Gatheral's SVI approximation

Some (=1) example for Gatheral's variance approximation. Data are for Dax, Eurex settlement 29. July 2003.

Ref: Jim Gatheral, A parsimonious arbitrage-free implied volatility parameterization
with application to the valuation of volatility derivatives (June 2004).

```
> restart;
> Digits:=14: # to use hardware floatings
  with(plots): with(Optimization): # libraries need
```

Read and prepare data

Load data, market data and option volatilities are stored in separate files.
Note that the volatilities there are in %-points.

```
> currentdir():
  params:=cat(%,`\\param_22Jul03.txt`);
  data:= cat(%%,`\\data_22Jul03.txt`);
```

```
params := "C:\temp\param_22Jul03.txt"

data := "C:\temp\data_22Jul03.txt"
```

```
> # read in parameters for the expiries with 11 columns as
  #
  keyNr ExpiryDate Date time Dax Future Forward rate SeriesCounted downExcer
   upExcer
  #  1     2          3    4    5   6      7       8    9             10
  11
  fd := fopen(params,READ,TEXT):
  inParam:=readdata(fd, 11): # 11 columns
  fclose(fd):
  `expiries used` = nops(inParam); # how many expiries are used
```

```
expiries used = 12
```

```
> # read in option data, 2 columns
  # strike, vola in %
  #  1       2
  fd := fopen(data,READ,TEXT):
  inData:=readdata(fd, 2): # 2 columns
  fclose(fd):
  `strikes used`= nops(inData); # how many strikes are used
```

```
strikes used = 302
```

```
> # a small routine to determine at which position a new expiry starts
  # group change is given in the parameter file
  myStart:=proc(i::posint)
    local j, iOut;
    if i=1 then iOut:=0: # <--- !
    else iOut:=add(floor(inParam[j-1,9]),j=2..i):
    end if:
    iOut;
  end proc:

  # and now group data according to experies
  allData:= [seq (
    [seq( inData[s], s=myStart(i)+1..floor(myStart(i)+inParam[i,9]) )],
  i=1..12)]:
  originalData:=allData:
```

a little check for expiry no 2, which is SEP03 (19 Sep 2003)

```
> i:=2: allData[i]; `strikes`=nops(%);
```

```
  inParam[i]: `days until expiry`= (%[2]-%[3]); ``; i:='i':
[[1000., 91.90934689], [1200., 78.35167286], [1400., 66.91300713], [1600., 60.31822374],
    [1800., 55.62485748], [2000., 50.82906804], [2100., 48.48733938], [2200., 46.95318283],
    [2250., 45.80432482], [2300., 44.30567303], [2350., 43.06379848], [2400., 42.24679733],
    [2450., 41.64913518], [2500., 40.03246326], [2550., 38.7976168], [2600., 38.1902077],
    [2650., 37.0658179], [2700., 35.80421072], [2750., 35.5361623], [2800., 34.41713068],
    [2850., 33.88505641], [2900., 33.25353346], [2950., 32.2145758], [3000., 31.38398186],
    [3050., 30.78696344], [3100., 29.90882832], [3150., 29.42137707], [3200., 28.51799403],
    [3250., 28.1346747], [3300., 27.3492621], [3350., 27.1945772], [3400., 26.33625863],
    [3450., 26.13504918], [3500., 25.46419359], [3550., 25.13236192], [3600., 24.67427843],
    [3650., 24.5078563], [3700., 24.06276457], [3800., 23.55074498], [3900., 23.53770316],
    [4000., 24.05267137], [4100., 24.82280142], [4200., 25.07322512], [4300., 25.53910103],
    [4400., 26.52966927], [4500., 26.98236138], [4600., 27.66464918], [4800., 29.14842886],
    [5000., 32.05231864], [5200., 34.81788107], [5400., 37.45710829]]
                                   strikes = 51
                          days until expiry = 58.70834
```

Replace the volatilities by variances and strikes by moneyness
run through all expiries and accordingly set the x and y values,
x = ln(strike/fwd), y = vol^2 * time/365.25

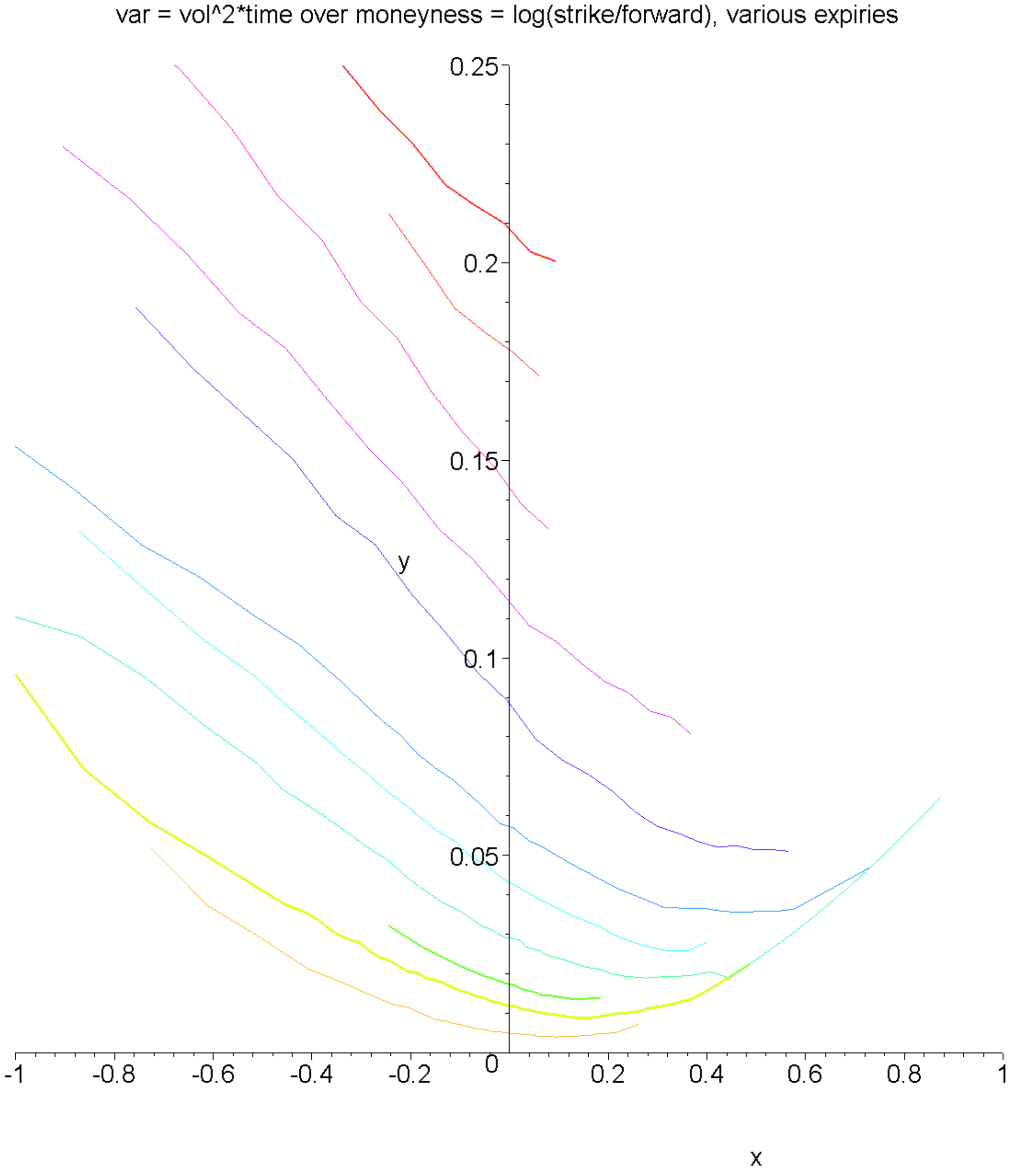
```
> mny_var_Data:=copy(allData): # take a copy and over-write it
  for i from 1 to 12 do
    for j from 1 to nops(allData[i]) do
      # fwd = column 7, time in days = column 4
      mny_var_Data[i][j,1]:= ln(allData[i][j,1]/inParam[i,7]) ;
      mny_var_Data[i][j,2]:= (allData[i][j,2]/100)^2 * (inParam[i,4]/365.25)
  ;
    end do;
  end do;
```

Plot variances

look at the variance smiles first

```
> # define the plots the simple way
  P1:=plot(mny_var_Data[1],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE, 0.1)):
  P2:=plot(mny_var_Data[2],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE,
  0.2),thickness=3):
  P3:=plot(mny_var_Data[3],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE,
  0.3),thickness=2):
  P4:=plot(mny_var_Data[4],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE, 0.4)):
  P5:=plot(mny_var_Data[5],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE, 0.5)):
  P6:=plot(mny_var_Data[6],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE, 0.6)):
  P7:=plot(mny_var_Data[7],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE, 0.7)):
  P8:=plot(mny_var_Data[8],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE, 0.8)):
  P9:=plot(mny_var_Data[9],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE, 0.9)):
  P10:=plot(mny_var_Data[10],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE, 1.0)):
  P11:=plot(mny_var_Data[11],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE,
  1.2),thickness=2):
  P12:=plot(mny_var_Data[12],x=-1.0..1.0,y=0..0.25,colour=COLOR(HUE,
  1.3),thickness=3,
    labels=["moneyness", "variance"],
    title='`var = vol^2*time over moneyness = log(strike/forward), various
  expiries`'):
```

