Report Laboratory on quantitative finance

This report presents my attempt to reproduce two research papers:"Low bias simulation scheme for Heston model by Inverse Gaussian approximation", Tse and Wan (2014) and "Variance Reduction for Asian Options under a General Model Framework", Dingeç, Sak and Hörmann (2014). In the first paper the authors analysed the Heston stochastic volatility model, trying to find a new scheme to simulate option prices comparing their results with other existing schemes (Quadratic expansion and Gamma expansion). Meanwhile the second paper develops a new Control variate method under a unified framework, working on Lévy processes, Heston stochastic volatility model and Regime switching. For coherence, I focus exclusively on the Heston model component of the second paper.

1 Low-bias simulation scheme for Heston model by Inverse Gaussian approximation, Tse and Wan (2014)

It is well known that closed-form solutions for path-dependent derivatives under the Heston model are unavailable. Therefor, Monte Carlo simulations are required for practical applications of this model. However, Monte Carlo simulation has two sources of error in calculating derivatives prices: variance and bias.The variance in Monte Carlo simulations originates from randomness and decreases as the number of samples increases, in accordance with the Law of Large Numbers. Bias, on the other hand, arises from the inexact time discretization of the underlying stochastic differential equations (SDES). To reduce bias, the authors design a more accurate discretization scheme for the SDEs. They build upon the work of Broadie and Kaya (2006), who designed an exact simulation scheme based on sampling from two distributions: the conditional transition distribution of variance, denoted by $(V(t_2)|V(t_1))$, and (2) the time-integrated variance conditional on the variance levels at the endpoints, denoted by I_c. The main results by Broadie and Kaya (2006) is an exact simulation scheme for I_c, but the numerical results in the literature shows that the exact scheme is computationally expensive. The exact scheme lead to the development of discretization schemes which approximate the exact distributions. Two of them are: Quadratic Exponential (QE) scheme (Andersen 2007), and the Gamma Expansion (GE) scheme (Glasserman and Kim 2011). The problem pointed out by Tse and Wan are:

1. the accuracy of the QE is sufficient when the number of steps is large, and it deteriorates when a small number of steps is used.
2. GE, while having an accuracy very similar to that of the exact scheme, is very expensive and its time-accuracy trade off is less favourable than QE when more than few time steps (two, three) are needed in pricing path dependent options.

Tse and Wan proposed a scheme to price path-dependent options with a moderate number of time steps (from four to sixteen) under Heston model of course. They proposed the so called IPZ-IG scheme which consists of using the Inverse Gaussian (IG) approximation to I_c (The paper proves the asymptotic exactness of this approximation) and the IPZ scheme for sampling $(V(t_2)|V(t_1)$.
In this paper the Heston stochastic volatility model is defined by the following equations:

$$\frac{dX(t)}{X(t)} = r\,dt + \sqrt{V(t)}\left(\rho\,dW_V(t) + \sqrt{1-\rho^2}\,dW_X(t)\right), \tag{1}$$

$$dV(t) = \kappa\big(\theta - V(t)\big)\,dt + \sigma\sqrt{V(t)}\,dW_V(t). \tag{2}$$

in which (W_V, W_X) is a standard two-dimensional Brownian motion in the time variable $t, \kappa, \theta, \sigma$ are positive constants, r is a non-negative constant, and the correlation ρ is a constant in $[-1, 1]$. The initial conditions $X(0)$ and $V(0)$ are assumed to be strictly positive. $X(t)$ represents the price of an underlying asset and $V(t)$ represents the variance of its instantaneous returns. Equation (2) represents a square root diffusion process, whose conditional transition density $(V(t_2)|V(t_1))$ distribution is known to be that of a scaled noncentral chi-square distribution (Andersen 2007). Assuming $t_2 > t_1$ the distribution of $V(t_2)$ conditional on $V(t_1)$ is given by

$$\begin{aligned} V(t_2) &= \frac{e^{-\kappa(t_2-t_1)}}{n(t_1,t_2)}\,\chi'^2_\delta\left(n(t_1,t_2)\,V(t_1)\right), \\ \delta &= \frac{4\kappa\theta}{\sigma^2}, \\ n(t_1,t_2) &= \frac{4\kappa e^{-\kappa(t_2-t_1)}}{\sigma^2\left(1-e^{-\kappa(t_2-t_1)}\right)}. \end{aligned} \tag{3}$$

Where $\chi'^2_\delta(\lambda)$ denotes a non-central chi-square random variable with δ degrees of freedom. It can be sampled also as follows ((Van Haastrecht and Pelsser 2008)

$$V(t_2) = \frac{2e^{-\kappa(t_2-t_1)}}{n(t_1,t_2)}\,\text{Gamma}\left[\text{Poisson}\left(\frac{V(t_1)\,n(t_1,t_2)}{2}\right) + \frac{\delta}{2}\right] \tag{4}$$

where Gamma(s) is a unit-scale gamma variate with shape parameter s and Poisson(mp) is a Poisson variate with mean parameter mp. As shown in Broadie and Kaya (2006) the distribution of $\log(X(t_2)/X(t_1))$ is conditionally normal given $V(t_1), V(t_2)$ and $I = \int_{t_1}^{t_2} V(s)ds$:

$$\log\left(\frac{X(t_2)}{X(t_1)}\right) \sim \mathcal{N}\Big(r(t_2-t_1) - \tfrac{1}{2}I + \frac{\rho}{\sigma}\big(V(t_2) - V(t_1) - \kappa\theta(t_2-t_1) + \kappa I\big),\ (1-\rho^2)\, I\Big). \tag{5}$$

This way simulating $(X(t_2), V(t_2))$ given $(X(t_1), V(t_1))$ reduces to sampling from the joint distribution of $(V(t_2)|V(t_1), I)$. Since we can sample $(V(t_2)|V(t_1))$ using (3) or (4), the problem is sampling

$$I_c = \left(\int_{t_1}^{t_2} V(s)ds \middle| V(t_1), V(t_2)\right).$$

Tse and Wan propose to approximate I_c by the Inverse Gaussian (IG) distribution. This is a family of distributions parametrized by mean parameter m and shape parameter s, which are determined by moment matching in their approximation. The mean and variance of IG(m, s) are m and m^3/s respectively. The authors derive explicit formulae for the first two moments of I_c and prove that the exact distribution converges to the moment-matched IG distribution as the time interval $t_2 - t_1$ goes to infinity. In the following I will recall some useful theoretical facts quoted by the paper:

Proposition 2.1: Let $\delta = 4\kappa\theta/\sigma^2$, $v = \delta/2 - 1$, $\triangle t = t_2 - t_1$, and $C_z = 2\kappa[\sigma^2 \sinh(\kappa\triangle t/2)]^{-1}$. The random variable I_S representing the time-integrated variance admits the representation

$$I_c \equiv \left(\int_{t_1}^{t_2} V(s)ds \mid V(t_1), V(t_2)\right) = X_1 + X_2 + \sum_{j=1}^{\eta} Z_j, \tag{6}$$

in which X_1, X_2, η, Z_1, Z_2, $\ldots$, Z_η are mutually independent, the Z_j are independent copies of a random variable Z. η is a Bessel random variable with parameters v and $z = C_z\sqrt{V(t_1)V(t_2)}$.

