```
> restart;
> Digits:=14;
```

$$Digits := 14$$

MCC Madan, Carr, Chang 1998

```
> 'N(a/sqrt(u)+b*sqrt(u))*u^(c-1)*exp(-u)';
  Int(%,u=0..infinity)/GAMMA(c): #value(%):
  M:= unapply(%,a,b,c); #assume(0<c): assume(a::real): assume(b::real):
```

$$N\left(\frac{a}{\sqrt{u}}+b\sqrt{u}\right)u^{(c-1)}\,\mathbf{e}^{(-u)}$$

$$M := (a, b, c) \to \frac{1}{\Gamma(c)}\int_0^\infty N\left(\frac{a}{\sqrt{u}}+b\sqrt{u}\right)u^{(c-1)}\,\mathbf{e}^{(-u)}\,du$$

that function is called $\Psi(a, b, \gamma)$ in the paper (appendix).

```
> N:= x -> 1/2+1/2*erf(1/2*x*2^(1/2));
```

$$N := x \to \frac{1}{2}+\frac{1}{2}\operatorname{erf}\left(\frac{1}{2}x\sqrt{2}\right)$$

```
> #assume(0<sigma): assume(nu::real): assume(theta::real):

  zeta:=-theta/sigma^2;
  s:='sigma/sqrt(1+(theta/sigma)^2*nu/2)';

  alpha:='zeta*s';
  c1:='nu*(alpha+s)^2/2';
  c2:='nu*alpha^2/2';

  d0:='1/s*(ln(S/K)+r*t+t/nu*ln((1-c1)/(1-c2)))';
```

$$\zeta := -\frac{\theta}{\sigma^2}$$

$$s := \frac{\sigma}{\sqrt{1+\frac{\theta^2\,\nu}{2\,\sigma^2}}}$$

$$\alpha := \zeta\, s$$

$$c1 := \frac{\nu\,(\alpha+s)^2}{2}$$

$$c2 := \frac{\nu\,\alpha^2}{2}$$

$$d0 := \frac{\ln\left(\frac{S}{K}\right)+r\,t+\frac{t\ln\left(\frac{1-c1}{1-c2}\right)}{\nu}}{s}$$

```
> 'S*M(d0*sqrt((1-c1)/nu),(alpha+s)*sqrt(nu/(1-c1)),t/nu) -
    K*exp(-r*t)*M(d0*sqrt((1-c2)/nu),(alpha+0)*sqrt(nu/(1-c2)),t/nu)':

  MCC:='unapply(%,S,K,t,r,sigma,nu,theta)'; # indets(MCC(S,K,t,r)):
  indets(%,atomic): indets(%);
  assume(0<S); assume(0<K); assume(0<t); assume(0<=r);
```

MCC := unapply(

$$S\,M\left(d0\sqrt{\frac{1-c1}{\nu}},(\alpha+s)\sqrt{\frac{\nu}{1-c1}},\frac{t}{\nu}\right)-K\,e^{(-r\,t)}\,M\left(d0\sqrt{\frac{1-c2}{\nu}},\alpha\sqrt{\frac{\nu}{1-c2}},\frac{t}{\nu}\right),S,K,t,r,\sigma,\nu,\theta\Big)$$

that function is called c(S(0) ; K , t) in the paper

Examples

```
> tstData:=[S=100.0, t=1.0 ,r=0.02,sigma=0.1737,nu=0.679,theta=0.0492];
```

$$tstData := [S = 100.0, t = 1.0, r = 0.02, \sigma = 0.1737, \nu = 0.679, \theta = 0.0492]$$

let the Call depend only on K (to have 'good' run times)

```
> MCC(S,K,t,r,sigma,nu,theta): eval(%,tstData):

  VGC:=unapply(%,K):;

> VGC( 90.0): evalf(%);
  VGC(100.0): evalf(%);
  VGC(110.0): evalf(%);
```

14.123021770626

7.477234761060

3.408571065263

now do it the reasonable way

```
> params:=[sigma=0.1737,nu=0.679,theta=0.0492];
```

$$params := [\sigma = 0.1737, \nu = 0.679, \theta = 0.0492]$$

```
> MCC(S,K,t,r,sigma,nu,theta):
  subs(S=spot,K=strike,t=time,r=rates,%):
  eval(%,params): evalf(%):

  VGCall:=unapply(%,spot,strike,time,rates):

> for K in [90.0, 100.0, 110.0] do
    price[K]=evalf(VGCall(100.0, K, 1.0, 0.02));
  end do;
```

$$price_{90.0} = 14.123021770623$$

$$price_{100.0} = 7.477234761060$$

$$price_{110.0} = 3.408571065264$$

check runtime:

```
> st:=time():
  noSteps:=100;
  for iStep from 0 to noSteps do
    evalf(VGCall(100.0, 100.0 -noSteps/2 + iStep  ,0.5,0.00));
  end do:
  `average time in seconds`=evalf[3]((time()-st)/(noSteps+1));
```

noSteps := 100

average time in seconds = 0.0189

a pure C program may faster by a factor of 10 - 100 ...