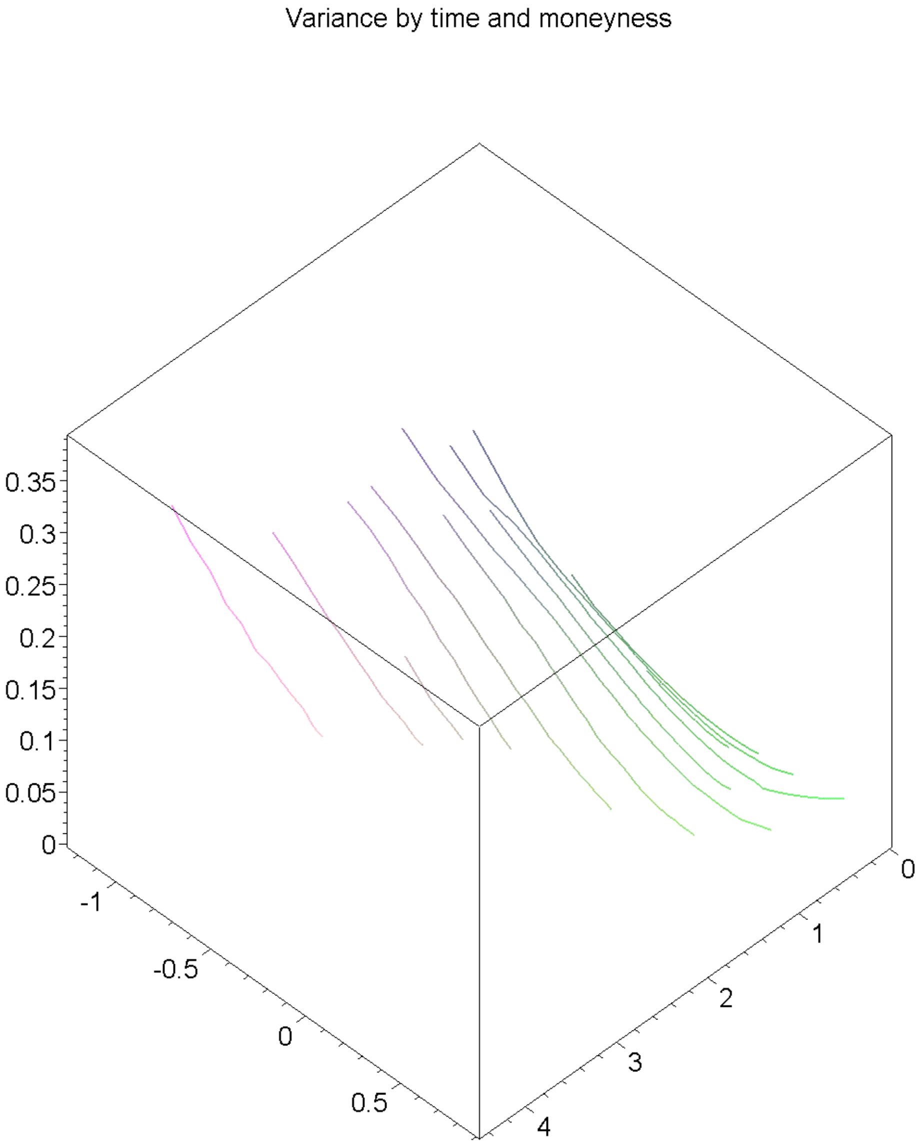
```
> display({P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12});
```



To see it better make a 3 dim picture by time scaling:

```
> # resort data in experies
  tstData:= [seq (
    [seq( inData[s], s=myStart(i)+1..floor(myStart(i)+inParam[i,9]) )],
  i=1..12)]:

  # sort experies by their time
  i:=1: j:=1:
  nData:=[ seq(
    [seq( [inParam[i,4]/365.25,
           ln(tstData[i][j,1]/inParam[i,7]),
           (tstData[i][j][2]/100)^2 * (inParam[i,4]/365.25)]
          ,j=1..nops(tstData[i]))], i=1..12)]:;

  # define plots
  for i from 1 to 12 do
  R||i:=nData[i];
```

```
end do:

# plot them
PLOT3D(CURVES(R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,THICKNESS(2)),
AXESSTYLE(BOX),
 TITLE(`Variance by time and moneyness`));
```



The 'largest' smile is SEP03, we will do fitting for it in the following

Fitting Gatheral's variance function

His function for moneyness k = log(strike/forward) is defined as

```
> var:='a + b*(rho*(k-m)+sqrt((k-m)^2+sigma^2))':
  g:=unapply('%', k, a,b,sigma,rho,m);
```

$$g := (k, a, b, \sigma, \rho, m) \rightarrow a + b\,(\rho\,(k - m) + \sqrt{(k - m)^2 + \sigma^2}\,)$$

where the parameters have the following (geometric) meaning:

```
 a gives the overall level of variance
 b gives the angle between the left and right asymptotes
```

σ determines how smooth the vertex is
ρ determines the orientation (rotation) of the graph
m translates the graph

We want to fit for SEP03 and as expiry is in 2 month it is a good idea to kick out far OTMs, so i restrict to | moneyness | <= 0.6 (naming that data2):

```
> myFilter:= proc(x) abs(x[1])<=0.6; end proc:
  data:=mny_var_Data[2]:
  data2fit:=select(myFilter,data):;
```

Now i want to use Maple's (9.5) least square solver, for that some initial values have to be set (rough guess only):

```
> p:=subs(k=x,var):
  residues := map((d) -> eval(p, x=d[1])-d[2], data2fit):
  IP:={a=0.0,b=0.2,sigma=0.4,rho=0.0,m=0.0};
  sol := LSSolve(residues,initialpoint=IP): #
```

$$IP := \{ b = 0.2, \sigma = 0.4, \rho = 0., m = 0., a = 0. \}$$

The estimated approximation is:

```
> fct := eval(p, sol[2]):
  gApprox:=unapply(fct,x);
```

$$gApprox := x \to -0.11157757854210 + 0.062157725759513\,x + 0.168498560260931768\sqrt{x^2 - 0.81268269312714\,x + 0.53723827393110}$$

First check the simpliest condition for non-arbitrage:

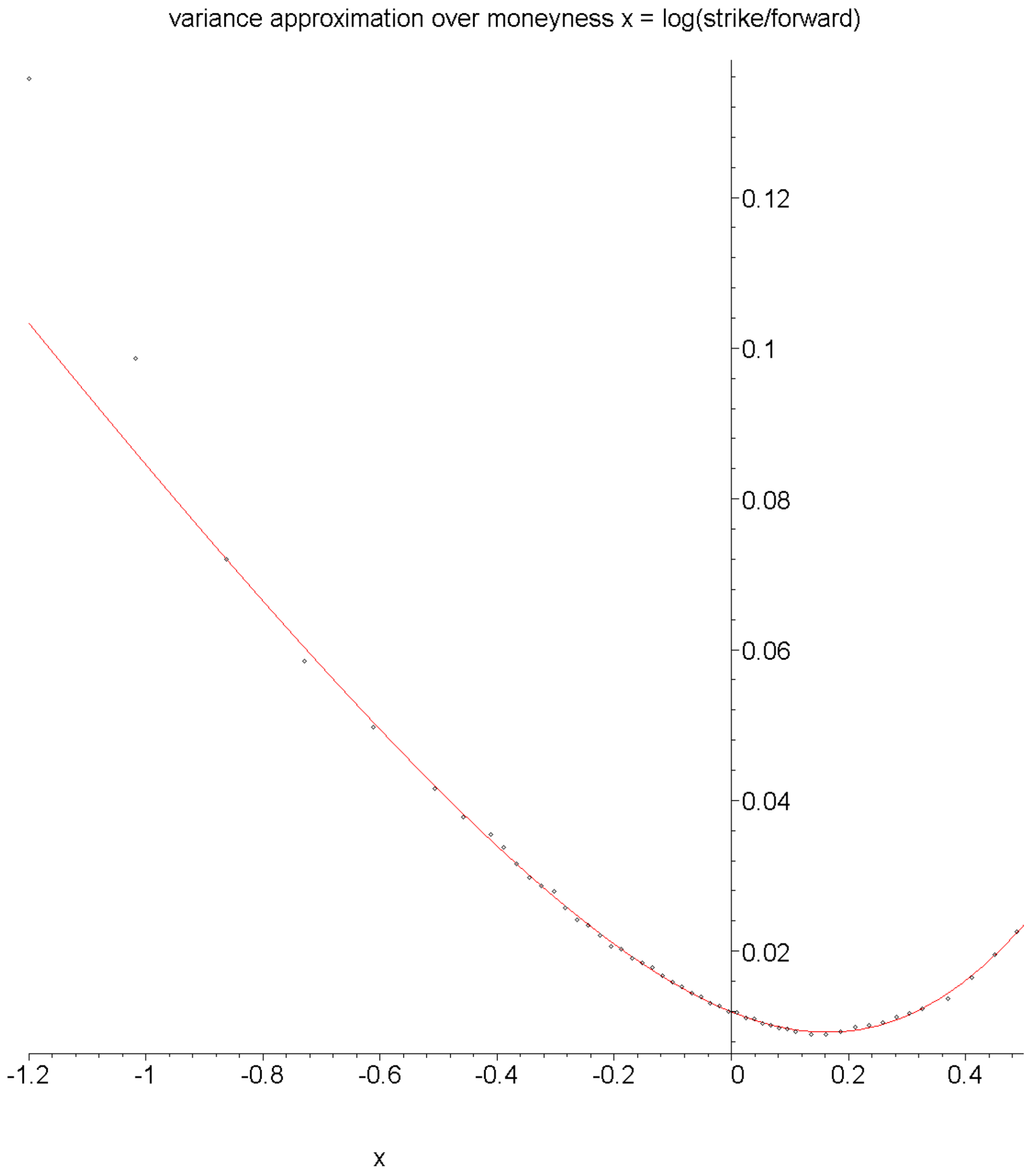
```
> b*(1+abs(rho)) <= 4 / T;
  eval(%,sol[2]):
  eval(%,T=inParam[2][4]/365.25): # time in years
  is(%);
```