Proposition 3.1: Let $C_1 = \coth(\kappa\triangle t/2)$ and $C_2 = \operatorname{csch}^2(\kappa\triangle t/2)$. The mean $E[I_S]$ and the variance $\mathrm{Var}[I_c]$ of I_c are given by

$$E[I_c] = E[X_1] + E[X_2] + E[\eta]E[Z], \tag{7}$$

$$\mathrm{Var}[I_c] = \sigma^2_{X_1} + \sigma^2_{X_2} + E[\eta]\sigma^2_Z + (E[\eta]^2 - E[\eta]^2)E[Z]^2, \tag{8}$$

where

$$
\begin{aligned}
E[X_1] &= (V(t_1)+V(t_2))(C_1/\kappa - \triangle t C_2/2),\\
\sigma^2_{X_1} &= (V(t_1)+V(t_2))(\sigma^2 C_1/\kappa^3 + \sigma^2 \triangle t C_2/(2\kappa^2)\\
&\quad - \sigma^2(\triangle t)^2 C_1 C_2/(2\kappa)),\\
E[X_2] &= \delta\sigma^2(-2+\kappa\triangle t C_1)/(4\kappa^2),\\
\sigma^2_{X_2} &= \delta\sigma^4(-8+2\kappa\triangle t C_1+\kappa^2(\triangle t)^2 C_2)/(8\kappa^4),\\
E[Z] &= 4E[X_2]/\delta,\\
\sigma^2_Z &= 4\sigma^2_{X_2}/\delta,\\
E[\eta] &= zI_{v+1}(z)/(2I_v(z)),\\
E[\eta^2] &= z^2 I_{v+2}(z)/(4I_v(z)) + E[\eta].
\end{aligned}
$$

For the convergence results see section 3.3 and 3.4 of the paper.

1.1 Sampling of $(V(t_2)|V(t_1))$

Keeping in mind equation (4), the authors noted that generating Gamma(s) when $s < 1$ is significantly more costly than for $s \geq 1$. For typical Heston parameters, $\delta/2 < 1$, and so the case $s < 1$ occurs if and only if Poisson$(V(t_1)n(t_1,t_2)) = 0$. This case can occur very often, so it can be useful to use precomputation and interpolation to speed up the computation of the case $s = \delta/2$. The authors suggested methods to interpolate the values of $(V(t_2)|V(t_1))$, however, in my implementation, I use only the direct formula, which is slower but avoids introducing additional approximation errors.

1.2 Sampling I_c by the Inverse Gaussian approximation

As said before, the authors approximate I_c by the moment-matched IG distribution. The following code implements the algorithm proposed by the authors:

```
%generation of IG, Algorithm 3 of the paper
function ris=IG(m,s)
% parameter: m is the mean, s the shape
ris=zeros(size(m));
% generating a vectors N,U from multivariate normal
    and uniform distribution
N=randn(size(m));
U=rand(size(m));
x=1+N.^2./(2.*s./m)-sqrt((4.*s./m).*N.^2+(N.^4))./(2.*
    s./m);
mask=((U.*(1+x))>1);
ris(mask)=m(mask)./x(mask);
ris(~mask)=m(~mask).*x(~mask);
end
```

In section 4.3 the authors suggests to use interpolation again to make faster calculation of the formulae in proposition 3.1, I did not use this interpolation in my code. Indeed my implementation of the algorithms for pricing the European call option in the paper is

```
function estimation=simHestonIG(N,M,sigma,k,theta,V0,rho,r,T,X0,K)
%N= # of time steps; M= # of simulations; sigma,k,theta,V0,rho,r,T,X0,K are the option parameters
dt=T/N; %constant time step
% given by proposition 2.1
delta=4*k*theta/(sigma^2);
vu=delta/2-1;
% formulae given by propoposition 3.1 and equation 3
n_dt=4*k*exp(-k*dt)/(sigma^2*(1-exp(-k*dt)));
C_z=2*k*(sigma^2*sinh(k*dt/2))^(-1);
C_1=coth(k*dt/2);
C_2=csch(k*dt/2)^2;
E_X2=delta*sigma^2*(-2+k*dt*C_1)/(4*k^2);
sgX2=delta*sigma^4*(-8+2*k*dt*C_1+(k^2)*(dt^2)*C_2)/(8*(k^4));
E_Z=4*E_X2/delta;
sgZ=4*sgX2/delta;
V=zeros(N+1,M);
X=zeros(N+1,M);
V(1,:)=V0;
X(1,:)=X0;
% Preallocation Gaussian random variables
Gaussian=randn(N,M);




for i=1:N
    coeff=2*exp(-k*dt)/n_dt;
    m_p=poissrnd(V(i,:).*(n_dt/2));
    V(i+1,:)=coeff.*gamrnd(m_p+delta/2,1); %computing the next value of the variance
    z=sqrt(V(i,:).*V(i+1,:)).*C_z;
    % computation for the moment matching Ic, formulae of proposition 3.1
    E_nu=z.*besseli(vu+1,z)./(2.*besseli(vu,z));
    E_nu2=z.^2.*besseli(vu+2,z)./(4.*besseli(vu,z))+E_nu;
    E_X1=(V(i,:)+V(i+1,:)).*(C_1/k-dt*C_2/2);
    sgX1=(V(i,:)+V(i+1,:)).*(sigma^2*C_1/k^3+sigma^2*
        dt*C_2/(2*k^2)-(sigma^2)*(dt^2)*C_1*C_2/(2*k));
    E_Ic=E_X1+E_X2+E_nu.*E_Z;
    Var_Ic=sgX1+sgX2+E_nu.*sgZ+(E_nu2-E_nu.^2).*(E_Z
        ^2);
    % computation of Ic
    Ic=IG(E_Ic,(E_Ic.^3)./Var_Ic);
    % computing the log ratio
    log_ratio=r*dt-0.5.*Ic+(rho/sigma).*(V(i+1,:)-V(i
        ,:)-k*theta*dt+k.*Ic)+sqrt(Ic.*(1-rho^2)).*
        Gaussian(i,:);
    X(i+1,:)=exp(log_ratio).*X(i,:);

end

% price for the European Call option
S_T=X(N+1,:);
payoffs=max(S_T-K,0);
estimation=exp(-r*T)*mean(payoffs);

end
```

And we can use the above code also for pricing Asian Call option if we change the part after the for loop with

```
%price for the Asian Call Option
    Na_values=[2, 4, 8, 16];
estimated_price=zeros(size(Na_values));
for j=1:length(Na_values)
    Na=Na_values(j);

    sums=X(N/Na+1,:);
    for ii=2:Na
        sums = sums + X(ii * (N/Na) + 1, :);
    end
    sums=sums./Na;
    payoffs=max(sums-K,0);
    estimated_price(j)=exp(-r*T).*mean(payoffs);
```

1.3 Numerical results

The paper presents several test cases, which I use to compare my results with those of the authors. The paper's objective is to compare the IPZ-IG scheme with QE scheme (and indirectly GE scheme), so for such goal take a look at the paper.
The test scenarios are as follows:

Table 1: Heston model parameter cases (Tse and Wan)

Case	σ	κ	θ	$V(0)$	ρ	r	T	$X(0)$	K	Exact price
1	1	0.5	0.04	0.04	-0.9	0	10	100	100	13.08467014
2	1	0.5	0.04	0.04	-0.9	0	10	100	140	0.29577444
3	1	0.5	0.04	0.04	-0.9	0	10	100	70	35.84976970
4	0.61	6.21	0.019	0.010201	-0.7	0.0319	1	100	100	6.80611331
5	1	2	0.09	0.09	-0.3	0.05	5	100	100	34.99975835
6	0.5196	1.0407	0.0586	0.0194	-0.6747	0	4	100	100	15.16790670

Cases 1–3 cover at-the-money, out-of-the-money, and in-the-money options, respectively. Case 4 has a relatively short maturity $T = 1$ and large $\kappa = 6.21$. Case 5 has a big σ and a slightly negative $\rho = -0.3$. Case 6 has moderate parameters. A constant time step were used for each cases.
Table 1 reports the percentage bias (obtained by implementing the algorithm described above) for the European call option using 2^{23} simulations

