From a brute RND to a NIG estimation

AVt, Apr 2006

```
> restart;
> with(avtbslib): read cat(myLib,"BesselK_numeric.mpl"):
  kernelopts(version);
  Digits:=14;
```

Maple 10.03, IBM INTEL NT, Mar 24 2006 Build ID 222373

$$Digits := 14$$

Data for the Example

For the brute way see a separate sheet ...

A brute way to get a RND from option prices

Norming: fwd = 1, time = 1, rates = 0

```
>
```

Lazy RND and PDF for the normed situation

```
>
```

Extending to the full range

```
>
```

Pricing with the PDF

```
>
```

Statistics for that PDF

```
>
```

NIG

The normal inverse Gauss (with a shift) has 3 parameters and the following density

```
> alpha*delta/Pi*exp(delta*(alpha^2-beta^2)^(1/2)+beta*(x-mu))*BesselK(1,alp
  ha*(delta^2+(x-mu)^2)^(1/2))/(delta^2+(x-mu)^2)^(1/2):
  pdf_NIG:= unapply(%, x, alpha, beta, delta, mu);
  ``;
  Assumptions_NIG:=
    (0<alpha, (beta<alpha), (-beta<alpha), 0<delta, mu::real);
```

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

$$Assumptions\_NIG := 0 < \alpha, \beta < \alpha, -\beta < \alpha, 0 < \delta, \mu{::}real$$

where for estimations it is convenient to explicitely enforce positivity for roots and as Maple is slow with Bessel functions a purely numerical routine will increase speed

```
> PDF_NIG := (x, alpha, beta, delta, mu) -> alpha*delta/Pi *
    exp(delta*abs(alpha^2-beta^2)^(1/2)+beta*(x-mu))*

  BesselK(1,alpha*abs(delta^2+(x-mu)^2)^(1/2))/abs(delta^2+(x-mu)^2)^(1/2);
  ``;
  PDF_NIG_num := (x, alpha, beta, delta, mu) -> alpha*delta/evalf(Pi) *
    exp(delta*abs(alpha^2-beta^2)^(1/2)+beta*(x-mu))*

  BesselK1(alpha*abs(delta^2+(x-mu)^2)^(1/2))/abs(delta^2+(x-mu)^2)^(1/2);
```

```
``;
id:= x -> x: # to get rid of absolute signs
```

$$\mathrm{PDF\_NIG} := (x, \alpha, \beta, \delta, \mu) \rightarrow \frac{\alpha\,\delta\,\mathbf{e}^{(\delta\sqrt{|\alpha^2-\beta^2|}+\beta\,(x-\mu))}\,\mathrm{BesselK}(1, \alpha\sqrt{|\delta^2+(x-\mu)^2|})}{\pi\sqrt{|\delta^2+(x-\mu)^2|}}$$

$$\mathrm{PDF\_NIG\_num} := (x, \alpha, \beta, \delta, \mu) \rightarrow \frac{\alpha\,\delta\,\mathbf{e}^{(\delta\sqrt{|\alpha^2-\beta^2|}+\beta\,(x-\mu))}\,\mathrm{BesselK1}(\alpha\sqrt{|\delta^2+(x-\mu)^2|})}{\mathrm{evalf}(\pi)\sqrt{|\delta^2+(x-\mu)^2|}}$$

Then the risk neutral density over the strikes is given by

```
> RND_NIG:= (K, alpha, beta, delta, mu) -> pdfNIG(
  ln(abs(K)),alpha,beta,delta,mu)/K;
```

$$\mathrm{RND\_NIG} := (K, \alpha, \beta, \delta, \mu) \rightarrow \frac{\mathrm{pdfNIG}(\ln(|K|), \alpha, \beta, \delta, \mu)}{K}$$