$$b\,(1 + |\rho|) \le \frac{4}{T}$$

$$true$$

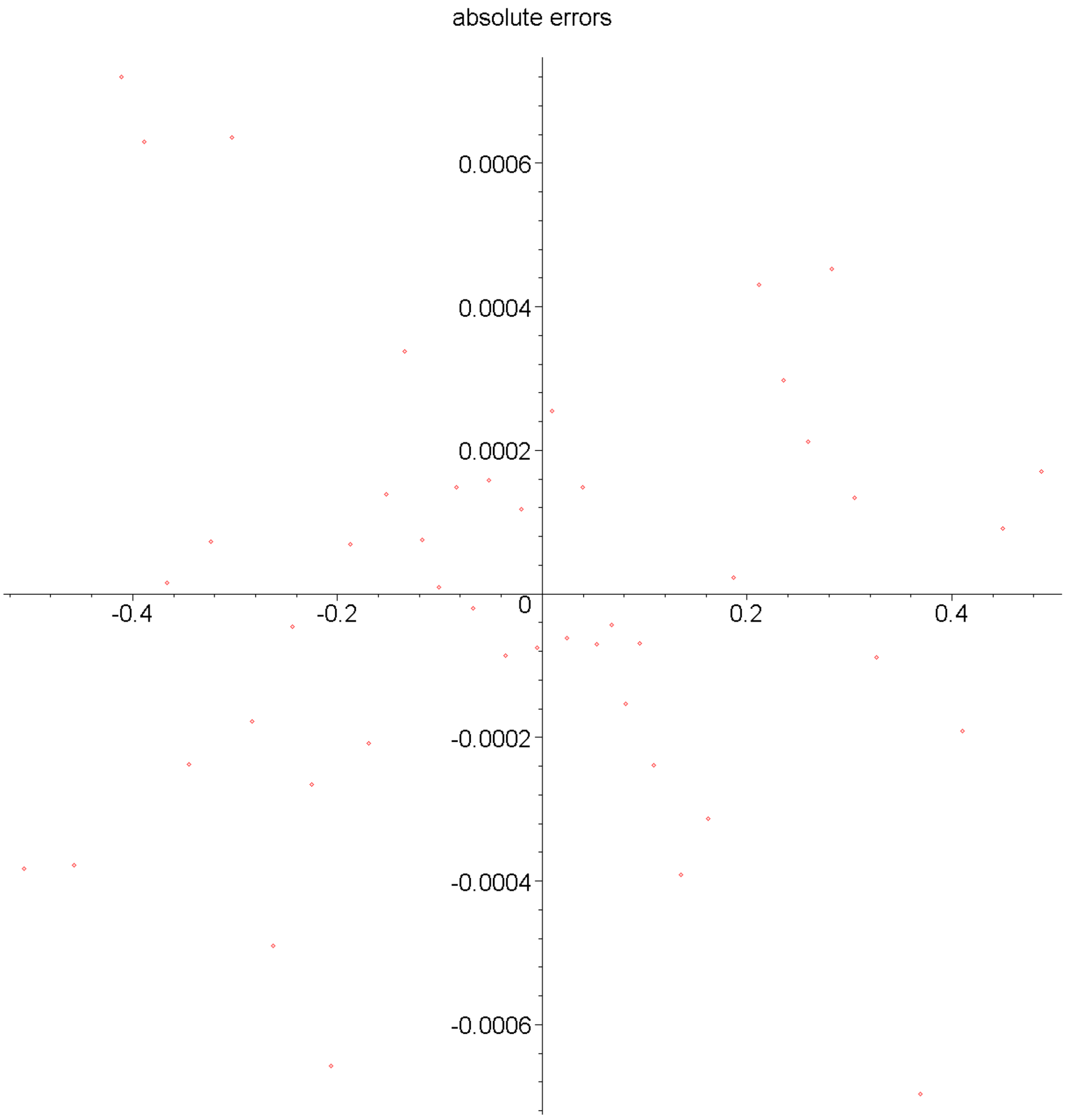
As it is ok now plot the approximation against the data (remind: |moneyness| <= 0.6 assumed)

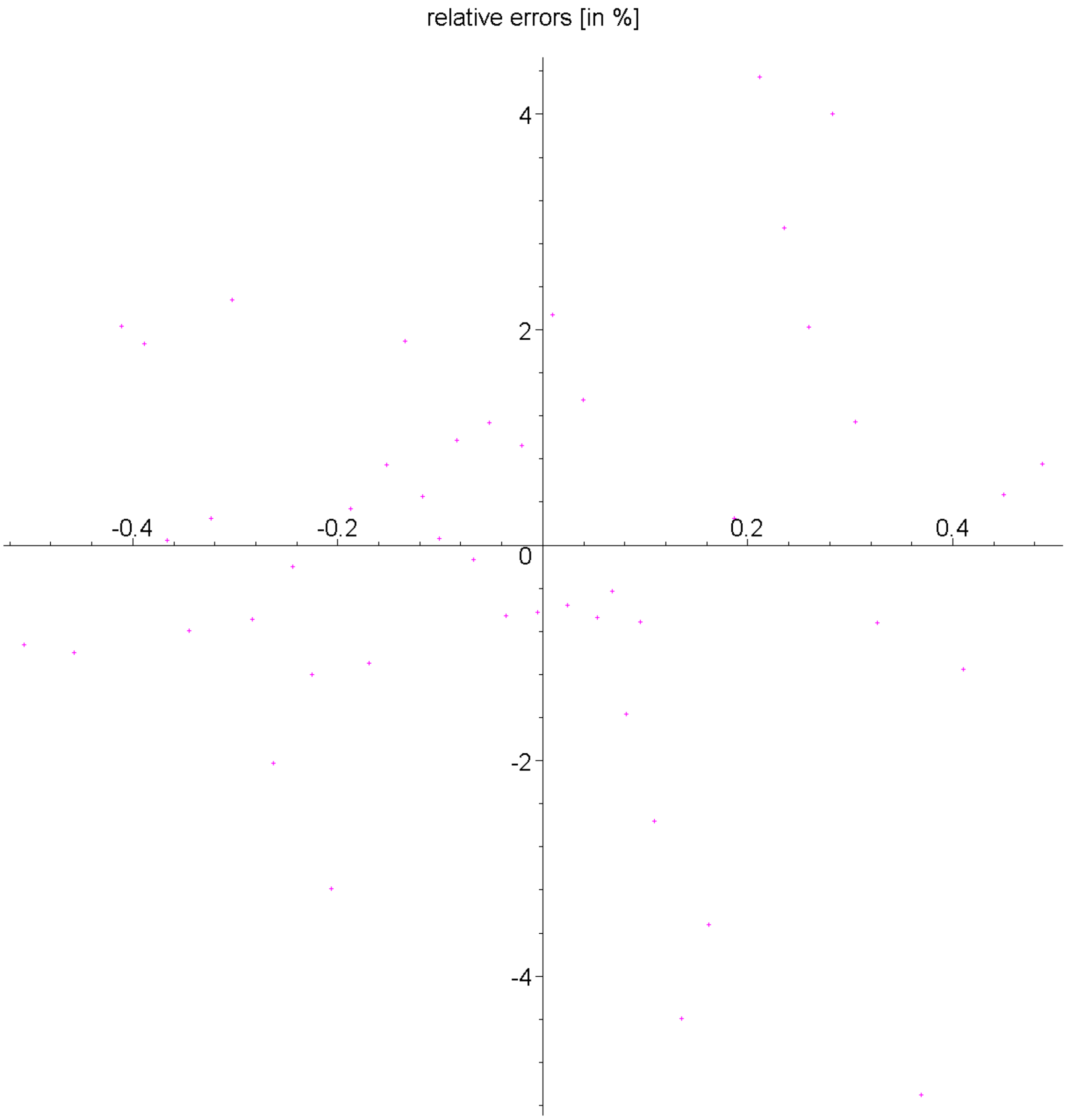
```
> p1 := pointplot(data):
  p2 := plot(fct, x=-1.2..0.5,
    title=`variance approximation over moneyness x = log(strike/forward)`):
  display(p1, p2);
```



That looks quite nice, so plot the approximation errors:

```
> [seq( [ data2fit[i][1],data2fit[i][2] - gApprox(data2fit[i][1]) ] ,
    i=1..nops(data2fit) )]:;
  plot(%, style=POINT,title=`absolute errors`);

  [seq( [ data2fit[i][1], 100*(data2fit[i][2] -
  gApprox(data2fit[i][1]))/data2fit[i][2] ] ,
    i=1..nops(data2fit) )]:;
  plot(%, style=POINT, symbol=CROSS, colour=magenta,
    title=`relative errors [in %]`);
```


absolute errors
0.0006
0.0004
0.0002
0
-0.0002
-0.0004
-0.0006
-0.4
-0.2
0.2
0.4



```
>
```

⊟ Switching back to volatility

What i actually want is volatility for pricing, so look at the ATMF to check this, forward and time are given as

```
> Fwd:= inParam[2][7];
  Time:=inParam[2][4]/365.25;
```

$$Fwd := 3317.880454$$

$$Time := 0.16073465661875$$

so the traded ATMF is item 30, which has strike = 3300 and vol = 27.35%

```
> expiryNo:=2;
  itemNo:=30;
  originalData[expiryNo][itemNo]; # originalData[2][30];
  originalData[2][24];
```

$$expiryNo := 2$$

$$itemNo := 30$$

$$[3300., 27.3492621]$$

solve the approximation at ATMF for volatility:

```
> 'gApprox( ln(strike/forward))' = volatility^2*lifetime;
```

```
  subs(strike=originalData[expiryNo][itemNo][1], forward=Fwd, lifetime=Time,
  %):;
  fsolve(%,volatility):
  `100*volatility ATMF`=100*abs(%[1]); rhs(%):
  `estimation error for volatility`=originalData[expiryNo][itemNo][2] - %;
  rhs(%):
  `relative error for volatility [%]`=100*%/originalData[2][itemNo][2];
```

$$\mathrm{gApprox}\left(\ln\left(\frac{\mathrm{strike}}{\mathrm{forward}}\right)\right) = \mathrm{volatility}^2\ \mathrm{lifetime}$$