$\lvert\%bias\rvert$	N=1	N=2	N=4	N=8	N=16
Case 1	1.1600	0.4787	1.0088	0.5609	0.0632
Case 2	3.3496	0.2064	2.6545	2.2603	0.6765
Case 3	0.4971	0.5425	0.5527	0.1308	0.0093
Case 4	0.0410	0.0579	0.0619	0.0049	0.0222
Case 5	0.0996	0.0448	0.0493	0.0412	0.0329
Case 6	0.0491	0.1825	0.0301	0.0853	0.0152

Table 2: Percentage bias for the price of the European call option

The results from the paper were

$\lvert\%bias\rvert$	N=1	N=2	N=4	N=8	N=16
Case 1	1.2320	0.4520	1.0358	0.4907	0.0811
Case 2	4.0352	0.3914	3.1003	2.3760	0.0706
Case 3	0.4932	0.5312	0.6011	0.0710	0.0241
Case 4	0.1144	0.1394	0.0685	0.0150	0.0947
Case 5	0.0884	0.0291	0.0177	0.0092	0.0660
Case 6	0.0353	0.2199	0.0406	0.0279	0.0214

Table 3: Percentage bias for the price of the European call option (IPZ-IG)

The authors also report the computational time after improving the efficiency of their code through preallocation and interpolation. Since these techniques were not implemented in my replication, the reported time comparisons are not meaningful in this context. As shown in the tables, the numerical results are nevertheless comparable. In particular, there is a clear tendency toward smaller bias as the number of time steps N increases. The largest bias is observed in Case 2 (the out-of-the-money scenario), while Cases 5 and 6 exhibit the smallest bias. Table 4 presents the arithmetic Asian call option prices obtained using a fixed number of time steps ($N = 128$) and computing the mean on $N_A \in \{1, 2, 4, 8, 16\}$ steps. In this simulation I used 10^5 trajectories instead of 2^{30} as the paper does.

	$N_A = 2$	$N_A = 4$	$N_A = 8$	$N_A = 16$
Case 1	10.2544	8.9275	8.2867	7.9637
Case 2	0.0825	0.0396	0.0247	0.0189
Case 3	33.7660	33.0026	32.6351	32.4505
Case 4	5.2001	4.3984	4.0037	3.8077
Case 5	26.3525	22.2671	20.2301	19.2274
Case 6	11.3981	9.6442	8.8014	8.3953

Table 4: My prices of the Asian call option

Meanwhile the results obtained by the paper are

	$N_A = 2$	$N_A = 4$	$N_A = 8$	$N_A = 16$
Case 1	10.297189	8.963870	8.322643	8.000178
Case 2	0.083577	0.040602	0.026770	0.020761
Case 3	33.837538	33.067611	32.694753	32.500390
Case 4	5.187462	4.389704	3.996457	3.801746
Case 5	26.378587	22.245779	20.214603	19.203531
Case 6	11.469086	9.708003	8.857739	8.441458

Table 5: Paper's prices of the Asian call option with 2^{30} simulations

The results remain comparable, considering the substantially smaller sample size used in my implementation. Overall, the relative errors remain below 1% for Cases 1, 3, 4, 5, and 6 across all values of N_A. Case 2 shows larger relative errors (1-9%) due to the very small magnitude of the option prices (order of 10^{-2}), which amplifies relative discrepancies despite small absolute differences. Given that I use approximately 10^4 times fewer simulations, one would theoretically expect errors roughly $\sqrt{10^4} = 100$ times larger under standard Monte Carlo theory (since Monte Carlo error typically decreases at a rate of $O(1/\sqrt{N})$), yet the observed discrepancies remain remarkably small, highlighting the efficiency of the IG method (since I did not implement the IPZ interpolation component for sampling $(V(t_2)|V(t_1))$).

1.4 Summary

This implementation reproduces the key numerical results of Tse and Wan (2014), demonstrating the accuracy of the Inverse Gaussian approximation to I_c. While the IPZ interpolation scheme for $(V(t_2)|V(t_1))$ and the fast moment calculation were not implemented, the results confirm that the IG approximation maintains low bias across diverse parameter scenarios. The largest discrepancies occur in Case 2, consistent with the paper's findings, where the out-of-the-money option with extreme parameters ($\sigma = 1$, $T = 10$, $\rho = -0.9$) presents the most challenging numerical case.

2 Variance Reduction for Asian Options under a General Model Framework, Dingeç, Sak and Hörmann (2014).

In this paper, a new variance reduction method is presented for a general model framework. The special cases considered by the paper are Lévy processes, Heston stochastic volatility, and regime switching models. I implement their method exclusively for the Heston model. The method proposed combines control variate with conditional Monte Carlo. In particular this method is used for pricing Arithmetic Asian call options (put option prices can be obtained via put-call parity). The technical requirement for their control variate (CV) method is the availability of the multivariate characteristic function of the log-returns vector (so it can be used in more models as the ones proposed), the conditional Monte Carlo (CMC) method is based on the unified normal mean variance mixture representation of the three models.
Lets define the following notation for the price process

$$S(t) = S(0)e^{X(t)}.$$

In pricing path dependent options we are interested in their log-return vector $(\Delta X_1, \ldots, \Delta X_d)$. Under the three proposed models (in particular the Heston stochastic volatility model (HSVM)) the log-returns ($\Delta X_i = X(t_i) - X(t_{i-1})$) admits the following representation

$$\Delta X_i = \Gamma_i + \Lambda_i Z_i \quad i = 1, \ldots, d \tag{1}$$

where log-returns are observed at d time points $0 < t_1 < \cdots < t_d$, assuming constant time intervals $\Delta t = t_i - t_{i-1}$. Γ_i, Λ_i are random variates functions of a single stochastic process $\{V(t), t \geq 0\}$, in particular we can say $\Gamma = f_m(V)$ and $\Lambda = f_v(V)$. The Z_i's are i.i.d. standard normal variables independent of $V(t)$.So, Equation (1) corresponds to a multivariate normal mean variance mixture representation where the means and the variances are driven by the process $V(t)$.
The authors adopt a slightly different notation than the one used in Section 1, in this part I will use this new notation for the HSVM

$$\begin{aligned}
\frac{dS(t)}{S(t)} &= r\,dt + \sqrt{V(t)}\left[\rho\,dB_1(t) + \sqrt{1-\rho^2}\,dB_2(t)\right], \\
dV(t) &= \alpha(\beta - V(t))\,dt + \sigma\sqrt{V(t)}\,dB_1(t),
\end{aligned}$$

where $(B_1(t), B_2(t))$ is a two dimensional standard Brownian motion. Under this framework, the i-th log-return is given by

$$\Delta X_i = (r - \alpha\beta\rho/\sigma)\Delta t + \rho/\sigma(V(t_i) - V(t_{i-1})) + (\alpha\rho/\sigma - 0.5)\int_{t_{i-1}}^{t_i} V(u)du$$
$$+ \left(\sqrt{(1-\rho^2)\int_{t_{i-1}}^{t_i} V(u)du}\right) Z_i, \quad \text{for } i = 1, \ldots, d, \tag{2}$$

where Z_i are our i.i.d. standard normal variables independent of $V(t)$. This way we have

$$\Gamma_i = (r - \alpha\beta\rho/\sigma)\Delta t + \rho/\sigma(V(t_i) - V(t_{i-1})) + (\alpha\rho/\sigma - 0.5)\int_{t_{i-1}}^{t_i} V(u)du,$$

and

$$\Lambda_i = \sqrt{(1-\rho^2)\int_{t_{i-1}}^{t_i} V(u)du.}$$

Notation correspondence: In this section, we adopt the notation from Dingeç, Sak, and Hörmann (2014), where:

- α corresponds to κ in Section 1 (mean-reversion speed),
- β corresponds to θ in Section 1 (long-run variance level).