Initial guessing and mean correction

The moments for NIG are known explicitely ($\mu = 0$):

```
> NIG_Mean:=delta*beta/sqrt(alpha^2-beta^2);
  NIG_Var:=alpha^2*delta*(alpha^2-beta^2)^(-3/2);
  NIG_Skew:=3*beta*alpha^(-1)*delta^(-1/2)*(alpha^2-beta^2)^(-1/4);
  NIG_Kurt:=3*(1 + (alpha^2+4*beta^2)/(delta*alpha^2*sqrt(alpha^2-beta^2)));
```

$$\mathrm{NIG\_Mean} := \frac{\delta\,\beta}{\sqrt{\alpha^2-\beta^2}}$$

$$\mathrm{NIG\_Var} := \frac{\alpha^2\,\delta}{(\alpha^2-\beta^2)^{(3/2)}}$$

$$\mathrm{NIG\_Skew} := \frac{3\,\beta}{\alpha\sqrt{\delta}\,(\alpha^2-\beta^2)^{(1/4)}}$$

$$\mathrm{NIG\_Kurt} := 3 + \frac{3\,(\alpha^2+4\,\beta^2)}{\delta\,\alpha^2\sqrt{\alpha^2-\beta^2}}$$

Having the statistics for a brutely guessed PDF one can numerical solve for the parameters:

```
> '[PDF_Mean, PDF_Var,PDF_Skew,PDF_Kurt]' =
  [PDF_Mean,PDF_Var,PDF_Skew,PDF_Kurt];
  ``;
  'fsolve({NIG_Var=PDF_Var,NIG_Skew=PDF_Skew,NIG_Kurt=PDF_Kurt},
    {alpha,beta,delta})':
  NIG_estim:=convert(%, list);
```

$$[\mathrm{PDF\_Mean}, \mathrm{PDF\_Var}, \mathrm{PDF\_Skew}, \mathrm{PDF\_Kurt}] = [-0.0031280202660226, 0.0067303643999048, -1.0877266683135, 5.8795009225031]$$

$$\mathrm{NIG\_estim} := [\delta = 0.10397362561475, \alpha = 26.543248402873, \beta = -14.608747386882]$$

The mean might not have the correct value, but after shifting by a new one it does (almost):

```
> `mean of that NIG`=eval(NIG_Mean,NIG_estim);
  ``;
  mu_new:='delta*(sqrt(alpha^2-(beta+1)^2) - sqrt(alpha^2-beta^2))';
  'mu_new'=eval(%, NIG_estim);
```

mean of that NIG = -0.068539129388999

$$\text{mu\_new} := \delta\,(\sqrt{\alpha^2 - (\beta+1)^2} - \sqrt{\alpha^2 - \beta^2})$$

mu_new = 0.065268900382264

One can take that parameters as an initial guess to estimate against data

```
> initParams:=[alpha = 26.543248402850, beta = -14.608747386865, delta =
  .10397362561471]; InitParams:=eval([alpha,beta,delta,mu_new],initParams);
```

$$\text{initParams} := [\alpha = 26.543248402850, \beta = -14.608747386865, \delta = 0.10397362561471]$$

$$\text{InitParams} := [26.543248402850, -14.608747386865, 0.10397362561471, 0.065268900382239]$$

and they satisfy the parameter restrictions needed:

```
> [Assumptions_NIG]; eval(%, initParams): map(is, %) assuming mu::real;
```

$$[0 < \alpha, \beta < \alpha, -\beta < \alpha, 0 < \delta, \mu\text{::real}]$$

[true, true, true, true, true]

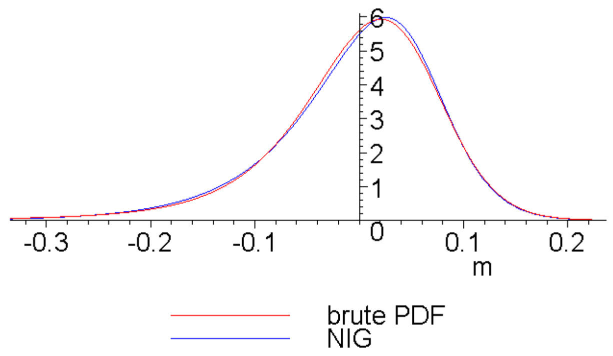
Fitting the NIG

Prepare for fitting and check the initial guess against the 'real' data which is the pdf given through option prices and a polynomial vola interpolation:

```
> arrayPDF:= [seq( evalf(PDF(arrayMoneyness[i])),i=1..nData)]:;
  arrayLogPDF:= [seq( evalf(-ln(PDF(arrayMoneyness[i]))),i=1..nData)]:

  residues:= [seq(
    (PDF_NIG(arrayMoneyness[i], alpha, beta, delta, mu_new)) -
    arrayPDF[i],
    i =1 ..nData)]:

  ' (PDF_NIG(m, alpha, beta, delta, mu_new))';
  eval(%, initParams): subs(abs=id,%): eval(%);
  plot(%, m=minMny..maxMny,color=blue, legend="NIG"):
  plot(PDF(m), m=minMny..maxMny, legend="brute PDF"):
  plots[display](%,%%);
```