$$\text{100*volatility ATMF} = 27.433953486164$$

$$\text{estimation error for volatility} = -0.084691386164$$

$$\text{relative error for volatility [\%]} = -0.30966607382069$$

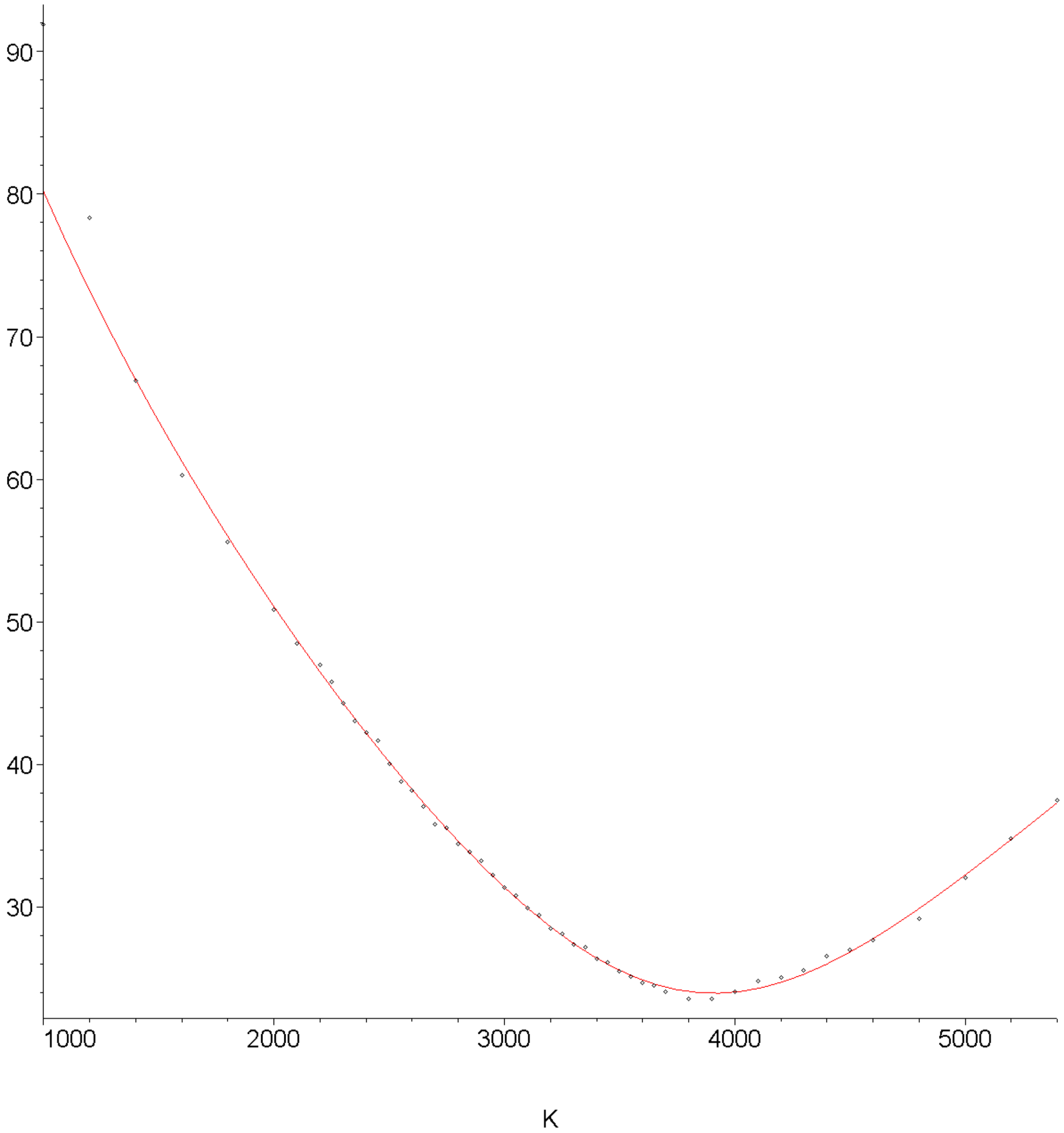
really fine ... visualize it:

```
> sqrt(gApprox(x)/Time): subs(x=ln(strike/Fwd),%):
  vol:=unapply( (%),strike);

  map2(op,1,originalData[2]): seq(%[i],i=1..nops(%)):
  lower:=min(%);
  upper:=max(%%);

  q1 := pointplot(originalData[2]):
  q2 := plot(100*vol(K), K=lower..upper):
  display(q1, q2);
  'vol(3300)': '%'=%;
```

$$\mathrm{vol} := \mathrm{strike} \to (-0.69417250074919 + 0.38671016610280 \ln(0.00030139723653829\ \mathrm{strike}) + 1.0483026113068\ (\ln(0.00030139723653829\ \mathrm{strike}))^2 - 0.81268269312714 \ln(0.00030139723653829\ \mathrm{strike}) + 0.53723827393110)^{(1/2)})^{(1/2)}$$