All other parameters (σ, ρ, r) have the same meaning in both papers.
As previously discussed, the payoff of an arithmetic average Asian call option depends on the path of the stock price process observed at d time points and is given by

$$P_A(S(t_1, \ldots, S(t_d)) = (A - K)^+$$

where $A = \sum_{i=1}^{d} S(t_i)/d$ is the average of the stock price, K is the strike price and $t_d = T$ denots the maturity. The price of the option is given by the discounted risk neutral expectation of the payoff function

$$e^{-rT}\mathbb{E}[P_A(S(t_1), \ldots, S(t_d))],$$

where r is the risk-free interest rate.
For the HSVM, simulating the variance path V requires computing the integral previously denoted as I_c. The authors suggest using the algorithm from Glasserman (2004), which I implement as follows:

```
function [V,Ic]=Glasserman_pathV(d,dt,V_0,alpha,beta,
    sigma)
%from pag 124 Glasserman
dparam=4*alpha*beta/sigma^2;
k=20;%number used for the integral
```

```
dt_small=dt/k;
V=zeros(k*d+1,1);
Ic=zeros(d,1);
V(1)=V_0;
% Generate variance path V(t)
if dparam>1
    for ii=1:k*d
        c=sigma^2*(1-exp(-alpha*dt_small))/(4*alpha);
        lambda=V(ii)*exp(-alpha*dt_small)/c;
        Z=randn(1);
        X=chi2rnd(dparam-1);
        V(ii+1)=c*((Z+sqrt(lambda))^2+X);
    end
end
if dparam<=1
    for ii=1:k*d
        c=sigma^2*(1-exp(-alpha*dt_small))/(4*alpha);
        lambda=V(ii)*exp(-alpha*dt_small)/c;
        N=poissrnd(lambda/2);
        X=chi2rnd(dparam+2*N);
        V(ii+1)=c*X;

    end
end
% computing the integral with the trapezoidal rule
for j=1:d
    idx_start=(j-1)*k+1;
    idx_end=j*k+1;
    V_seg=V(idx_start:idx_end);
    for m = 1:k
        Ic(j) = Ic(j) + (V_seg(m) + V_seg(m+1))/2 *
            dt_small;
    end
end
% Extract V(t) at d+1 observation points
V=V(1:k:end);


end
```

The integral is approximated using the method of Andersen (2008):

$$\int_{t_{i-1}}^{t_i} V(u)du \approx \sum_{j=1}^{k} (\Delta t/k)(\gamma_1 V(u_{j-1}) + \gamma_2 V(u_j)),$$

where $t_{i-1} = u_0 < u_1 < \ldots < u_k = t_i$ and $u_j - u_{j-1} = \Delta t/k$, $j = 1, \ldots, k$.

Notice that if we select $\gamma_1 = \gamma_2 = 1/2$, we simply get the trapezoidal rule. Alternative methods for simulating the variance path and computing the integral exist, such as the IPZ-IG scheme discussed in Section 1. I tried also to use this kind of V-path simulation using the following code, which is a modified version of the code from Section 1, utilizing the IG function defined previously:

```
function [V,Ic]=pathsim(d,dt,alpha,beta,sigma,V_0)

delta=4*alpha*beta/(sigma^2);
vu=delta/2-1;

n_dt=4*alpha*exp(-alpha*dt)/(sigma^2*(1-exp(-alpha*dt)
    ));
C_z=2*alpha*(sigma^2*sinh(alpha*dt/2))^(-1);
C_1=coth(alpha*dt/2);
C_2=csch(alpha*dt/2)^2;
E_X2=delta*sigma^2*(-2+alpha*dt*C_1)/(4*alpha^2);
sgX2=delta*sigma^4*(-8+2*alpha*dt*C_1+(alpha^2)*(dt^2)
    *C_2)/(8*(alpha^4));
E_Z=4*E_X2/delta;
sgZ=4*sgX2/delta;
V=zeros(d+1,1);

V(1)=V_0;
Ic=zeros(d,1);




for i=1:d
    coeff=2*exp(-alpha*dt)/n_dt;
    m_p=poissrnd(V(i)*(n_dt/2));
    V(i+1)=coeff*gamrnd(m_p+delta/2,1);
    z=sqrt(V(i,:)*V(i+1,:))*C_z;
    E_nu=z.*besseli(vu+1,z)/(2*besseli(vu,z));
    E_nu2=z.^2.*besseli(vu+2,z)/(4*besseli(vu,z))+E_nu
        ;
    E_X1=(V(i)+V(i+1))*(C_1/alpha-dt*C_2/2);
    sgX1=(V(i)+V(i+1))*(sigma^2*C_1/alpha^3+sigma^2*dt
        *C_2/(2*alpha^2)-(sigma^2)*(dt^2)*C_1*C_2/(2*
        alpha));
    E_Ic=E_X1+E_X2+E_nu*E_Z;
    Var_Ic=sgX1+sgX2+E_nu*sgZ+(E_nu2-E_nu^2)*(E_Z^2);
    Ic(i)=IG(E_Ic,(E_Ic^3)/Var_Ic);
end
```

However, this simulation approach occasionally produces numerical issues,specifically, approximately 0.4% of the simulated prices result in NaN values. Recomputing these cases yields accurate results.Nevertheless, I adopt the Glasserman (2004) implementation to remain consistent with the paper's methodology.

2.1 Typical CV methods under the unified framework

The Monte Carlo simulation for the expectation of a variable Y using a control variate W is represented as

$$Y_{CV} = Y - c(W - E[W]).$$

The optimal coefficient minimizing the variance of Y_{CV} is $c^* = \mathrm{Cov}(Y, W)/\mathrm{Var}(W)$. The variance reduction factor (VRF) of the CV method is $VRF = 1/(1-\rho^2)$ which is increasing with respect to ρ, the correlation between Y an W. Kemna and Vorst (1990) showed that a CV for the arithmetic average Asian options is the geometric average Asian option for which we have the following payoff

$$\begin{aligned} P_G &= (G-K)^+ \\ G &= \left(\prod_{i=1}^{d} S(t_i)\right)^{1/d} = \exp\left(\frac{\sum_{i=1}^{d} \log(S(t_i))}{d}\right) \end{aligned}$$

The paper than briefly describes two typical CV methods that can be used for the simulation of the Asian options under the presented framework.

2.2 The new CV method

If we use G as a conditioning variable, the payoff function P_A (the payoff of the arithmetic Asian option) can be split into two parts

$$(A-K)^+ = (A-K)^+\mathbf{1}_{\{G\leq K\}} + (A-K)^+\mathbf{1}_{\{G>K\}} = (A-K)^+\mathbf{1}_{\{G\leq K\}} + (A-K)\mathbf{1}_{\{G>K\}}, \quad (3)$$

where the last equality comes from the fact that the arithmetic mean is always strictly greater than the geometric one. The authors used the second term of the last part of the above equation as CV for P_A. This way we get

$$Y_{CV} = P_A - c(W - E[W]),$$

where $W = (A-K)\mathbf{1}_{\{G>K\}}$.
The expectation of our CV, $\mu_W = E[W]$ can be written as

$$\mu_W = E[W] = E[(A-K)\mathbf{1}_{\{G>K\}}] = E[(A-K)\mathbf{1}_{\{\bar{X}>L\}}] = E[(E[A|\bar{X}]-K)\mathbf{1}_{\{\bar{X}>L\}}],$$

where $L = \log(K/S(0))$ and $\bar{X} = \log(G/S(0)) = \frac{1}{d}\sum_{j=1}^{d} X_j$, where X_j stands for $X(j)$ for simplicity. After some mathematical passages the authors got to

$$\mu_W = \frac{g(0)}{2} + \frac{1}{2\pi}\int_{-\infty}^{\infty} \frac{Im(e^{-i\omega L} g(\omega))}{\omega} d\omega, \quad (*)$$

where $Im(x)$ stands for the imaginary part of the number x and the function $g(\omega)$ is given by

$$g(\omega) = \frac{S(0)}{d} \sum_{j=1}^{d} \varphi_{\bar{X},X_j}(\omega, -i) - K\varphi_{\bar{X}}(\omega) \tag{4}$$

