```
> #Digits:=16:
> with(avtbslib):
  #assume(0<s):assume(0<e):assume(0<t):assume(0<v): #assume(0<r):
  #with(RealDomain):
  #assume(0<=d): # dividend
```

Collector's data with a short hand to evaluate for them

```
> valCollector :=
    proc(x,td) evalf(eval(x,[S=100.0, K=100, t=1.0, r=0.06, v=0.30, d=7.0,
  tau=td])) end proc;
```

valCollector :=

proc(x, td) evalf(eval(x, [S = 100.0, K = 100, t = 1.0, r = 0.06, v = 0.30, d = 7.0, τ = td])) **end proc**

Alan Lewis suggests the following solution

```
> # the usual transition density
  p:= (S,K,t,r,v) -> diffN(dTwo(S,K,t,r,v))/K/v/sqrt(t): p(s,e,t,r,v);
```

$$\frac{1}{2}\,\frac{\mathbf{e}^{\left(-1/2\left(\frac{\ln\left(\frac{s}{e}\right)+r\,t}{v\sqrt{t}}-\frac{v\sqrt{t}}{2}\right)^2\right)}\sqrt{2}}{\sqrt{\pi}\,e\,v\sqrt{t}}$$

```
> # Alan's solution
  kern:= p(S,S1,tau,r,v)*BSCall(S1-d,K,t-tau,r,v):
  call:='exp(-r*tau)*int(p(S,S1,tau,r,v)*BSCall(S1-d,K,t-tau,r,v),S1=d..infinit
  y)';
```

$$\text{call} := \mathbf{e}^{(-r\,\tau)}\int_{d}^{\infty} \mathrm{p}(S, S1, \tau, r, v)\,\mathrm{BSCall}(S1-d, K, t-\tau, r, v)\,dS1$$

Let us test it. First look what happens, if the dividend is payed in almost 1 year:

```
> valCollector(call,0.99);
```

$$11.57961537$$

This is (up to digits) the result given by Alan Lewis.

Now the case, when the dividend is coming up soon:

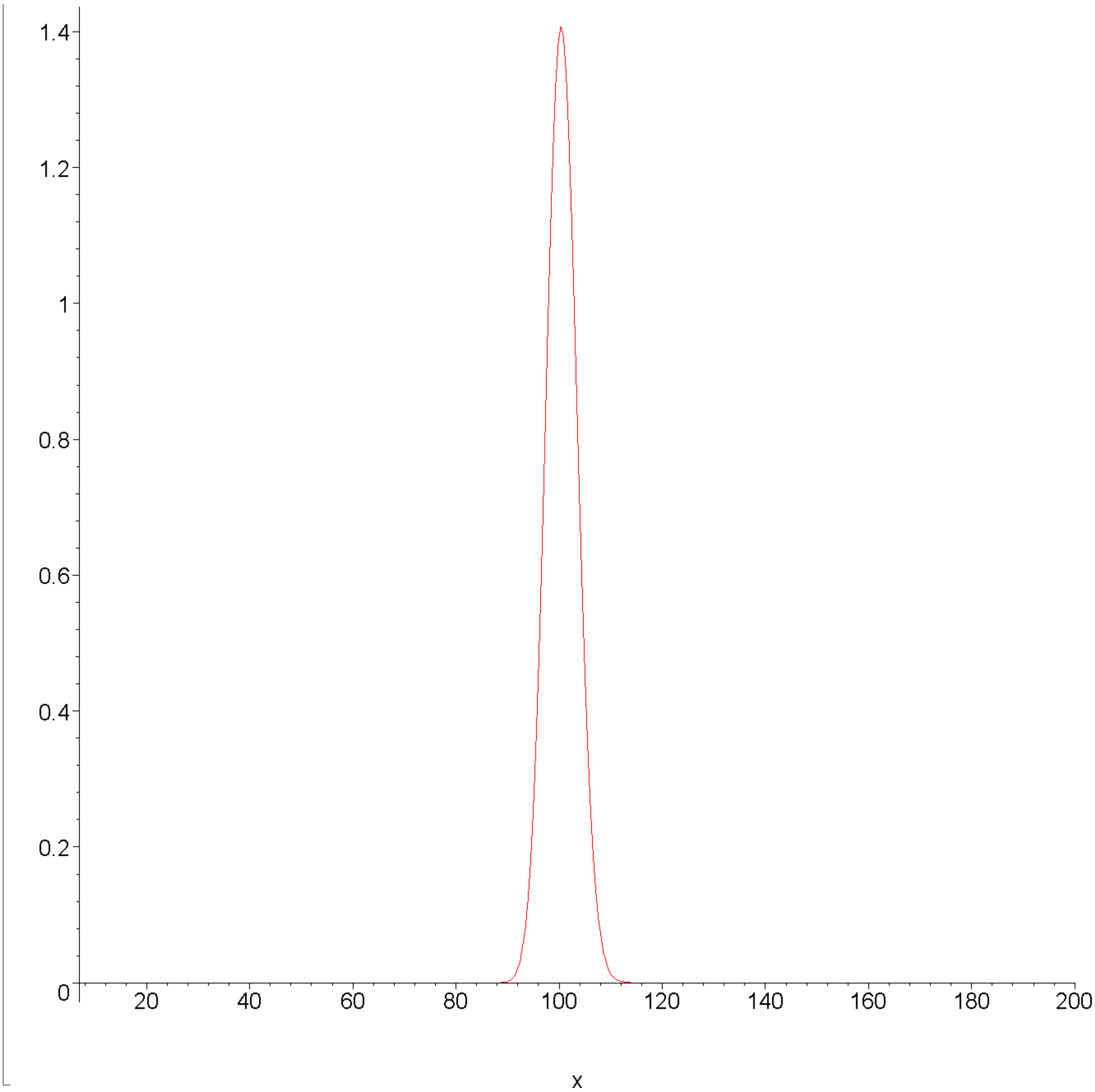
```
> valCollector(call,0.01);
```

$$0.3119367191\ 10^{-43}$$

This is a crazy result.
And Alan adviced me to integrate more carefully since the program might have problems with the peak if S and S1 are close:

```
> kernNum:=unapply(valCollector(kern,0.01), S1):
  'limit(kernNum(x),x=0)'=limit(kernNum(x),x=0);
  plot(kernNum(x),x=7..200);
```

$$\lim_{x \to 0} \mathrm{kernNum}(x) = 0.$$



So more carefully: integrate around the peak and handle the tails extra

```
> # determine the peak: here it should be around the peak of p, ie for dTwo =
  0
  xp:=solve(dTwo(S,S1,t,r,v)=0,S1); #diff(kernNum(x),x):
  xp:=fsolve(%,x=100..2*100);
```

$$xp := \frac{S}{e^{(-r\,t + 1/2\,v^2\,t)}}$$

```
> xp:=valCollector(xp,0.01);
  yp:=kernNum(xp);
```

$$xp := 101.5113065$$

$$yp := 1.313820973$$

```
> x0:=fsolve(kernNum(x)=yp*2^(-16),x,0..xp);
  x1:=solve(ln(x/xp)=-ln(x0/xp),x); # x1 := xp^2/x0
```

$$x0 := 87.16766455$$

$$x1 := 118.2152281$$

If S1 becomes very large the Call with strike S1 in the kernel goes to 0 and
the probability as well: it contributes almost nothing
to the integral, the tail can be cut off. Find a compromise to do it soon, but

not at cost of exactness. I am to lazy for estimating,
Alan suggests:

```
> S*exp(n*v*sqrt(t)): eval(%, n= 16): valCollector(%,0.01):
  smax:=%;
```

$$smax := 12151.04175$$

and indeed

```
> int(kernNum(S1),S1=smax..smax+(2^16)^2);
```

$$0.$$

Now do the example again

```
> exp(-r*tau)*
   (evalf(Int(kernNum(z),z=7..x0))
   + evalf(Int(kernNum(z),z=x0..x1))
   + evalf(Int(kernNum(z),z=x1..smax))):
  valCollector(%,0.01);
```

$$10.59143844$$

which is ok: i did work with 10 digits of exactness, using 16 i get 10.59143873835987 and it needs some time.

```
>
```