$$\mathrm{lower} := 1000.$$

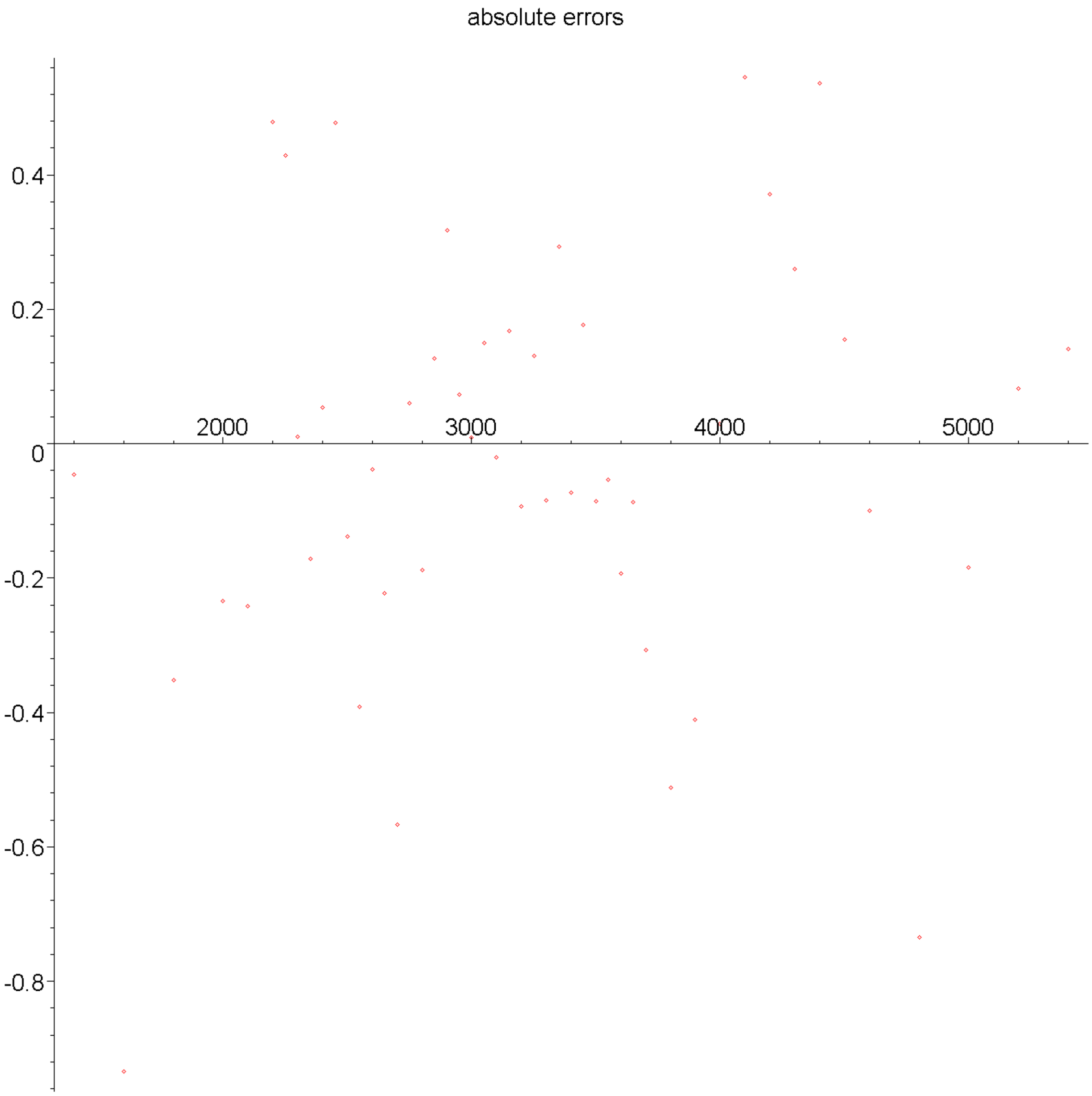
$$\mathrm{upper} := 5400.$$

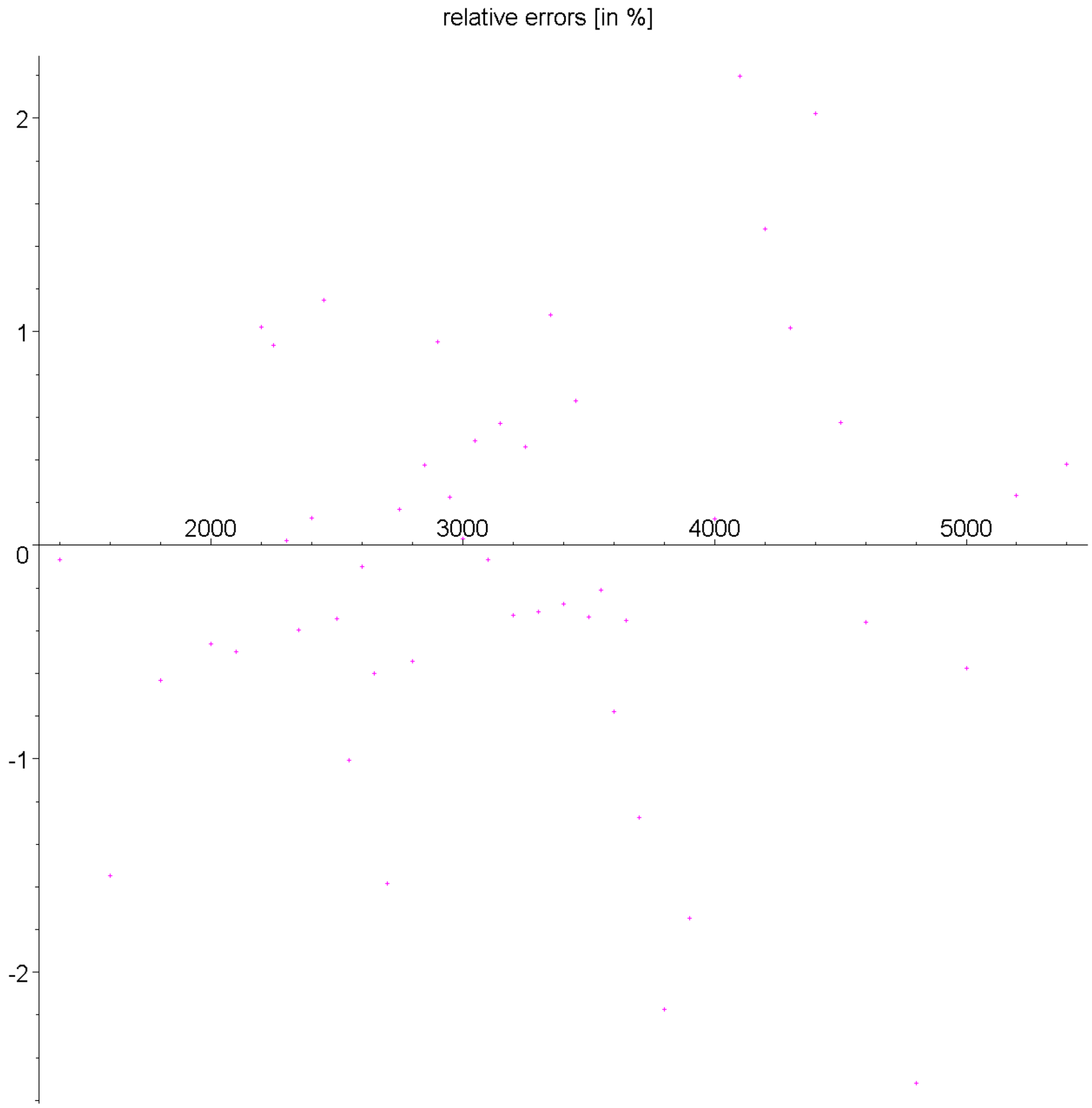


```
vol(3300) = 0.27433953486164
```

and check approximation errors:

```
> [seq( [ originalData[expiryNo][i][1],
    originalData[expiryNo][i][2] - 100*vol(originalData[expiryNo][i][1]) ]
  ,
    i=3..nops(originalData[expiryNo]) )]:
  plot(%, style=POINT,title=`absolute errors`);

  [seq( [ originalData[expiryNo][i][1],
    100*(originalData[expiryNo][i][2] -
  100*vol(originalData[expiryNo][i][1]))/
    originalData[expiryNo][i][2] ] ,
    i=3..nops(originalData[expiryNo]) )]:
  plot(%, style=POINT, symbol=CROSS, colour=magenta,
    title=`relative errors [in %]`);
```


absolute errors
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
2000
3000
4000
5000



This is really nice!

As it is a different least-square question it might be better
to fit volatility directly ...

Fitting Vol directly

So let us fit the smile, again filter for extreme strikes first (between 2000 and 5400)

```
> Data:=originalData[2]:
  myFilter:= proc(x) 2000<=x[1] and x[1]<=5400; end proc:
  select(myFilter,Data):
  [seq( [%[i][1], %[i][2]/100 ], i=1..nops(%))]:
  Data2fit:=%:
```

Re-write the variance function to take strikes as arguments and give volatility:

```
> `variance` = vola^2*time;
  `variance` = var;
  ``;
  rhs(%%):
```

```
'eval(%,k=ln(K/fwd))'= volatility^2*time;
```

$$\mathrm{variance} = \mathrm{vola}^2\ \mathrm{time}$$

$$\mathrm{variance} = a + b\left(\rho\,(k - m) + \sqrt{k^2 - 2\,k\,m + m^2 + \sigma^2}\,\right)$$

$$\left.\left(a + b\left(\rho\,(k - m) + \sqrt{k^2 - 2\,k\,m + m^2 + \sigma^2}\,\right)\right)\right|_{k = \ln\left(\frac{K}{fwd}\right)} = \mathrm{volatility}^2\ \mathrm{time}$$

so substitue and solve for volatility:

```
> subs(k=ln(K/Fwd),var): sqrt(abs(%)/Time):
  # take some care: absolute value before using square root ...
  P:=%;
```

$$P := 2.4942801849900\,\left|-a - b\left(\rho\,(\ln(0.00030139723653829\,K) - m) + \sqrt{\ln(0.00030139723653829\,K)^2 - 2\ln(0.00030139723653829\,K)\,m + m^2 + \sigma^2}\,\right)\right|^{(1/2)}$$

do the least square fit

```
> Residues := map((d) -> eval(P, K=d[1])-d[2], Data2fit):
  IP:=[a=0.0,b=0.2,sigma=0.4,rho=0.4,m=0];
  Sol := LSSolve(Residues,{0<=sigma},initialpoint=IP);
```

$$IP := [a = 0.,\ b = 0.2,\ \sigma = 0.4,\ \rho = 0.4,\ m = 0\,]$$

$$Sol := [\,0.000184370580384426688,\ [\,b = 0.160371124431923917,\ m = 0.381967445777821268,\ a = -0.0810178848939402119,\ \sigma = 0.599063412677262152,\ \rho = 0.343023011800208943\,]\,]$$

The arbitrage condition is ok:

```
> b*(1+abs(rho)) <= 4 / T;
  eval(%,Sol[2]):
  eval(%,T=Time):;
  is(%);
```

$$b\,(1 + |\rho|) \le \frac{4}{T}$$

$$\mathrm{true}$$

Now define the smile function (now one can omit the absolute sign):

```
> subs(k=ln(K/Fwd),var): sqrt( (%)/Time):
  eval(%,Sol[2]):
  Vol:=unapply(%,K);
```

$$\mathrm{Vol} := K \to \left(-0.63477468328129 + 0.34224719961237\ln(0.00030139723653829\,K) + 0.99773830862319\sqrt{\ln(0.00030139723653829\,K)^2 - 0.76393489155564\ln(0.00030139723653829\,K) + 0.50477610204256}\,\right)^{(1/2)}$$

Check it ATM ...

```
> expiryNo:=2;
  itemNo:=30;
  originalData[expiryNo][itemNo];
  Vol(3300);
```