where i is the imaginary unit and $\varphi_{\bar{X},X_j}(\omega_1, \omega_2)$ the bivariate characteristic function of $\bar{X}$ and X_j

$$\varphi_{\bar{X},X_j}(\omega_1, \omega_2) = E[\exp(i\omega_1\bar{X} + i\omega_2 X_j)]$$

and $\varphi_{\bar{X}}(\omega) = E[e^{i\omega\bar{X}}]$ the characteristic function of $\bar{X}$. Then we can see that

$$g(0) = E[A] - K = \frac{S(0)}{d} \sum_{j=1}^{d} e^{jr\Delta t} - K.$$

The integral in equation (*) can be computed using numerical integration routine, meanwhile the characteristic functions are special to the model of interest, and we are looking for the Heston model.
Let $\varphi_{\Delta X}(u)$ denote the joint characteristic function of the random vector $(\Delta X_1, \ldots, \Delta X_d)$, that is

$$\varphi_{\Delta X}(u) = E[e^{iu^T\Delta X}].$$

It can be seen that

$$\varphi_{\bar{X},X_j}(\omega_1, \omega_2) = E[\exp(i\omega_1\bar{X} + i\omega_2 X_j)] = \varphi_{\Delta X}(u_1, \ldots, u_d),$$

where $u_l = \omega_1(d - l + 1)/d + \omega_2$ for $l = 1, \ldots, j$ and $u_l = \omega_1(d - l + 1)/d$ for $l = j + 1, \ldots, d$. It can be also seen that

$$\varphi_{\bar{X}}(\omega_1) = \varphi_{\Delta X}(u_1, \ldots, u_d),$$

where $u_j = \omega_1(d - j + 1)/d$ for $j = 1, \ldots, d$. The formula for the computation of $\varphi_{\Delta X}(u)$ are taken from an adaptation of the general formula of Rockinger and Semenova (2005) for the Heston model. The formula is given by the following

$$\varphi_{\Delta X}(u) = \mathrm{E}\left[e^{iu^T\Delta X}\right] = \exp\left(\sum_{j=1}^{d} C(\Delta t; u_j, u_j^*) + D(\Delta t; u_1, u_1^*)V(0)\right), \tag{5}$$

where $u_d^* = 0$, $u_j^* = -iD(\Delta t; u_{j+1}, u_{j+1}^*)$, $j = 1, \ldots, d - 1$, and

$$C(\tau; w_1, w_2) = iw_1 r\tau + \frac{\alpha\beta}{\sigma^2}\left[\log\left(\frac{h^2(1 + \gamma^2)}{\sigma^2(2biw_2 - \sigma^2 w_2^2 + iw_1(iw_1 - 1))}\right) - b\tau\right],$$
$$D(\tau; w_1, w_2) = \frac{\gamma h - b}{\sigma^2},$$

with

$$\begin{aligned}
b(w_1, w_2) &= iw_1\sigma\rho - \alpha, \\
h(w_1, w_2) &= \sqrt{\sigma^2 iw_1(iw_1 - 1) - b^2}, \\
\gamma(w_1, w_2) &= \tan\left(\frac{h\tau}{2} + \arctan\left(\frac{b + iw_2\sigma^2}{h}\right)\right).
\end{aligned}$$

My implementation (from the paper's algorithm and formula) of the formulae above and the function $g(\omega)$ are the following

```
function ris=charfun(u,dt,d,V_0,alpha,beta,sigma,rho,r
    )
%control on the dimension of u
if ~isvector(u) || length(u) ~= d
    error('charf:dimensionError','u␣must␣be␣a␣vector␣
        of␣length␣d.');
end
%putting the result equal to 1 when u is close to zero
tol = 1e-10;
if norm(u,Inf) < tol
    ris = 1;
    return
end
%creating the vector u_star
u_star=zeros(d,1);

for j=d-1:-1:1
    u_star(j)=-1i*Dfun(u(j+1),u_star(j+1),dt,alpha,
        sigma,rho);
end
% vector to be summed as can be seen in the formula
    for the characteristic function
Cvec=zeros(d,1);
for ii=1:d
    Cvec(ii)=Cfun(u(ii),u_star(ii),dt,r,alpha,beta,
        sigma,rho);
end
% value of the characteristic function
ris=exp(sum(Cvec)+Dfun(u(1),u_star(1),dt,alpha,sigma,
    rho)*V_0);
% function which are given by the formulas
    function [b,h,gamma]=usefulfun(w1,w2,alpha,sigma,
        rho,dt)
        b=1i*w1*sigma*rho-alpha;
        h=sqrt(sigma^2*1i*w1*(1i*w1-1)-b^2);
        gamma=tan(h*dt/2+atan((b+1i*w2*sigma^2)/h));
    end
    function Cval=Cfun(w1,w2,dt,r,alpha,beta,sigma,rho
        )
        [bval,hval,gammaval]=usefulfun(w1,w2,alpha,
            sigma,rho,dt);
        Cval=1i*w1*r*dt+alpha*beta/sigma^2*(log(hval
            ^2*(1+gammaval^2)/(sigma^2*(2*bval*1i*w2-
            sigma^2*w2^2+1i*w1*(1i*w1-1))))-bval*dt);
    end
    function Dval=Dfun(w1,w2,dt,alpha,sigma,rho)
        [bval,hval,gammaval]=usefulfun(w1,w2,alpha,
            sigma,rho,dt);
        Dval=(gammaval*hval-bval)/sigma^2;
    end
end
```

The function $g(\omega)$ in Equation (4) is evaluated using the characteristic function. For the Heston model, the joint characteristic function $\varphi_{\Delta X}(u)$ is given by Equation (5), which requires recursive computation of the coefficients u_j^* and evaluation of the functions $C(\tau; w_1, w_2)$ and $D(\tau; w_1, w_2)$.

```
function g=g(w,d,S0,K,dt,V_0,alpha,beta,sigma,rho,r)
% creating vectors for more efficiency
a=zeros(d,1);
b=zeros(d,1);
x=zeros(d,1);
for j=1:d
    a(j)=w*(d-j+1)/d-1i;
    b(j)=w*(d-j+1)/d;
end
for ii=1:d-1
    v=b;
    v(1:ii)=a(1:ii);
    x(ii)=charfun(v,dt,d,V_0,alpha,beta,sigma,rho,r);
end
x(d)=charfun(a,dt,d,V_0,alpha,beta,sigma,rho,r);
y=charfun(b,dt,d,V_0,alpha,beta,sigma,rho,r);
% final value from the formula
g=S0/d*sum(x)-K*y;
end
```

One call to $\varphi_{\Delta X}(u)$ requires $O(d)$ computations. So the computational complexity for the g function is $O(d^2)$.
In this new CV method (CVN) the authors set $c = 1$ due to the results of their numerical experiments which showed that the optimal $c^* = \mathrm{Cov}(P_A, W)/\mathrm{Var}(W)$ is very close to one. By setting $c = 1$ the CV estimator becomes $Y_{CV} =$

$P_A - W + \mu_W$. Thus, the Equation (3) implies that

$$Y_{CV} = Y + \mu_W \quad \text{with} \quad Y = (A-K)^+ \mathbf{1}_{\{G \le K\}}.$$

This can be interpreted as an *indirect estimator* because it indirectly estimate the price by simulating Y and adding μ_W instead of simulating the payoff $(A-K)^+$ itself. For such a simulation importance sampling and conditional Monte Carlo (CMC) are the most suitable variance reduction techniques.