$$\text{PDF\_NIG}(m, \alpha, \beta, \delta, \text{mu\_new})$$

$$2.7597977720362\,\mathbf{e}^{(3.2577000620691 - 14.608747386865\,m)}\,\text{BesselK}(1, 26.543248402850\sqrt{0.010810514823468 + (m - 0.065268900382239)^2}) \Big/ (\pi\sqrt{0.010810514823468 + (m - 0.065268900382239)^2})$$



This is already quite close (the red line comes through Breeden Litzenberger and a vola interpolation over the 'observed' strikes). But it can be improved by least-square fitting:

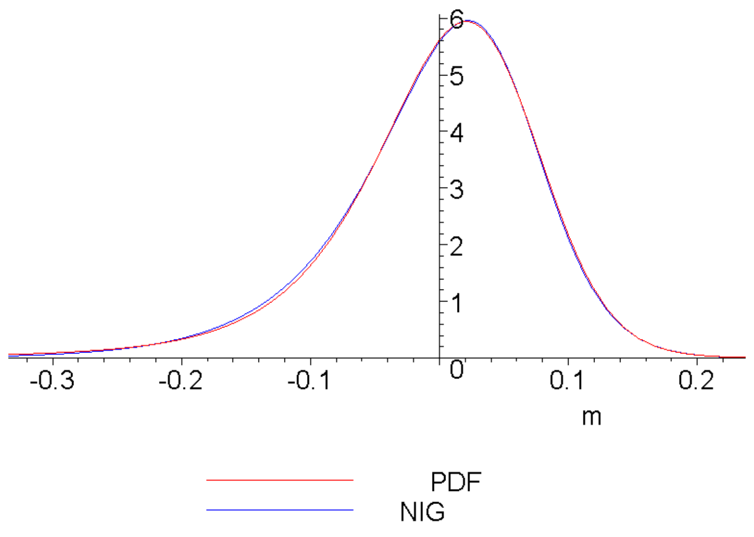
```
> infolevel[`Optimization` ]:=5:
  sol := Optimization:-LSSolve(
    residues
    ,initialpoint=initParams
    ,iterationlimit =100
  ):
  infolevel[`Optimization` ]:=0:

LSSolve:   calling nonlinear LS solver
SolveUnconstrained:   using method=modifiednewton
SolveUnconstrained:   number of problem variables   3
SolveUnconstrained:   number of residuals   41
PrintSettings:   optimality tolerance set to   .3256082241e-11
PrintSettings:   iteration limit set to   100
SolveUnconstrained:   trying evalhf mode
SolveUnconstrained:   trying evalf mode
E04FCF:   conditions for a minimum are not all satisfied, but a better point
could not be found
E04FCF:   number of major iterations taken   5
```

The result looks almost perfect:

```
> ' (PDF_NIG(m, alpha, beta, delta, mu_new))';
  eval(%, sol[2]): subs(abs=id,%): eval(%);
  plot(%, m=minMny..maxMny,color=blue, legend="NIG"):
  plot(PDF(m), m=minMny..maxMny, legend="    PDF"):
  plots[display](%,%%);
```

$$\mathrm{PDF\_NIG}(m, \alpha, \beta, \delta, \mathrm{mu\_new})$$

$$3.0186544979690\, \mathbf{e}^{(3.4548503331792 - 13.6774892379700718\, m)}$$
$$\mathrm{BesselK}\left(1, 27.0791855213980277 \sqrt{0.012426694495404 + (m - 0.062113326815885)^2}\right) \Big/ \left(\pi \sqrt{0.012426694495404 + (m - 0.062113326815885)^2}\right)$$



Check Prices for that pdf

The NIG model is known to be arbitrage free and a may be found in Schoutens book "Levy Processes for Finance", the shift used is called Mean-Correcting Martingale, p.79.

So let's look how good prices are reproduced. Pricing a Call is by

```
> 'Spot*Int((exp(xi)-K/Fwd)*pdf(xi),xi = ln(K/Fwd) .. infinity)';
```

$$\mathrm{Spot}\int_{\ln\left(\frac{K}{\mathrm{Fwd}}\right)}^{\infty}\left(\mathbf{e}^{\xi}-\frac{K}{\mathrm{Fwd}}\right)\mathrm{pdf}(\xi)\,d\xi$$

since the pdf is over log moneyness and fitting was done in a normed situation.