$$\mathrm{expiryNo} := 2$$

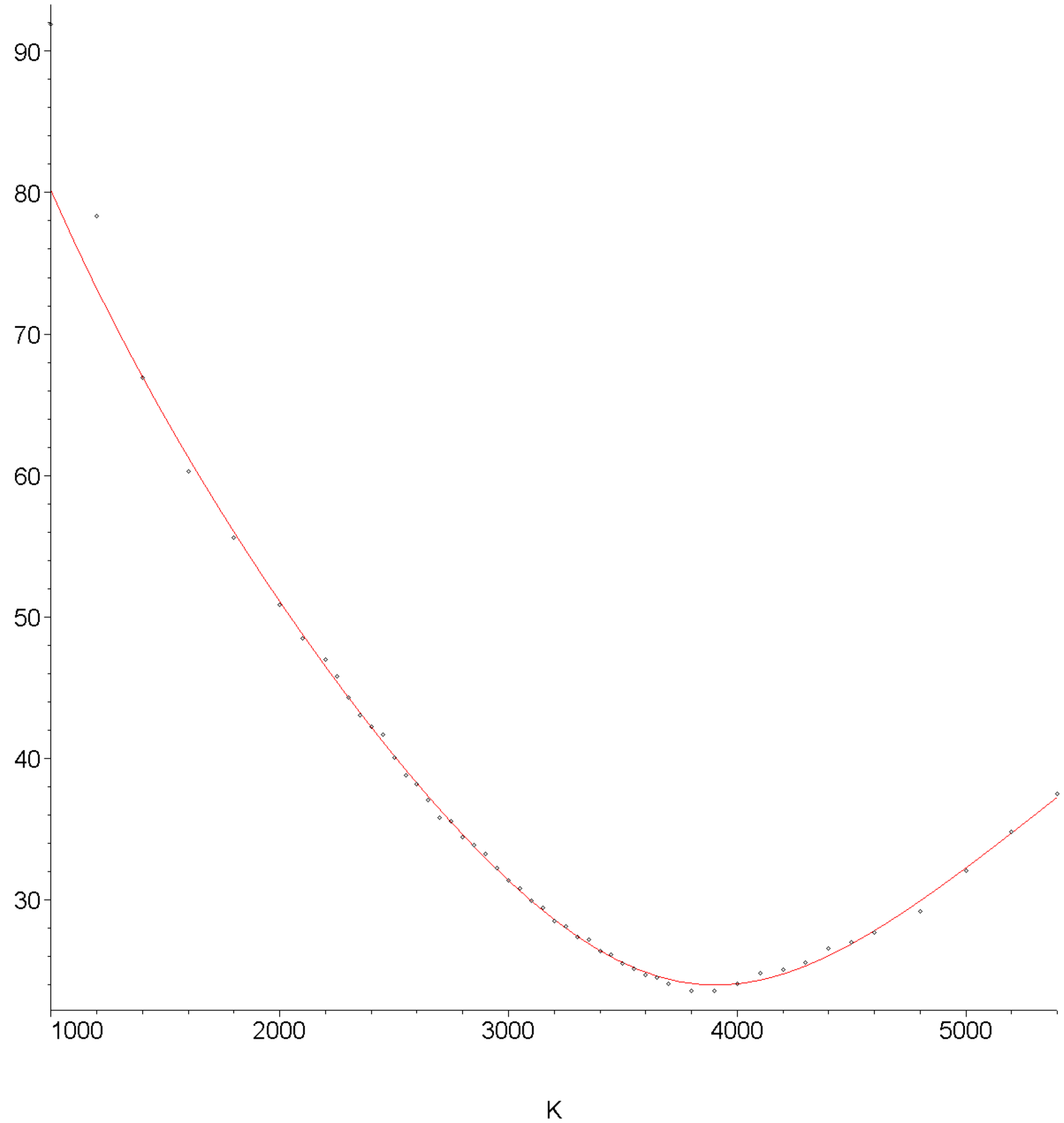
$$\mathrm{itemNo} := 30$$

$$[\,3300.,\ 27.3492621\,]$$

```
0.27414977627310
```

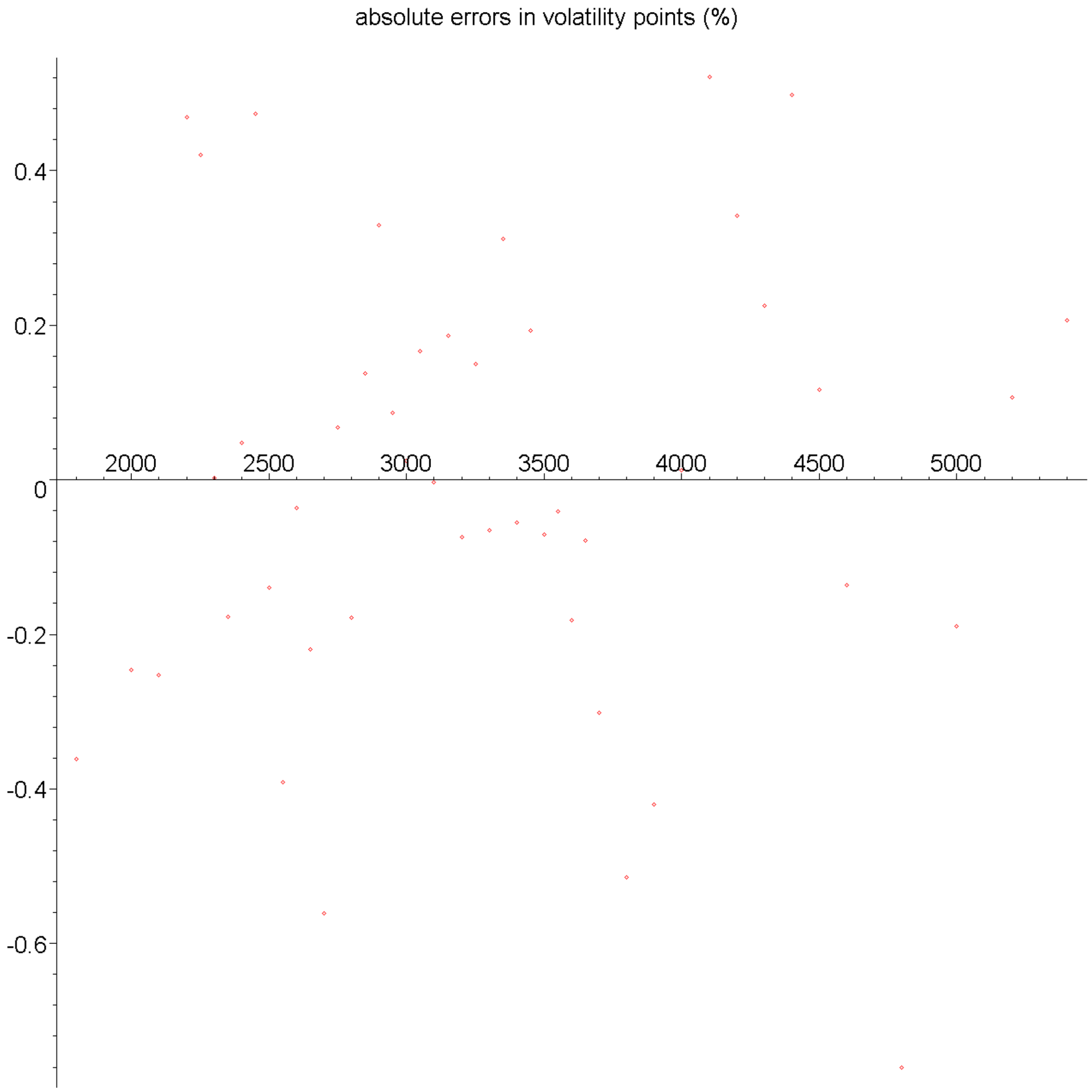
... and plot it against the data

```
> Q1 := pointplot(originalData[2]):
  Q2 := plot(Vol(K)*100, K=lower..upper):
  display(Q1, Q2);
```

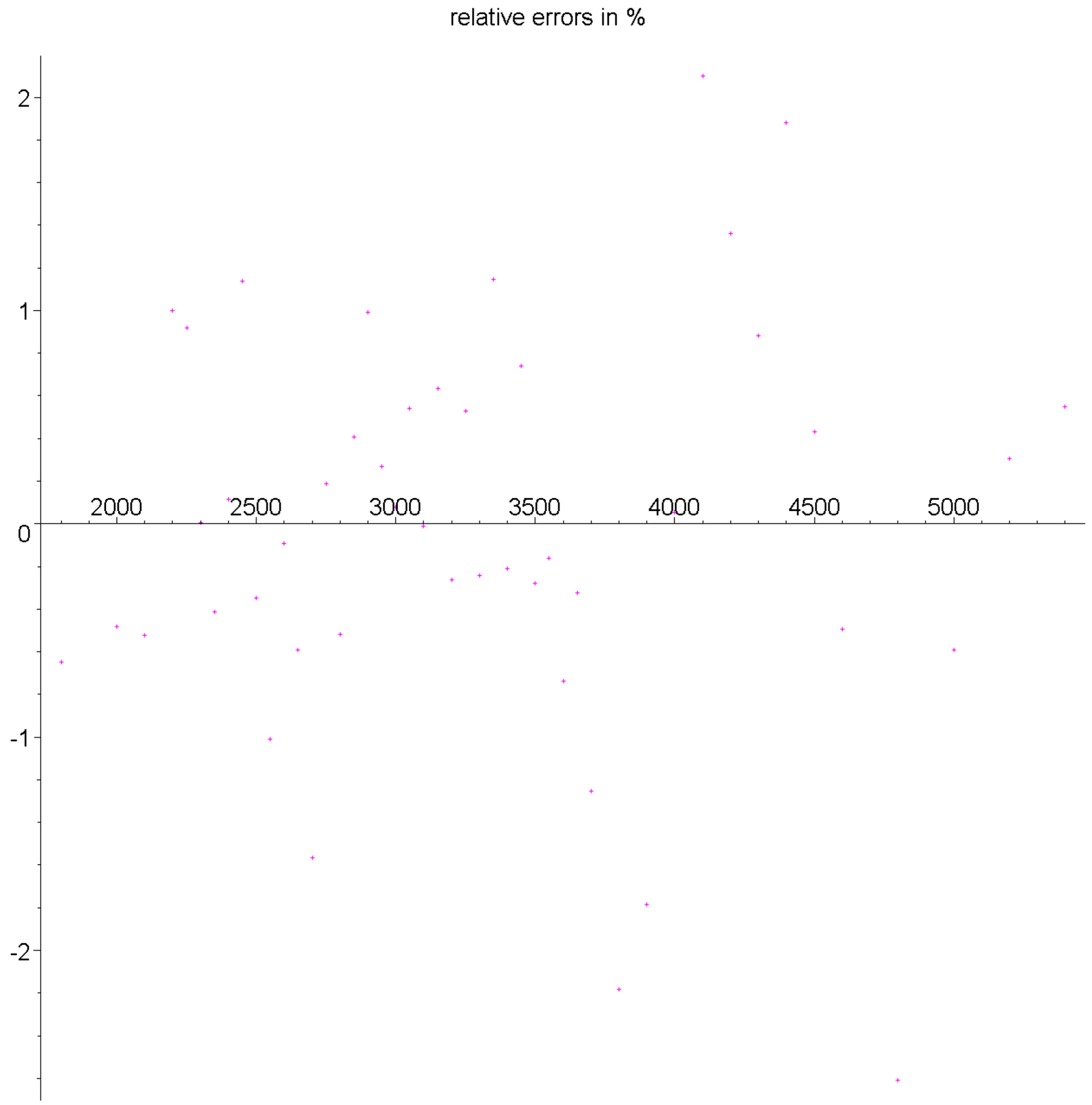


now check the errors ...

```
> [seq( [ Data[i][1],Data[i][2] - 100*Vol(Data[i][1]) ] ,i=5..nops(Data)
  )]:;
  #[seq( [ Data2[i][1],Data2[i][2] - Vol(Data2[i][1]) ] ,i=1..nops(Data2)
  )];
  plot(%, style=POINT, title=`absolute errors in volatility points (%)`);

  [seq( [ Data[i][1], 100*(Data[i][2] - 100*Vol(Data[i][1]))/Data[i][2] ] ,
    i=5..nops(Data) )]:;
  plot(%, style=POINT, symbol=CROSS, colour=magenta, title=`relative errors
  in %`);
```



absolute errors in volatility points (%)

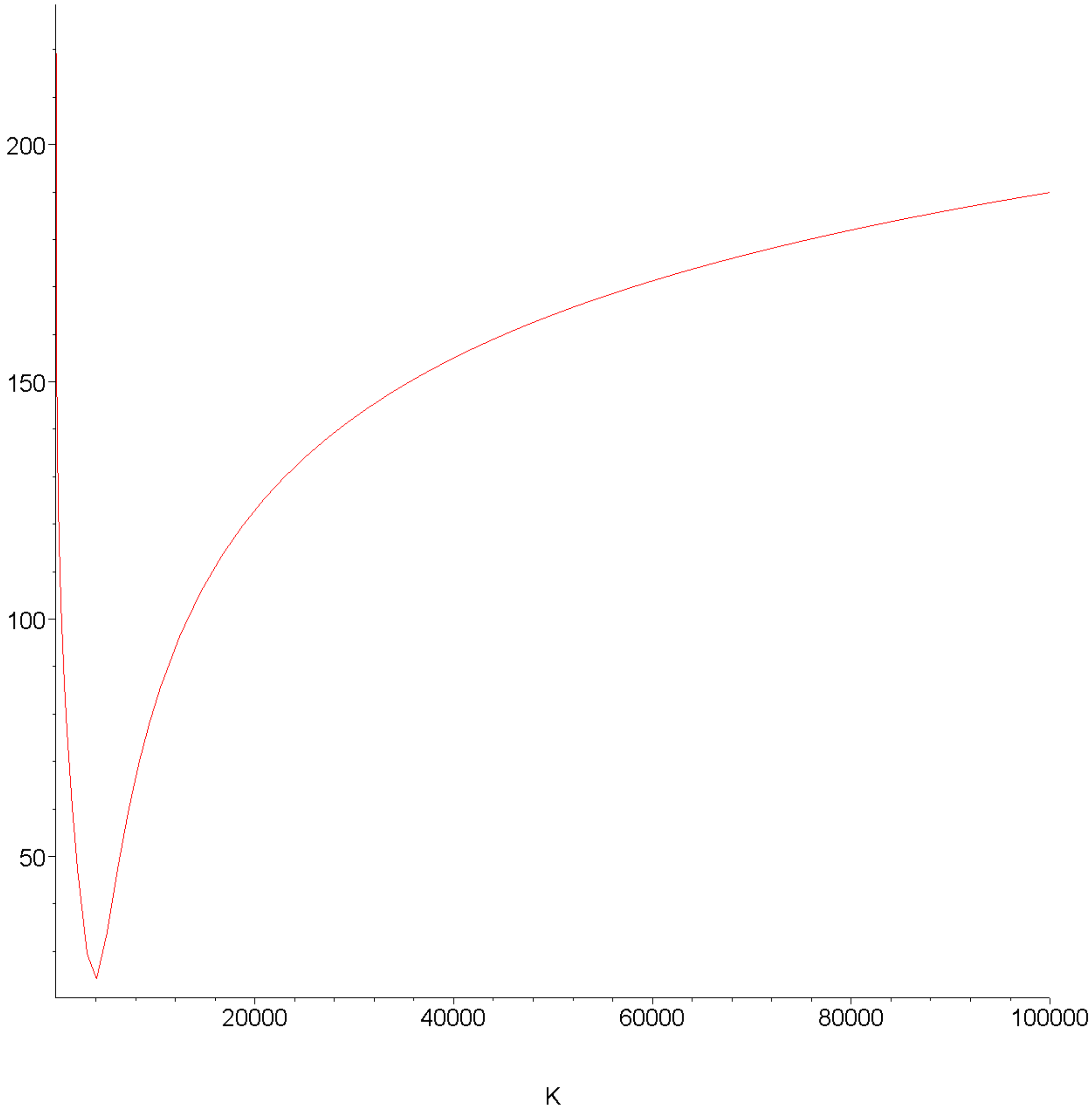


which makes me happy: errors are below half a vol point, and relative errors are also small (as vol level is high).