2.3 CMC

Lets rewrite Y as a function of $Y = q(V, Z)$. Its variance can be written as

$$\mathrm{Var}(Y) = E[\mathrm{Var}(Y|V)] + \mathrm{Var}(E[(Y|V)]).$$

The first term is the variance due to Z while the second explains the variance from V. The goal is to reduce the first term by exploiting the analytical properties of the normal distribution. The important point of the CVN is that the variance to which Z contributes is already reduced from $\mathrm{Var}(P_A) = \mathrm{Var}((A-K)^+)$ to $\mathrm{Var}(Y) = \mathrm{Var}((A-K)^+ \mathbf{1}_{\{G \le K\}})$. The authors tried to improve CVN by adopting CMC to reduce further the variance originating from Z.
Starting by noticing that the standard multinormal vector Z can be simulated in a specific direction $\theta \in \mathbb{R}^d$, $||\theta|| = 1$ by the formula

$$Z = \theta \Xi + (I_d - \theta\theta^T)Z', \quad \Xi \sim \mathcal{N}(0,1), Z' \sim \mathcal{N}(0, I_d) \tag{6}$$

which can be rewritten

$$Z = \theta\Xi + Z' - \theta(\theta^T Z').$$

The above formula corresponds to the simulation of Z conditional on the linear combination $\Xi = \theta^T Z = \sum_{i=1}^d \theta_i Z_i$. For this problem, the authors selected the direction depending on V,

$$\vartheta_i(V) = \frac{(d-i+1)\Lambda_i}{\sqrt{\sum_{j=1}^d (d-j+1)^2 \Lambda_j^2}}, \quad i = 1, \ldots, d,$$

where Λ has been described above. The idea is to use $E[Y|Z', V]$ as an estimator.

2.3.1 Evaluation of the conditional expectation

First write the arithmetic average A as a function of the three random inputs: $A = \zeta(\Xi, Z', V)$ where $\Xi \sim N(0, I)$ and $Z' \sim N(0, I_d)$. Then we get

$$\begin{aligned} \mathrm{E}[V|Z' = z, V = v] &= \mathrm{E}[(\zeta(\Xi, Z', V) - K)^+ \mathbf{1}_{\{\Xi \le k(v)\}} | Z' = z, V = v] \\ &= \int_{-\infty}^{k(v)} (\zeta(x, z, v) - K)^+ \phi(x) dx, \end{aligned} \tag{7}$$

where $\phi(\cdot)$ is the standard normal density,

$$k(v) = \frac{\log K - \mu(v)}{\sigma(v)}, \tag{8}$$

$$\mu(v) = \mathrm{E}[\log G | V = v] = \log S(0) + \frac{1}{d}\sum_{i=1}^{d} \Gamma_i (d - i + 1),$$

and

$$\sigma^2(v) = \mathrm{Var}(\log G | V = v) = \frac{1}{d^2}\sum_{i=1}^{d} \Lambda_i^2 (d - i + 1)^2, \tag{9}$$

where $\Gamma = f_\mu(v)$ and $\Lambda = f_\sigma(v)$ are given by the functionals of V described above. As the integral in Equation (7) is one dimensional, the authors tried to find a closed form expression for $\mathrm{E}[V|Z' = z, V = v]$. For that purpose, we write the following expression for $A = \zeta(x, z, v)$,

$$\zeta(x, z, v) = \frac{1}{d}\sum_{i=1}^{d} e^{a_i(v) s_i(z,v)},$$

by using the conditional simulation formula given in Equation (6). Here the coefficients s_i and a_i, $i = 1, \dots, d$ are randomly changing depending on Z' and V. They are given by

$$a_i(v) = \sum_{j=1}^{i} \Lambda_j \vartheta_j(v),$$

and

$$s_i(z, v) = S(0) \exp\left(\sum_{j=1}^{i} \Gamma_j + \Lambda_j \xi_j\right),$$

where $\xi = z - \vartheta(\vartheta^T z)$. It can be seen that the s_i's can be calculated by the following recursion

$$s_j(z, v) = s_{j-1}(z, v) \exp(\Gamma_j + \Lambda_j \xi_j),$$

with $s_0(z, v) = S(0)$.

Since all a_i's are positive, $\zeta(x, z, v)$ is a strictly increasing function of x. So, $\zeta(x, z, v) - K$ has a unique root. Let $b(z, v)$ denotes the root for a given fixed z and v, that is, $\zeta(b(z, v), z, v) - K = 0$. Then the integral in Equation (7) can be rewritten as

$$\begin{aligned}
\mathrm{E}[V|Z' = z, V = v] &= \int_{b(z,v)}^{k(v)} (\zeta(x,z,v) - K)\phi(x)dx \\
&= \int_{b(z,v)}^{k(v)} \left(\frac{1}{d} \sum_{i=1}^{d} e^{a_i(v)} s_i(z,v) - K \right) \phi(x)dx \\
&= \frac{1}{d} \sum_{i=1}^{d} s_i(z,v) \int_{b(z,v)}^{k(v)} e^{a_i(v)x} \phi(x)dx - K \int_{b(z,v)}^{k(v)} \phi(x)dx \\
&= \frac{1}{d} \sum_{i=1}^{d} s_i(z,v) e^{a_i(v)^2/2} [\Phi(k(v) - a_i(v)) - \Phi(h(z,v) - a_i(v))] \\
&\quad - K[\Phi(k(v)) - \Phi(h(z,v))].
\end{aligned} \tag{10}$$

For the evaluation of $b(z,v)$, Newton's root finding method is used with the following algorithm

```
function zero=Newton(fun,Dfun,x0,tol,nmax)
%pag228 of the book Matematica Numerica
err=tol+1;
iter=0;
fx0=fun(x0);
while iter<nmax && err>tol
    dfx0=Dfun(x0);
    if dfx0==0
        fprintf('Stop because the derivative is zero\n
           ');
        zero=[];
        return
    end
    x=x0-fx0/dfx0;
    err=abs(x-x0);x0=x;fx0=fun(x0);iter=iter+1;
end
zero=x0;
end
```

This method requires the first derivative of the function $p(x)$ of which the root is evaluated. In this case, $p(x) = \zeta(x,z,v) - K = \frac{1}{d}\sum_{i=1}^{d} e^{a_i(v)x} s_i(z,v) - K$ and $p'(x) = \frac{1}{d}\sum_{i=1}^{d} a_i e^{a_i x} s_i$. As the starting point for Newton's method the authors chose

