MC-Simulation for pathes in Heston's stochastic volatility model

References:

S. L. Heston, A closed-form solution for options with stochastic volatility ..., Review of Financial Studies 6, 327 (1993)

For applications on timeseries (instead of option prices) see: Yakovenko et al (2002),
Comparison between the probability distribution of returns in the Heston model and
empirical data for stock indexes at http://lanl.arXiv.org/abs/cond-mat/0211050
http://www2.physics.umd.edu/~yakovenk/personal_data/publications.html

Maple seems to be a little bit slow for that, but i always wanted to have it in a worksheet.
Have fun with it!
Axel

```
> restart; with(stats): # UseHardwareFloats := true:
```

The model is formulated in terms of stochastic differential equations, the SDEs allow correlation between spot returns and volatility (both are Brownian motions).

I do not use it, but as different notations are in use for the parameters, here they are:

```
> 'dS[t]=mu*S[t]*dt + sqrt(v[t])*S[t]*dB[t]';
  'dv[t]=kappa*(theta-v[t])*dt+sigma*sqrt(v[t])*dZ[t]';
  'dZ[t]=rho*dB[t]+sqrt(1-rho^2)*dW[t]'; 'cov(dB[t],dZ[t])=rho*dt':
```

$$dS_t = \mu\, S_t\, dt + \sqrt{v_t}\, S_t\, dB_t$$

$$dv_t = \kappa\,(\theta - v_t)\, dt + \sigma \sqrt{v_t}\, dZ_t$$

$$dZ_t = \rho\, dB_t + \sqrt{1-\rho^2}\, dW_t$$

Take a discretization for that

```
> 'S[k+1]'='S[k] + mu*S[k]*dt + sqrt(v[k]*dt) *S[k]* B[k]';
  'v[k+1]'='v[k]+kappa*(theta-v[k])*dt+sigma*sqrt(v[k]*dt)*Z[k]';
  'Z[k]'='rho*B[k]+sqrt(1-rho^2)*W[k]';
```

$$S_{k+1} = S_k + \mu\, S_k\, dt + \sqrt{v_k\, dt}\, S_k\, B_k$$

$$v_{k+1} = v_k + \kappa\,(\theta - v_k)\, dt + \sigma \sqrt{v_k\, dt}\, Z_k$$

$$Z_k = \rho\, B_k + \sqrt{1-\rho^2}\, W_k$$

Now choose model parameters and starting values ...
(for real markets one would need other parameter values)

```
> mu:=   0.025; # drift = rates
  sigma:=0.40;  # vol of vol
  kappa:=2.00;  # mean reversion
  theta:=0.04;  # long run variance
  rho:= -0.50;  # correlation
  #mu:=0.025; sigma:=0.6; kappa:=0.24; theta:=0.02;rho:= -0.64;
  ``;
  S0:=100.0;    # start values
  V0:= 0.04;    # instant vol
```

$$\mu := 0.025$$

$$\sigma := 0.40$$

$$\kappa := 2.00$$

$$\theta := 0.04$$

$$\rho := -0.50$$

$$S0 := 100.0$$

$$V0 := 0.04$$

... and let it run:

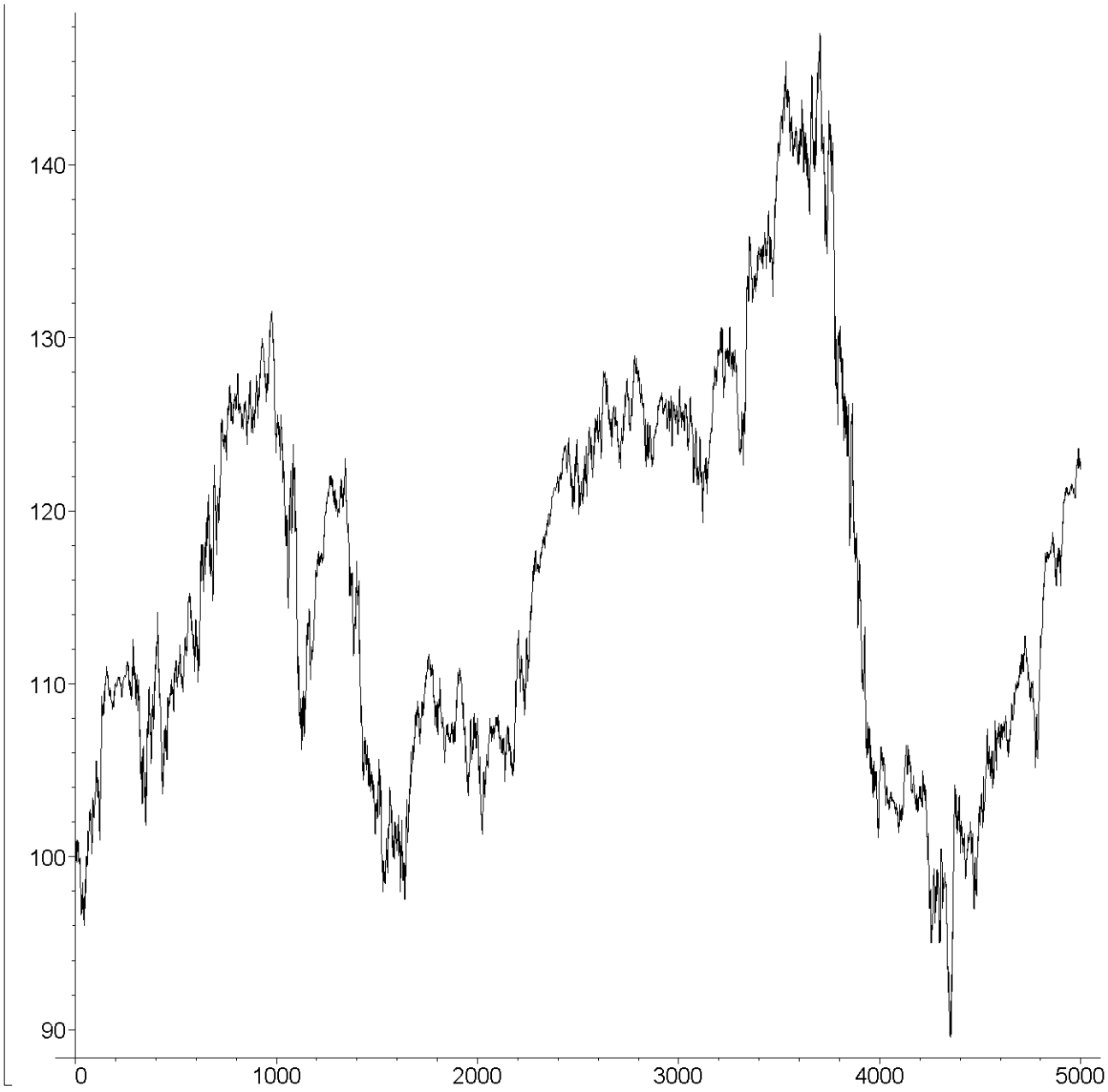
```
> N:=5000:      # number of steps
  dt:=1/252.0:  # time step = step size

  st:=time():
  S:=array(1..N):
  S[1]:=S0:
  v_t:=V0:
  randomize():
  B:=[stats[random,normald[0,1]](N+1)]:
  W:=[stats[random,normald[0,1]](N+1)]:
  Z:= zip((x,y) -> evalf( rho*x+sqrt(1-rho^2)*y ),B,W):
  `runtime random generation seconds`=time()-st;
  for k from 1 to N-1 do
    S[k+1]:=evalf(S[k] + mu*S[k]*dt + sigma*sqrt(v_t*dt) *S[k]*B[k]):
    v_t:=abs(evalf(v_t+kappa*(theta-v_t)*dt+sigma*sqrt(v_t*dt)*Z[k]));
  end do: k:='k':
  `runtime total seconds`=time()-st; # 15 - 25 seconds
  ``;
  `years`=evalf[3](N*dt);
  plots[listplot](S);
```

runtime random generation seconds = 4.418

runtime total seconds = 6.120

years = 19.8



compare it with riskless rates

```
> `bonds`='S0*(1+mu*dt)^N'; %: evalf[3](%); `final spot`=evalf[3](S[N]); ``;
```

$$bonds = S0\,(1+\mu\,dt)^{N}$$

bonds = 164.

final spot = 122.

statistics for the logarithmic returns:
skewed (not equal 0) with excess kurtosis (greater than 3) and a 'clustering' in the graph