Using Maple (and not a pure, coded routine) is a bit messy, but anyway first find where to cut the integral (cutLower and cutUpper have been used for the brute way and since the NIG tails are fatter one has to cut later):

```
> '(PDF_NIG(2*cutLower, alpha, beta, delta, mu_new))':
    '%'= evalf(eval(%, sol[2]));
  '(PDF_NIG(2*cutUpper, alpha, beta, delta, mu_new))':
    '%'= evalf(eval(%, sol[2]));
```

$$\mathrm{PDF\_NIG}(2\,\mathrm{cutLower}, \alpha, \beta, \delta, \mathrm{mu\_new}) = 0.52270854665982\ 10^{-19}$$

$$\mathrm{PDF\_NIG}(2\,\mathrm{cutUpper}, \alpha, \beta, \delta, \mathrm{mu\_new}) = 0.88914639883597\ 10^{-27}$$

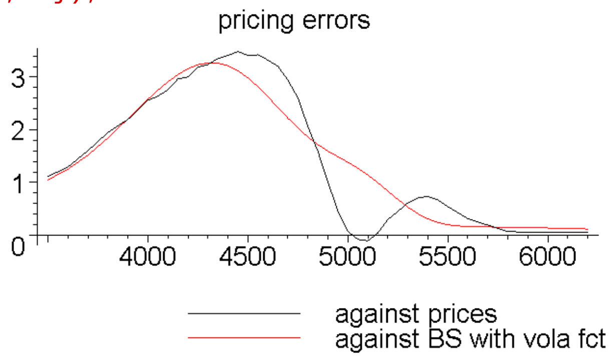
Then the following routine calculates call prices in reasonable time using global variables:

```
> SolParams:=eval([alpha,beta,delta,mu_new],sol[2]):
  SolParams_hf:=hfarray(1..4, SolParams);

  C_NIG:=proc(strike)
  local x,result; # theIntegrand;
  global SolParams_hf,
    m0, alpha0,beta0,delta0,mu0;

  m0:=evalhf(ln(strike/Fwd));
  alpha0:= evalf(SolParams_hf[1]);
  beta0:=  evalf(SolParams_hf[2]);
  delta0:= evalf(SolParams_hf[3]);
  mu0:=    evalf(SolParams_hf[4]);

  evalf(Int(
    'x ->
      (exp(x)-exp(m0))*alpha0*delta0/evalhf(Pi) *
      exp(delta0*(alpha0^2-beta0^2)^(1/2)+beta0*(x-mu0))*

  BesselK1(alpha0*(delta0^2+(x-mu0)^2)^(1/2))/(delta0^2+(x-mu0)^2)^(1/2)',
    m0 .. 2*cutUpper, #0 .. infinity,
    method = _d01ajc #method = _d01amc
   ));
  result:=evalhf(Spot*%);
  alpha0:=evalhf(0): beta0:=evalhf(0): delta0:=evalhf(0): mu0:=evalhf(0):
  m0:=evalhf(0):
  return(result);
  end proc:
```

```
SolParams_hf :=
    [27.0791855213980277, -13.6774892379700718, 0.111475084639587718, 0.0621133268158850019]
```

So we can check pricing errors over the whole range

```
> arrK:=[seq(arrData[i][1], i=1..nData ) ]:
  arrayCallsOriginal:=map( x -> x[2], arrData):
  nigPrices:=map(C_NIG,arrK):

  [seq([arrK[i], arrayCallsOriginal[i] - nigPrices[i]], i = 1..nData)]:
  P1:=plots[pointplot](%,
    color=black, symbol = circle, style=line, legend="against prices"):

  pricingErrors:=[seq(
    [arrK[i], BSCall(Spot,arrK[i],Time,Rates,volaFct(arrK[i])) -
  nigPrices[i]],
    i=1..nData)]:

  P2:=plots[pointplot](%,
    color=red, style=line, title="pricing errors", legend="against BS with
  vola fct"):
  plots[display]({P1,P2});
```