$$x_0(z,v) = k - \frac{p(k)}{p'(k)},$$

which is the root of the first-order approximation of $p(x)$ at the point k. It should be noted that CMC method reduces the variance but it increases the execution time. The following are the implementation of the algorithms in the paper for the calculation of the conditional expectation and the computation for the Asian call option price (CVN+CMC)

```
function [ris,a,kv,s,sigmaquad,b]=conditionalexp(Z,
    Vpath,Ic,d,S0,K,alpha,beta,sigma,rho,r,dt)
% creating the vector for efficiency
Gamma=zeros(d,1);
Lambda=zeros(d,1);
Theta=zeros(d,1);
coeff=0;
coeff2=0;
coeff3=0;
%following the paper's formulae
for j=1:d
    Gamma(j)=(r-alpha*beta*rho/sigma)*dt+(rho/sigma)*(
        Vpath(j+1)-Vpath(j))+(alpha*rho/sigma-0.5)*Ic(j
        );
    Lambda(j)=sqrt((1-rho^2)*Ic(j));
    coeff=coeff+(d-j+1)^2*Lambda(j)^2;
    coeff2=coeff2+Gamma(j)*(d-j+1);
    coeff3=coeff3+Lambda(j)^2*(d-j+1)^2;
end
for jj=1:d
    Theta(jj)=(d-jj+1)*Lambda(jj)/sqrt(coeff);
end
mu=log(S0)+coeff2/d;
sigmaquad=coeff3/d^2;
kv=(log(K)-mu)/sqrt(sigmaquad);
xi=Z-Theta*(Theta'*Z);
s=zeros(d+1,1);
a=zeros(d+1,1);
s(1)=S0;
for ii=2:d+1
    a(ii)=a(ii-1)+Lambda(ii-1)*Theta(ii-1);
    s(ii)=s(ii-1)*exp(Gamma(ii-1)+Lambda(ii-1)*xi(ii
        -1)); % watch out for the right indices
end
%setting Netwon's method
pfun=@(x) sum(exp(a(2:d+1).*x).*s(2:d+1))/d-K;% watch
    out for the right indices
Dpfun=@(x) sum(s(2:d+1).*a(2:d+1).*exp(a(2:d+1).*x))/d
    ;% watch out for the right indices
x0=kv-pfun(kv)/Dpfun(kv);
b=Newton(pfun,Dpfun,x0,1e-8,100);
vec=zeros(d,1);
```

```
for jj=1:d
    vec(jj)=s(jj+1)*exp(a(jj+1)^2/2)*(normcdf(kv-a(jj
        +1))-normcdf(b-a(jj+1)));% watch out for the
        right indices
end
ris=sum(vec)/d-K*(normcdf(kv)-normcdf(b));
end
```

```
function [price,conf_int]=Varreduction(dt,T,d,n,S0,K,
    V_0,alpha,beta,sigma,rho,r)
g_0=(S0/d)*sum(exp((1:d).*(r*dt)))-K;
L=log(K/S0);
% mu_W computation
integrand=@(w) arrayfun(@(ww) imag(exp(-1i*ww*L)*g(ww,
    d,S0,K,dt,V_0,alpha,beta,sigma,rho,r))/ww,w);
mu_W=g_0/2+integral(integrand,-Inf,Inf)/(2*pi);
Y=zeros(n,1);

for ii=1:n
        [Vpath,Ic]=Glasserman_pathV(d,dt,V_0,alpha,
            beta,sigma);
        Z=randn(d,1);
        [condexp,~,~,~,~,~]=conditionalexp(Z,Vpath,Ic,
            d,S0,K,alpha,beta,sigma,rho,r,dt);
        Y(ii)=exp(-r*T)*(condexp+mu_W);
end
price=mean(Y);
conf_int=norminv(1-0.05/2)*std(Y)/sqrt(n);
end
```

For a faster version one could use the trapezoidal rule instead of the built-in MATLAB function integral, substituting that two lines in the code with

```
% Grid for numerical integration with trapezoidal rule
w_max = 100;  % upper limit (empirically set)
dw = 0.1;     % integration step
w = -w_max:dw:w_max;
w(w==0) = [];  % remove w=0 to avoid division by zero

% Vectorized computation of the integrand
integrand_vals = zeros(size(w));
for i = 1:length(w)
    integrand_vals(i) = imag(exp(-1i*w(i)*L)*g(w(i),d,
        S0,K,dt,V_0,alpha,beta,sigma,rho,r))/w(i);
end

```

```
% trapezoidal rule
mu_W = g_0/2 + trapz(w, integrand_vals)/(2*pi);
```

2.4 Estimating greeks (Δ, Γ)

The authors suggests to use finite difference, pathwise derivative (PD) and likelihood ratio (LR) methods for the simulation of sensitivities. I decided to implement the PD method which is the one that can be applied to the CVN+CMC algorithm that has been described so far. This method requires the first and second derivative of μ_W (which formula was written before Equation(4)) with respect to $S(0) = S_0$. The following are the formulas we need

$$\mu'_W = \frac{\gamma}{2} + \frac{1}{2\pi}\int_{-\infty}^{+\infty} Im(g_1(w))dw \quad \text{and} \quad \mu''_W = \frac{1}{2\pi}\int_{-\infty}^{+\infty} Im(g_2(w))dw, \tag{11}$$

where

$$q_1(w) = \frac{e^{-iwL}}{w}\left[\frac{iw+1}{d}C_1 - \frac{iw}{S_0}C_2\right] \quad \text{and} \quad q_2(w) = \frac{e^{-iwL}}{S_0}\left[\frac{(i-w)}{d}C_1 + \frac{(i+w)}{S_0}C_2\right], \tag{12}$$

$$L = \log(K/S_0), \quad C_1 = \sum_{j=1}^{d}\varphi_{\tilde{X},X_j}(w,-i), \quad C_2 = K\varphi_{\tilde{X}}(w), \quad \text{and} \tag{13}$$

$$\gamma = 1/d\sum_{j=1}^{d} e^{jr\Delta t}.$$

Then let $f(S_0)$ denote the function corresponding to the simulation output of the CMC method (see Equation(10)). The estimator for the option price is $e^{-rT}(f(S_0) + \mu_W(S_0))$. In the PD method we take the derivatives of such estimator to estimate Δ, Γ, so we look at $e^{-rT}(f'(S_0) + \mu'_W(S_0))$ and $e^{-rT}(f''(S_0) + \mu''_W(S_0))$. The function $f(S_0)$ has the form

$$f(S_0) = \int_b^k (A(y) - K)\phi(y)dy \quad \text{where} \quad A(y) = \frac{1}{d}\sum_{i=1}^{d} e^{a_i y}s_i,$$

and b, k, s_i are the one described above. When we take the derivatives of f by using Leibniz rule and implicit differentiation, we get

$$f' = \frac{1}{S_0}\left[\frac{1}{d}\sum_{i=1}^{d} s_i e^{a_i^2/2}[\Phi(k-a_i) - \Phi(b-a_i)] - \frac{\phi(k)}{\sigma}(A(k)-K)\right],$$

$$f'' = \frac{1}{S_0^2}\left(\frac{K^2\phi(b)}{A'(b)} + \frac{\phi(k)}{\sigma^2}[A'(k) - 2A(k)\sigma + (A(k)-K)(\sigma-k)]\right),$$

where $A(y) = \frac{1}{d}\sum_{i=1}^{d} e^{a_i y} s_i$, $A'(y) = \frac{1}{d}\sum_{i=1}^{d} a_i e^{a_i y} s_i$, and σ^2 are given by Equation (9).
I leave my implementation of the above formula, but first we need to reconstruct the characteristic functions we need for C_1, C_2 from the equations before (5)

```
function phi_xbar=phi_xbar(w1,dt,d,V_0,alpha,beta,
    sigma,rho,r)
u=zeros(d,1);
for j=1:d
     u(j)=w1*(d-j+1)/d;
end
phi_xbar=charfun(u,dt,d,V_0,alpha,beta,sigma,rho,r);
end
```

```
function phi_xbar_xj=phi_xbar_xj(w1,w2,nj,dt,d,V_0,
    alpha,beta,sigma,rho,r)
u=zeros(d,1);
for ii=1:nj
     u(ii)=w2+w1*(d-ii+1)/d;
end
for k=nj+1:d
     u(k)=w1*(d-k+1)/d;
end
phi_xbar_xj=charfun(u,dt,d,V_0,alpha,beta,sigma,rho,r)
    ;

end
```

```
function [Delta_greek,Gamma_greek]=greek_computation(
    dt,T,d,n,S0,K,V_0,alpha,beta,sigma,rho,r)

L=log(K/S0);
gammaval=sum(exp((1:d)*r*dt))/d;
C1=@(w) sum(arrayfun(@(qq) phi_xbar_xj(w,-1i,qq,dt,d,
    V_0,alpha,beta,sigma,rho,r),1:d));
C2=@(w) K*phi_xbar(w,dt,d,V_0,alpha,beta,sigma,rho,r);


integrand1 = @(w) arrayfun(@(ww) imag(exp(-1i*ww*L)./
    ww .* (((1i*ww+1)/d).*C1(ww) - (1i*ww)/S0 .* C2(ww)
    )), w);
integrand2 = @(w) arrayfun(@(ww) imag(exp(-1i*ww*L) .*
     (((1i-ww)/(S0*d)).*C1(ww) + ((1i+ww)/S0^2).*C2(ww)
    )), w);


