```
% Take Home Exam 3 - Part F
% Anita Mezzetti

clc
clear all
close all

% initial conditions:
X0 = 0;
V0 = 0.04;

% parameters:
kappa = 0.5;
theta = 0.04;
sigma = 1;
r = 0;
rho = -0.5;
T = 1/12;
v_min = 1e-4;
v_max = 0.08;
k = -0.1;

% weighting gaussian distribution parameters
% GenJacobiCanonic function definded below
G = Generator_Canonic(kappa, theta, sigma, r, rho, v_min, v_max);
M = [1 V0 X0 V0^2 V0*X0 X0^2] * expm(T * G);

% mean and standard deviation:
mu_w = M(3);
sigma_w = sqrt(M(6) - M(3)^2);

% polynomial expansion price
N_pol = 50;          % approximation degree
tic
price_polynomial = PriceApprox(N_pol, V0, X0, kappa, sigma, theta, r,
 rho, ...
    T, v_min, v_max, mu_w, sigma_w, k);
fprintf('Polynomial expansion price with N_pol=%d: \n', N_pol)
disp(price_polynomial)
time_polynomial = toc;

% MC parameters:
NumSim = 1e8;
NT = 100;

% MC price:
tic
X = SimSDEJacobi( V0, X0, kappa, theta, sigma, r, rho, ...
    T, v_min, v_max, NumSim, NT);
price_MC = exp(-r*T)*mean(max(0,exp(X)-exp(k)));
fprintf('Monte Carlo simulation price with NumSim=%d: \n:', NumSim)
disp(price_MC)
```

```
time_mc = toc;

% absolute error
error = abs(price_polynomial - price_MC);
disp('Absolute Error between polinomial al Monte Carlo price:')
disp(error)

% time spent
fprintf('Time for polynomail : ')
disp(time_polynomial)
fprintf('Time for Monte Carlo: ')
disp(time_mc)

% Functions:

function [G] = Generator_Canonic(kappa, theta, sigma, r, rho, v_min,
 v_max)
% Build the matrix representation of the Jacobi model
%   generator with canonic polynomial basis up to second degree

% matrix initialization
G = zeros(6,6);

% vector Q
D = (sqrt(v_max)-sqrt(v_min))^(-2);
Q = D*[-v_max*v_min; v_max+v_min; 0; -1; 0; 0];

% generator matrix
G(1:2,2) = [kappa * theta; -kappa];
G(1:2,3) = [r; -0.5];
G(:,4) = [0; 2 * kappa * theta; 0; -2*kappa; 0; 0] + sigma^2 * Q;
G(:,5) = [0; r; kappa * theta; -0.5; -kappa; 0] + rho * sigma * Q;
G(:,6) = [0; 1; 2 * r; 0; -1; 0];

end
```

```
Polynomial expansion price with N_pol=50:
    0.0969

Monte Carlo simulation price with NumSim=100000000:
:    0.0965

Absolute Error between polinomial al Monte Carlo price:
   4.0199e-04

Time for polynomail :    40.0991

Time for Monte Carlo:   960.8939
```

Published with MATLAB® R2020a