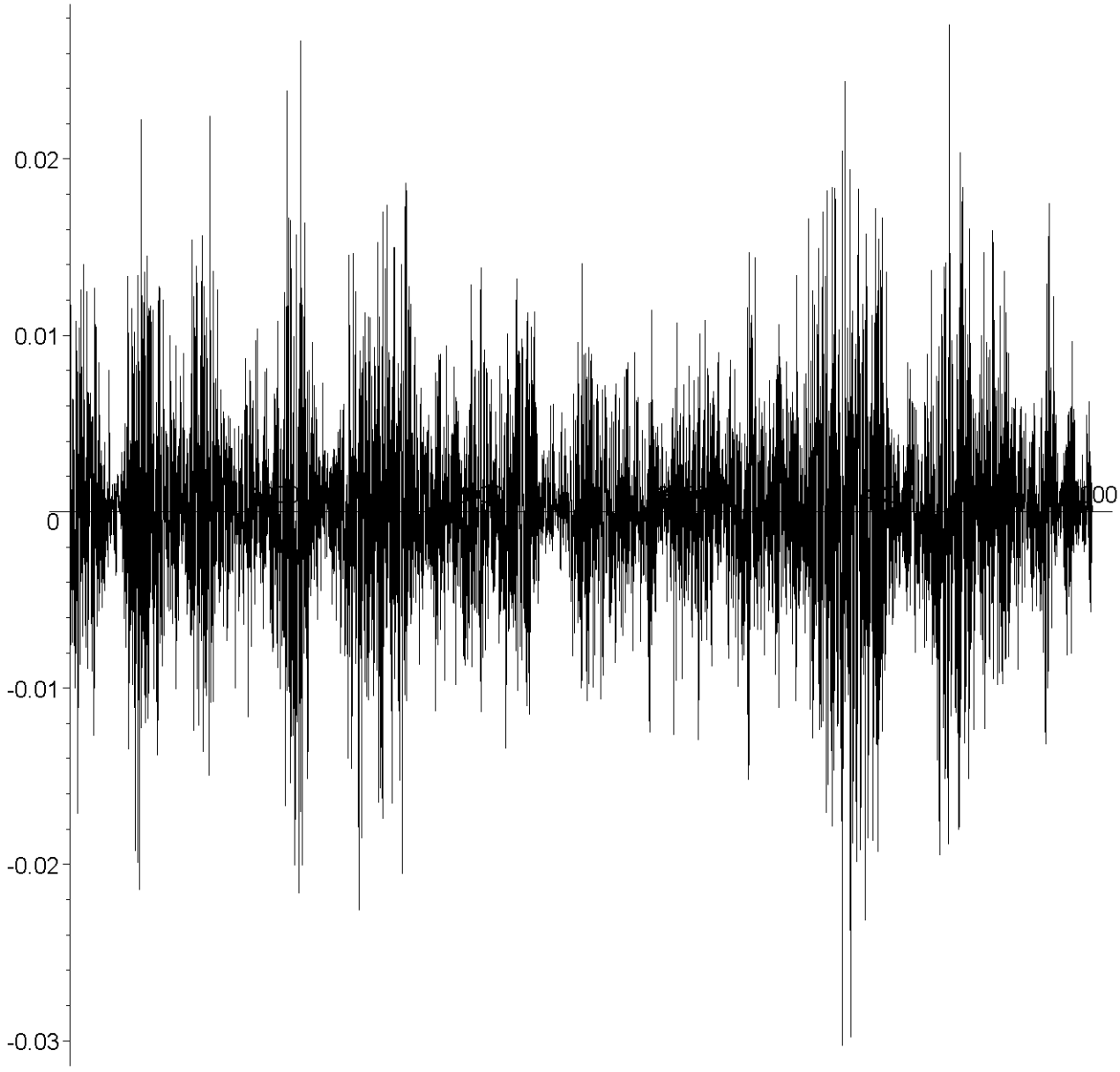
```
> returns:=[seq(evalf(ln(S[k]/S[k-1])),k=2..N)]: k:='k':
  `mean`=describe[mean](returns);
  `variance`=describe[variance](returns); #
  `vola`=sqrt(describe[variance](returns));
  `skewness`=describe[skewness](returns);
  `kurtosis`=describe[kurtosis](returns);
  plots[listplot](returns);
```

mean = 0.00004046391882

variance = 0.00002810221470

skewness = -0.05332529307

kurtosis = 5.383420543



>

Variant: Box-Muller

```
> N:=5000:      # number of steps
  dt:=1/252.0:  # time step = step size

  st:=time():
  S:=array(1..N):
  S[1]:=S0:
  v_t:=V0:

  # take 2 variates uniform and independently distributed in (0,1):
  randomize(): U1:=[stats[random,uniform](N+1)]:
  randomize(): U2:=[stats[random,uniform](N+1)]:
  # use Box-Muller to get a 2 normal variates (mean=0, variance=1)
  biv:=zip( (u1,u2) ->
    evalf([sqrt(-2*ln(u1))*cos(2*Pi*u2),sqrt(-2*ln(u1))*sin(2*Pi*u2)]) ,
  U1,U2):
  #bivcor:=map( x->[x[1],rho*x[1]+sqrt(1-rho^2)*x[2]],biv):
```

```
# now correlated them as above
B:=map(x -> x[1],biv):
Z:=map(x -> evalf(rho*x[1]+sqrt(1-rho^2)*x[2]), biv):
`runtime random generation seconds`=time()-st;
for k from 1 to N-1 do
  S[k+1]:=evalf(S[k] + mu*S[k]*dt + sigma*sqrt(v_t*dt) *S[k]*B[k]):
  v_t:=abs(evalf(v_t+kappa*(theta-v_t)*dt+sigma*sqrt(v_t*dt)*Z[k])):
end do: k:='k':
`runtime seconds`=time()-st; # 15 - 25 seconds
``;
`years`=evalf[3](N*dt);
plots[listplot](S);
```

runtime random generation seconds = 17.130

runtime seconds = 19.185

years = 19.8

[>