```

```
mu_W_prime=gammaval/2+(integral(integrand1,-Inf,Inf,'AbsTol', 1e-6, 'RelTol', 1e-6))/(2*pi);
mu_W_second=(integral(integrand2,-Inf,Inf,'AbsTol', 1e-6, 'RelTol', 1e-6))/(2*pi);


f_prime=zeros(n,1);
f_second=zeros(n,1);
for ii=1:n

          [Vpath,Ic]=Glasserman_pathV(d,dt,V_0,alpha,beta,sigma);
          Z=randn(d,1);
          [~,a,kv,s,sigmaquad,b]=conditionalexp(Z,Vpath,Ic,d,S0,K,alpha,beta,sigma,rho,r,dt);
          ai=a(2:end);
          si=s(2:end);
          first_term_f_prime=sum(si.*exp(ai.^2./2).*(normcdf(kv-ai)-normcdf(b-ai)))/d;
          second_term_f_prime=normpdf(kv)/sqrt(sigmaquad)*(sum(si.*exp(ai.*kv))/d-K);
          f_prime(ii)=(first_term_f_prime-second_term_f_prime)/S0;

          first_term_f_second=K^2*normpdf(b)/(sum(ai.*si.*exp(ai.*b))/d);
          second_term_f_second=normpdf(kv)/sigmaquad*((1/d)*sum(ai.*si.*exp(ai.*kv))-2*sqrt(sigmaquad)*(1/d)*sum(si.*exp(ai.*kv))+((1/d)*sum(si.*exp(ai.*kv))-K)*(sqrt(sigmaquad)-kv));
          f_second(ii)=(first_term_f_second+second_term_f_second)/S0^2;


end

Delta_greek=exp(-r*T)*(mean(f_prime)+mu_W_prime);
Gamma_greek=exp(-r*T)*(mean(f_second)+mu_W_second);
end
```

2.5 Numerical results

For the Heston stochastic volatility the authors tested two scenarios. In the first one (HSVM1) the parameters are $\alpha = 6.2, \beta = 0.02, \sigma = 0.6, \rho = -0.7, V(0) =$

0.01, while for the second $\alpha = 0.5, \beta = 0.04, \sigma = 1, \rho = -0.9, V(0) = 0.04$. For both model they used an initial stock price $S(0) = 100$ and a free risk rate $r = 0.05$. The sample size of the simulation is $n = 10^4$. The numerical results are presented for monthly ($\Delta t = 1/12$) and daily ($\Delta t = 1/250$) monitoring time intervals. The maturity was set at $T = 1$ and $T = 2$ years using strike prices $K = 90, 100, 110$. I reported just prices and their 95% error bound (EB). In the following my results with the ones of the paper

Model	T	K	Price	EB
HsV1	1	90	12.49626	9
	1	100	4.52671	3
	1	110	0.44092	2
	2	90	14.57204	18
	2	100	7.15505	10
	2	110	2.19651	5
HSV2	1	90	13.47168	32
	1	100	5.23729	8
	1	110	0.19418	3
	2	90	15.62893	57
	2	100	7.67982	27
	2	110	1.22922	7

Table 6: My results for using $\Delta t = 1/12$

Model	T	K	Price	EB
HSV1	1	90	12.49628	11
		100	4.52671	4
		110	0.44092	2
	2	90	14.57201	18
		100	7.15508	11
		110	2.19651	5
HSV2	1	90	13.47196	36
		100	5.23729	7
		110	0.19417	3
	2	90	15.62883	68
		100	7.67987	21
		110	1.22924	7

Table 7: Paper's results for $\Delta t = 1/12$

Model	T	K	Price	EB
HSV1	1	90	12.24977	8
		100	4.23020	3
		110	0.32236	2
	2	90	14.34318	19
		100	6.90764	10
		110	2.00872	5
HSV2	1	90	13.16767	25
		100	4.92950	7
		110	0.13896	3
	2	90	15.37906	64
		100	7.42753	21
		110	1.06753	6

Table 8: My results for $\Delta t = 1/250$.

Model	T	K	Price	EB
HSV1	1	90	12.24989	11
		100	4.23020	3
		110	0.32234	2
	2	90	14.34324	16
		100	6.90766	8
		110	2.00871	5
HSV2	1	90	13.16788	42
		100	4.92945	6
		110	0.13895	3
	2	90	15.37956	121
		100	7.42742	22
		110	1.06757	7

Table 9: Paper's results for $\Delta t = 1/250$

In the following tables there are the numerical results for the greeks (estimated using the PD method) with $n = 10^4$ and $T = 1$ under monthly and daily monitoring, using HSVM2 parameters

Model	Greek	K	Value
HSV2	Δ	90	0.911021
		100	0.816251
		110	0.099875
	Γ	90	0.004383
		100	0.018968
		110	0.055339

Table 10: My results of Delta and Gamma with $\Delta t = 1/12$.

Model	Greek	K	Value
HSV2	Δ	90	0.911007
		100	0.816271
		110	0.099876
	Γ	90	0.004393
		100	0.018968
		110	0.055340

Table 11: Paper's results Delta and Gamma with $\Delta t = 1/12$.

Model	Greek	K	Value
HSV2	Δ	90	0.909730
		100	0.810338
		110	0.074435
	Γ	90	0.004518
		100	0.020406
		110	0.043162

Table 12: My results of Delta and Gamma with $\Delta t = 1/250$.

Model	Greek	K	Value
HSV2	Δ	90	0.909657
		100	0.810324
		110	0.074434
	Γ	90	0.004499
		100	0.020399
		110	0.043161

Table 13: Paper's results for Delta and Gamma with $\Delta t = 1/250$.

It can be seen that, most of the times, the results are consistent with the paper's ones since the price coming from my code is in the range of the error bound. The time for such simulations goes up with smaller Δt since more control points are needed, and consequently using $T = 2$ requires double the steps. In fact for the last simulation I decided to use the trapezoidal rule instead of the built-in MATLAB function "integral", since the cost of the characteristic function is $O(d^2)$. Based on the comparison between my implementation results and the paper's reference values, the numerical agreement is quite satisfactory across different monitoring frequencies and model parameters. For monthly monitoring ($\Delta t = 1/12$), my price estimates consistently fall within the error bounds reported in the paper, with discrepancies typically within a few basis points. The error bounds (EB) I obtained are comparable in magnitude to those reported, suggesting similar convergence rates for the Monte Carlo simulation with $n = 10^4$ iterations.

The results under the HSV1 parameterization show particularly strong agreement, with prices matching to at least three decimal places across all strike prices and maturities. For instance, at $K = 100$ and $T = 1$, both implementations yield 4.52671, demonstrating the robustness of the variance reduction technique. Under HSV2, which exhibits higher volatility of volatility ($\sigma = 1$ versus $\sigma = 0.6$ for HSV1), the agreement remains excellent despite the more challenging numerical environment. The error bounds are naturally larger for HSV2, reflecting the increased variance in the underlying stochastic process.

When transitioning to daily monitoring ($\Delta t = 1/250$), the computational complexity increases substantially due to the larger number of time steps required. Nevertheless, my implementation maintains consistency with the paper's results, with prices matching to at least four significant figures in most cases.

For the Greek calculations under monthly monitoring with HSV2 parameters, my Delta estimates align closely with the paper's values across all strike levels. At $K = 90$, I obtained 0.911021 compared to the paper's 0.911007, a difference of approximately 0.0014%. The Gamma estimates show similarly tight agreement, with my value of 0.018968 at $K = 100$ matching the paper's result exactly to five decimal places. Under daily monitoring, the agreement persists, with my Delta estimate of 0.909730 at $K = 90$ differing from the paper's 0.909657 by merely 0.008%. The Gamma values also remain consistent, confirming that the pathwise derivative method combined with the variance reduction technique produces stable and accurate sensitivity estimates.