Check that at a large error. For strike = 4500 the price is given as

```
> 'eval(Spot*Int((exp(xi)-K/Fwd)*PDF_NIG(xi, alpha, beta, delta, mu_new),
    xi = ln(K/Fwd) .. infinity), K=4500)';
```

$$\left. \mathrm{Spot} \int_{\ln\left(\frac{K}{Fwd}\right)}^{\infty} \left(e^{\xi} - \frac{K}{Fwd}\right) \mathrm{PDF\_NIG}(\xi, \alpha, \beta, \delta, \mathrm{mu\_new})\, d\xi \right|_{K = 4500}$$

which is computed with a special integration method here and it gives

```
> 'Spot*Int((exp(xi)-K/Fwd)*PDF_NIG(xi, alpha, beta, delta, mu_new),
    xi = ln(K/Fwd) .. infinity, method=_NCrule)':
  eval(%, sol[2]): subs(abs=id,%): eval(%):
  eval(%,K=4500):
  evalf(%);
```

$$424.68823285184$$

which is the same as given by the pricing routine

```
> 'C_NIG(4500)': '%'=%;
```

$$\mathrm{C\_NIG}(4500) = 424.688232851852206$$

The original data however are **[4500.00, 428.10, 40.60, 18.700]** which is an error of 3 Euro or almost 1% for prices as the graphic also tells.

Note that the vola used is also 'correct':

```
> 'volaFct(4500.0)': '%'=%;
```

$$\mathrm{volaFct}(4500.0) = 0.18623787742268$$

So the graphics correctly reports serious pricing errors.

Improvement for pricing

One can improve the parameters to give better prices and I just give a result of a least square minimizer (using a descent method to fit against vola, weighted by vega to get good approximations around the forward):

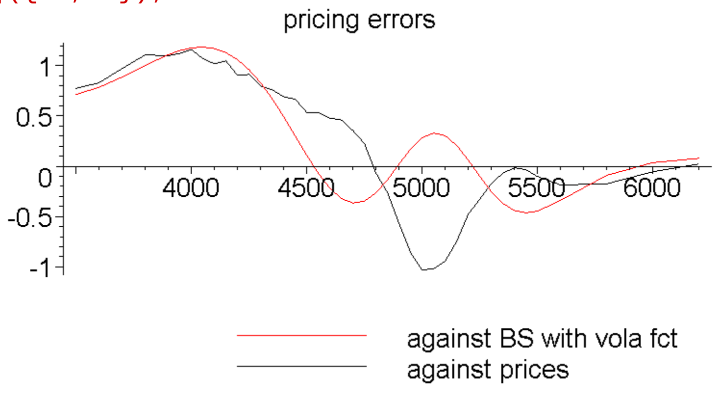
```
> sol_desc := [alpha = 22.410682349940, beta = -11.185865530124, delta =
  .988146272757705e-1]:

  SolParams:=eval([alpha,beta,delta,mu_new],sol_desc):
  SolParams_hf:=hfarray(1..4, SolParams);
```

SolParams_hf :=

$$[22.4106823499400001, -11.1858655301239996, 0.0988146272757050026, 0.0536250526489010018]$$

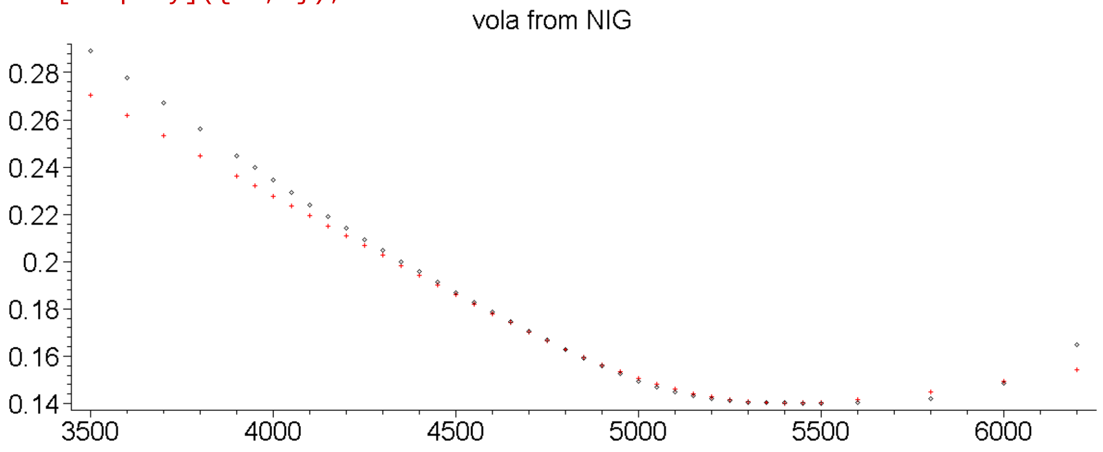
That changes the global parameters for the pricing routine, now plot the new pricing errors:

```
> nigPrices_new:=map(C_NIG,arrK):

  seq([arrK[i], arrayCallsOriginal[i] - nigPrices_new[i]], i = 1..nData):
  P1:= plots[pointplot]([%],
    color=black, symbol = circle, style=line, legend="against prices"):

  pricingErrors_new:=evalf([seq(
    [arrK[i], BSCall(Spot,arrK[i],Time,Rates,volaFct(arrK[i])) -
  nigPrices_new[i]],
    i=1..nData)]):
  P2:=plots[pointplot](%,color=red, style=line,
    title="pricing errors", legend="against BS with vola fct"):
  plots[display]({P1,P2});
```



