theIntegrand0 := evalF@define_external( 'theIntegrand_BS_style',AUTO,
    'a'::float[8],
    'b'::float[8],
    'd'::float[8],
    'mu'::float[8],
    'x'::float[8],
    'RETURN'::float[8] ,
    LIB=NIGDLL):
```

A constant factor was split off before integration (i do not know whether Maple does that automatically, but i suppose it). Note that we have to use a modified beta and write down a handy version of the integrand:

```
> theFactor:=a*d/Pi*exp(d*sqrt(a*a-b*b)-b*mu);
  subs(a=6.1882, b=0.217572637, d=0.1622*1.708, mu=0.0, %):
  theFactor:=evalf(%);
  ``;
  theIntegrand:=proc(x::numeric)
    return( theFactor * theIntegrand0(6.1882, 0.217572637, 0.1622*1.708, 0.0,
  1.0*x) );
  end proc;
```

$$\text{theFactor} := \frac{a\, d\, \mathbf{e}^{\left(d \sqrt{a^2 - b^2} - b\, \mu\right)}}{\pi}$$

$$\text{theFactor} := 3.027134452$$

theIntegrand :=

proc(x::numeric) **return** theFactor∗theIntegrand0(6.1882, 0.217572637, 0.1622∗(1.708), 0., 1.0∗x) **end proc**

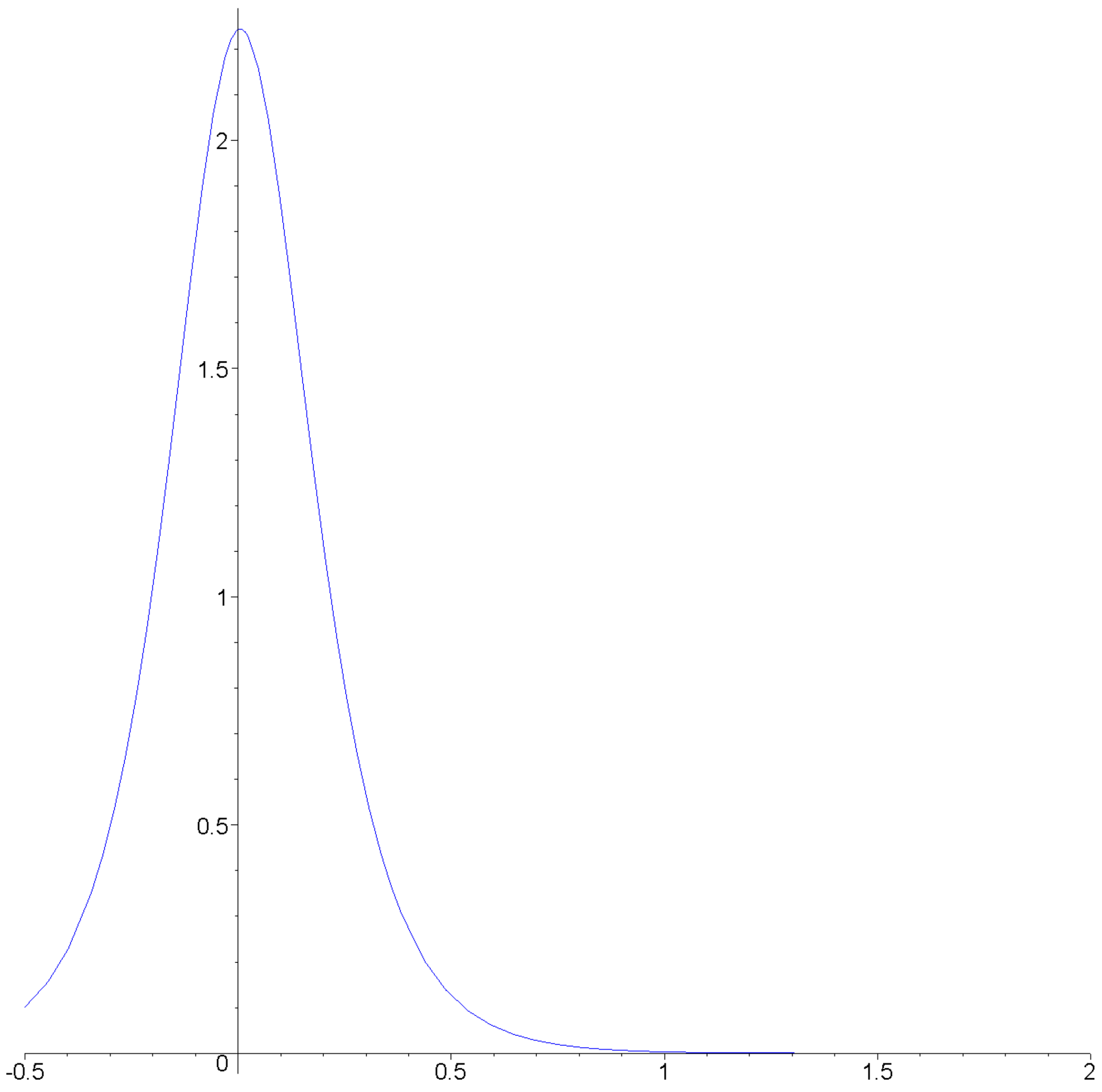
Look at large values for a cut off and plot the graph:

```
> 'theIntegrand(4.0)': '%'=%;
  'theIntegrand(6.0)': '%'=%;
  'theIntegrand(8.0)': '%'=%;
  plot(theIntegrand, -0.5..2.0, colour=blue); #plot(theIntegrand, 4.0..6.0,
  colour=orange);
  #`integrand2`=pdfNIG(x,6.1882,newBeta,0.1622*tstTime): evalf(rhs(%));
  #plot(%, x=ln(tstK/1124.47)..2);
```

$$\text{theIntegrand}(4.0) = 0.7714403096\ 10^{-11}$$

$$\text{theIntegrand}(6.0) = 0.2782724740\ 10^{-16}$$

$$\text{theIntegrand}(8.0) = 0.1187319008\ 10^{-21}$$



For a proper estimating to cut off integration one could take an asymptotic approximation of the integrand as in Schoutens book p. 61, which declines monotonously (remember abs(beta)<alpha):

```
> 'asympt( pdfNIG(x, alpha, beta, delta),x )' = x^(-3/2)*exp((-alpha+beta)*x);

  Int(x^(-3/2)*exp((-alpha+beta)*x), x=cutoff..infinity):
  %= value(%):
  subs(alpha-beta=tmp^2,%): simplify(%) assuming 0<tmp:
  collect(%,tmp): collect(%,cutoff^(1/2)): collect(%,Pi^(1/2)):
  subs(tmp=sqrt(alpha-beta),%);
```

$$\mathrm{asympt}(\mathrm{pdfNIG}(x, \alpha, \beta, \delta), x) = \frac{e^{((-\alpha+\beta)\,x)}}{x^{(3/2)}}$$

$$\int_{\mathrm{cutoff}}^{\infty} \frac{\mathbf{e}^{(-(\alpha-\beta)\,x)}}{x^{(3/2)}}\,dx = 2\,(-1+\mathrm{erf}(\sqrt{\alpha-\beta}\,\sqrt{\mathrm{cutoff}}))\,\sqrt{\alpha-\beta}\,\sqrt{\pi} + \frac{2\,\mathbf{e}^{(-(\alpha-\beta)\,\mathrm{cutoff})}}{\sqrt{\mathrm{cutoff}}}$$

Example: roughly estimate the tail to contribute errors less than 15 decimals:

```
> epsilon:=1E-16;
  ``;
  Int(x^(-3/2)*exp((-alpha+beta)*x), x=cutoff..infinity) <= 'epsilon';:
  int(x^(-3/2)*exp((-alpha+beta)*x), x=cutoff..infinity) = 'epsilon':

  lhs(%): subs(beta=newBeta,%): eval(%,tstData): evalf(%): expand(%):

  % = 'epsilon'; Digits:=20:
  fsolve(%%,cutoff,0..infinity):
  ``;
  `cutoff`=evalf(%%,3);
  epsilon:='epsilon': Digits:=14:
```

$$\varepsilon := 0.1\ 10^{-15}$$

$$\int_{\mathrm{cutoff}}^{\infty} \frac{\mathbf{e}^{((-\alpha+\beta)\,x)}}{x^{(3/2)}}\,dx \leq \varepsilon$$

$$-8.661934897 + \frac{2.000000000\,\mathbf{e}^{(-5.970627363\,\mathrm{cutoff})}}{\sqrt{\mathrm{cutoff}}} + 8.661934897\,\mathrm{erf}(2.443486722\,\sqrt{\mathrm{cutoff}}) = \varepsilon$$

$$\mathrm{cutoff} = 5.44$$

```
>
>
```