Some properties

Limiting behavior for the volatility:

```
> plot(Vol(K)*100, K=1..10^5);
  'limit(Vol(K),K=0,right)': '%'=%;
  'limit(Vol(K),K=infinity)': '%'=%;
  'limit(Vol(K)^2/ln(K),K=0,right)': '%'=%;
  'limit(Vol(K)^2/ln(K),K=infinity)': '%'=%;
```



$$\lim_{K \to 0+} \mathrm{Vol}(K) = \mathrm{Float}(\infty)$$

$$\lim_{K \to \infty} \mathrm{Vol}(K) = \mathrm{Float}(\infty)$$

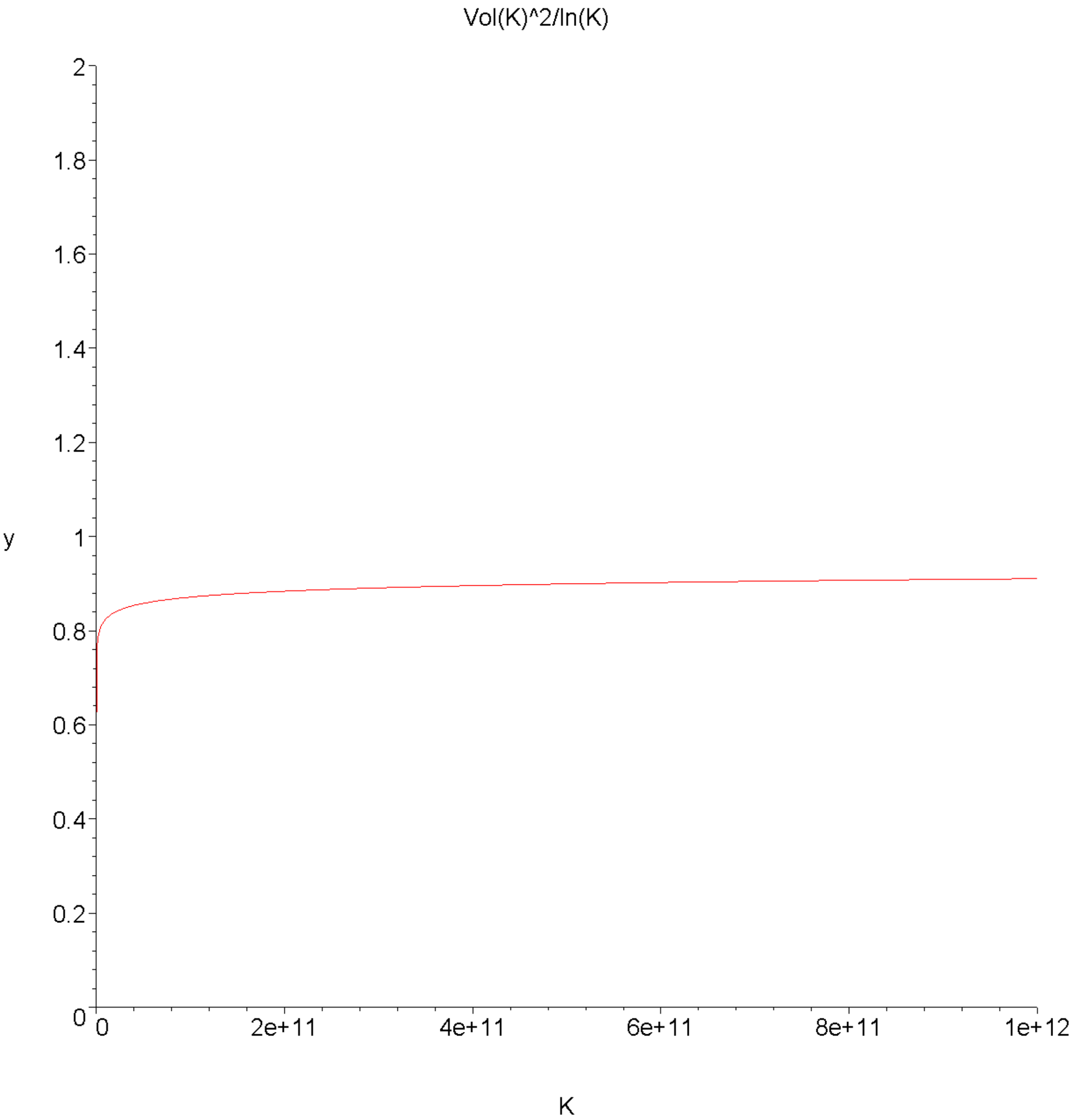
$$\lim_{K \to 0+} \frac{\mathrm{Vol}(K)^2}{\ln(K)} = -0.65549110901082$$

$$\lim_{K \to \infty} \frac{\mathrm{Vol}(K)^2}{\ln(K)} = 1.3399855082356$$

... which is ok, since impl variance should become linear over moneyness.

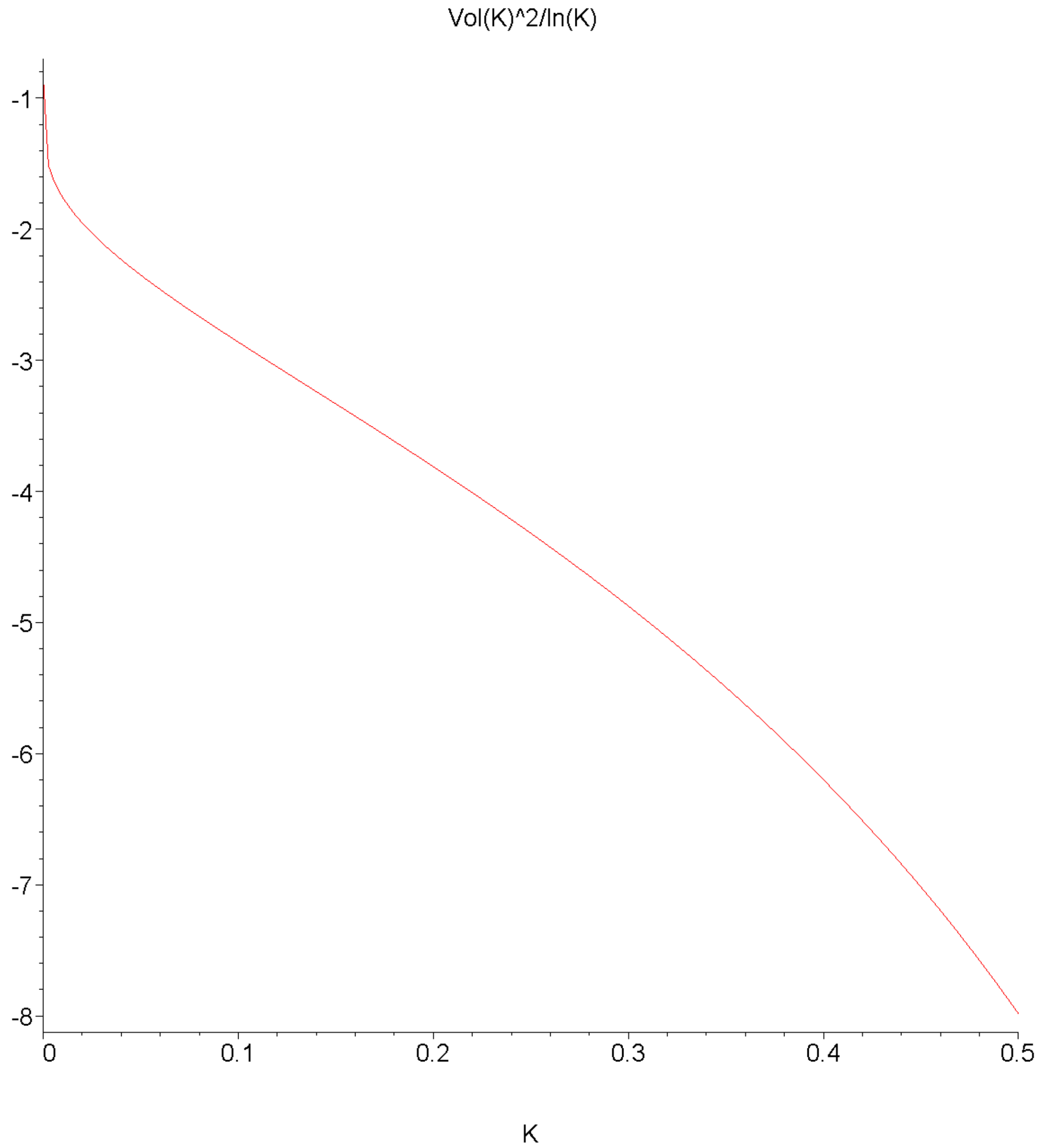
But that needs really extreme strikes:

```
> plot(Vol(K)^2/ln(K),K=100..10^12,y=0..2, title=`Vol(K)^2/ln(K)`);
  'eval(Vol(K)^2/ln(K),K=1e200)': '%'=%;
```



$$\left.\frac{\mathrm{Vol}(K)^2}{\ln(K)}\right|_{K=0.1\;10^{201}} = 1.3141909049179$$

```
> plot(Vol(K)^2/ln(K),K=10^(-12)..0.5, title=`Vol(K)^2/ln(K)`);
  'eval(Vol(K)^2/ln(K),K=1e-200)': '%'=%;
```



$$\left.\frac{\mathrm{Vol}(K)^2}{\ln(K)}\right|_{K = 0.1\ 10^{-199}} = -0.66648056259794$$