Certainly much better - but not convincing ... and the vola?

```
> vola_NIG:=evalf([seq(
    [arrK[i],
  Re(BSCallVolaNum(Spot,arrK[i],Time,Rates,nigPrices_new[i]))],
    i=1..nData)]):
  plots[pointplot](%, symbol=CROSS, color = red, title="vola from NIG"):
  plots[pointplot](map( x -> [x[1],x[4]/100], arrData)): # original vola
  plots[display]({%%,%});
```

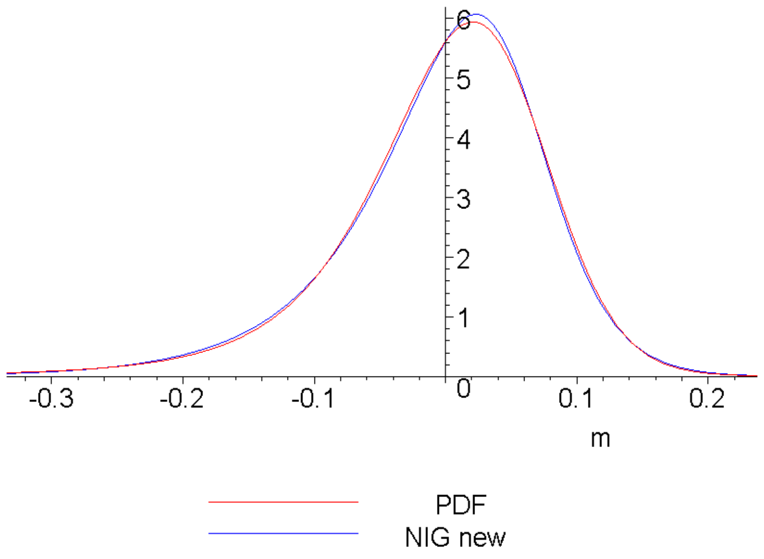


Compare the pdf for that better parameters against the pdf given through vola approximation:

```
> ' (PDF_NIG(m, alpha, beta, delta, mu_new))';
  eval(%, sol_desc): subs(abs=id,%): eval(%);
  plot(%, m=minMny..maxMny,color=blue, legend="NIG new"):
  plot(PDF(m), m=minMny..maxMny, legend="    PDF"):
  plots[display](%,%%);
```

$$\mathrm{PDF\_NIG}(m, \alpha, \beta, \delta, \mathrm{mu\_new})$$

$$2.2145032234035\, \mathbf{e}^{(2.5187684877384 - 11.185865530124\, m)} \mathrm{BesselK}\left(1, 22.410682349940 \sqrt{0.0097643305636365 + (m - 0.053625052648901)^2}\right) \Big/ \left(\pi \sqrt{0.0097643305636365 + (m - 0.053625052648901)^2}\right)$$



The additional cost: the pdf does not fit that nice ...

Even if no real trading data have been used (these are settlement data), the above tells that a good looking fit for a pdf does not mean a good fit for vola or pricing data.

Anyway: do not forget the input is a bit questionable (both the data and the vola approximation) and NIG is an arbitrage free model, so one can not expect a priori an 'exact' fit, neither for price data nor for the pdf (or RND).