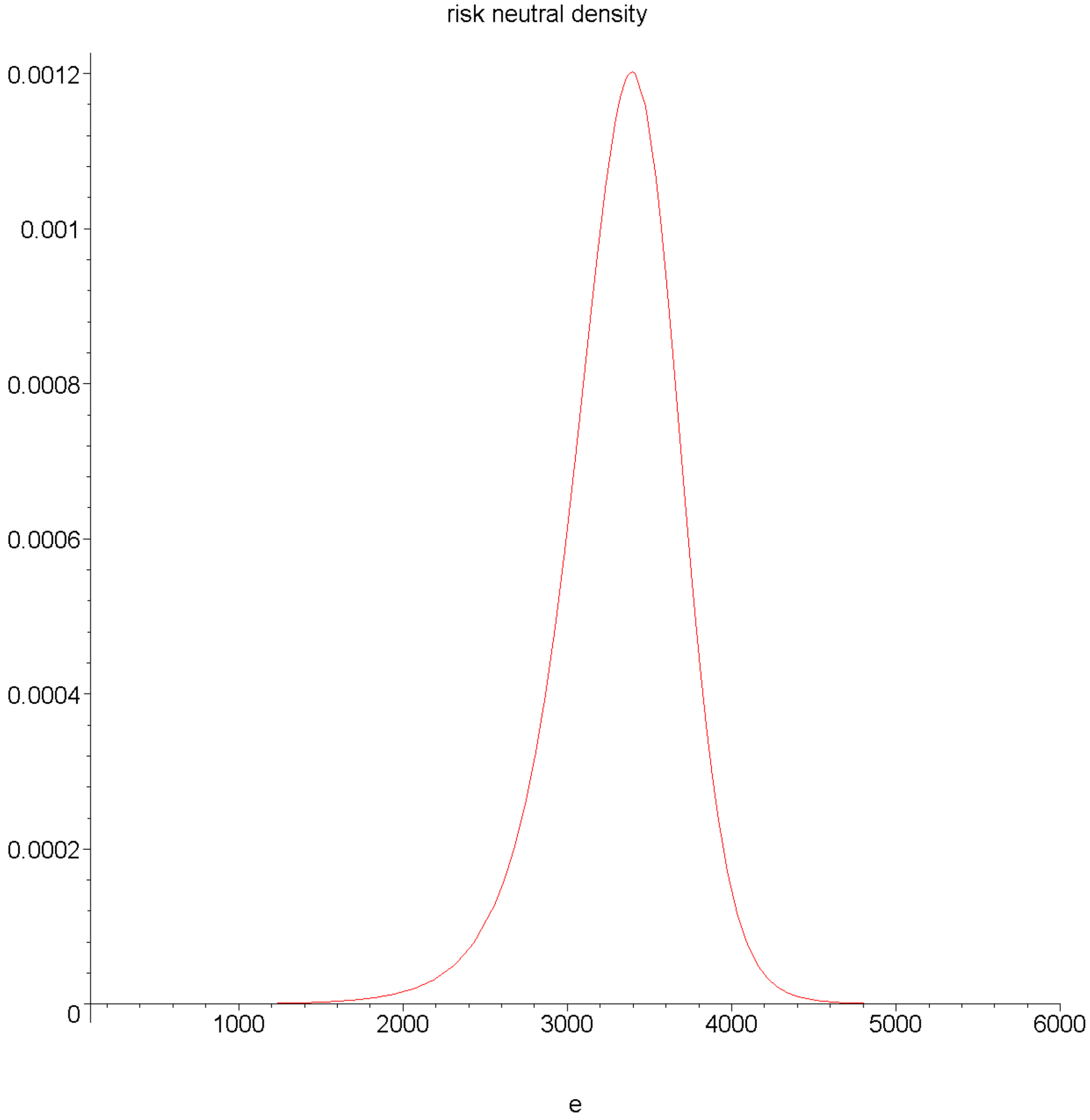
Let us look at the implied RND:

```
> with(avtbslib):
> #
  keyNr ExpiryDate Date time Dax Future Forward rate SeriesCounted downExcer
   upExcer
  #  1     2          3    4    5   6      7       8    9              10
  11
  marketData:=[s=inParam[2][5],t=inParam[2][4]/365.25,r=inParam[2][8]/100];
          marketData := [ s = 3306.516184, t = 0.16073465661875, r = 0.021331368710000 ]
> `'RND'` = 'diff(exp(r*t)*BSCall(s,strike,t,r,Vol(strike)) ,strike$2)';
  remDigits:=Digits: Digits:=remDigits+6:
  exp(r*t)*BSCall(s,e,t,r,Vol(e)): #eval(%,marketData): # indets(%,atomic):
  indets(%);
  diff(%,e$2): combine(%,[exp,ln]) assuming 0<e: combine(%,power):
  eval(%,marketData): #simplify(%,symbolic);
  RND:=evalf(%):  # length(simplify(rnd,symbolic));
  Digits:=remDigits:
```

$$\mathrm{RND} = \frac{\partial^2}{\partial \mathrm{strike}^2}\left(\mathbf{e}^{(r\,t)}\,\mathrm{BSCall}(s, \mathrm{strike}, t, r, \mathrm{Vol}(\mathrm{strike}))\right)$$

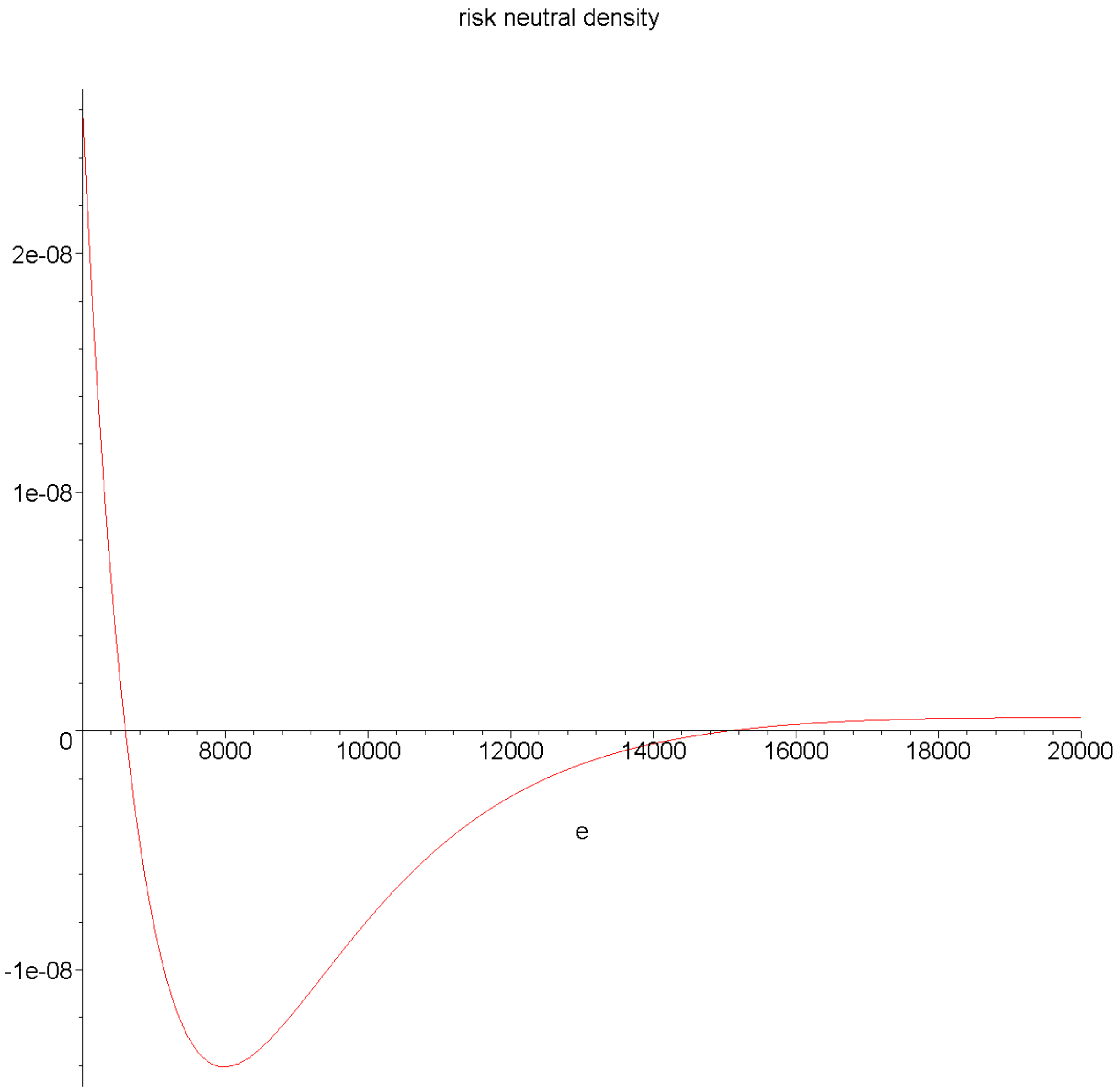
the expression for the RND is somewhat lenghty, so only plot it:

```
> plot(RND,e=100..6000, title=`risk neutral density`);
```



looking closer: there is a 'bad' behaviour

```
> plot(RND,e=6000..20000, title=`risk neutral density`);
  fsolve(RND=0,e=6000..8000);   #evalf(subs(e=%,rnd),24);
  fsolve(RND=0,e=14000..18000); #evalf(subs(e=%,rnd),24);
```



$$6597.5488483589$$

$$15040.579001634$$

i have neither checked nor located the numerical error for that (RND becomes negative, but is complicated), the downside is ok ... and it integrates to 1:

```
> Int('RND',e=0..infinity):
  '%'= evalf(%);
```

$$\int_0^{\infty} \mathrm{RND}\, de = 0.99999999999999$$