Computational Finance Problem Solving Assignment

As a part of the Master of Statistics (M. Stat.) Second year curriculum

Subhrajyoty Roy (MB1911)
Aakash Kumar Gupta (MB1933)

April 2021

Contents

1 Problems from "Computational Methods in Finance" - Ali Hirsa **2**

2 Extra Problems **19**

1 Problems from "Computational Methods in Finance" - Ali Hirsa

Problem 1. The characteristic function of the jumps governed by CGMY process is given by the expression

$$\mathbb{E}[e^{iuX_t}] = \exp\left[Ct\Gamma(-Y)((M-iu)^Y - M^Y + (G+iu)^Y - G^Y)\right]$$

Utilize the approach that was used in deriving the characteristic function for VGSA to calculate the characteristic function of the log of the underlying process under CGMYSA.

Solution. We start with a general discussion about how to incorporate stochastic arrival into a Levy process. In general, let $X(t)$ be a Levy process, then the corresponding process with stochastic arrival Incorporated is given by $Z(t) = X(Y(t))$ where

$$Y(t) = \int_0^t y(u)du, \qquad dy(t) = \kappa(\eta - y(t)) + \lambda\sqrt{y(t)}dW(t)$$

where $W(t)$ is a Brownian motion with unit volatility. Here, $y(t)$ follows a CIR process. Let, $\phi(u,t) = \mathbb{E}\left[e^{iuY(t)}\right]$ and $\psi_X(u)$ be such that, $\mathbb{E}\left[e^{iuX(t)}\right] = \exp(t\psi_X(u))$, which follows from Levy Khintchine representation of the characteristic function of the Levy process.

$$\begin{aligned}
\mathbb{E}(e^{iuZ(t)}) &= \mathbb{E}(e^{iuX(Y(t))}) \\
&= \mathbb{E}\left[\mathbb{E}(e^{iuX(y)} \mid Y(t) = y)\right] \\
&= \mathbb{E}\left[e^{Y(t)\psi_X(u)}\right] \\
&= \mathbb{E}\left[e^{i(-i\psi_X(u))Y(t)}\right] \\
&= \phi(-i\psi_X(u), t)
\end{aligned}$$

Based on this process, the stock price evolves according to a geometric rule as,

$$S(t) = S(0)e^{(r-q)t+Z(t)}/\mathbb{E}(e^{Z(t)})$$

where $\mathbb{E}(e^{Z(t)}) = \phi(-i\psi_X(-i), t)$. Therefore, the characteristic function of the log of the stock price is given by

$$\mathbb{E}(e^{iu\log S(t))}) = e^{iu(\log(S(0))+(r-q)t)}\frac{\phi(-i\psi_X(u), t)}{\phi(-i\psi_X(-i), t)^{iu}}$$

Now, starting with the characteristic function of CGMY process $X(t)$, we obtain the expression of $\psi_X(u) = C\Gamma(-Y)((M-iu)^Y - M^Y + (G+iu)^Y - G^Y)$. Also, $\psi_X(-i) = C\Gamma(-Y)((M-1)^Y - M^Y + (G+1)^Y - G^Y)$. Therefore, the characteristic function of the log of the underlying stock process under CGMYSA is given by

$$\mathbb{E}(e^{iu\log S(t))}) = e^{iu(\log(S(0))+(r-q)t)}\frac{\phi(-i\psi_X(u),t)}{\phi(-i\psi_X(-i),t)^{iu}}$$

where

$$\begin{aligned}
\psi_X(u) &= C\Gamma(-Y)((M-iu)^Y - M^Y + (G+iu)^Y - G^Y)\\
\psi_X(-i) &= C\Gamma(-Y)((M-1)^Y - M^Y + (G+1)^Y - G^Y)\\
\phi(u,t) &= A(u,t)e^{B(u,t)y(0)}\\
A(u,t) &= \frac{\exp\left(\dfrac{\kappa^2\eta t}{\lambda^2}\right)}{\left(\cosh(\gamma t/2) + \frac{\kappa}{\gamma}\sinh(\gamma t/2)\right)^{2k\eta/\lambda^2}}\\
B(u,t) &= \frac{2iu}{\kappa + \gamma\coth(\gamma t/2)}\\
\gamma &= \sqrt{\kappa^2 - 2\lambda^2 iu}
\end{aligned}$$

Problem 2. Consider the first order PDE

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$$

with the following initial condition $u(x,0) = f(x)$ where $c > 0$ is a constant. Lay out a grid in the (x,t) plane using points $x_j = j\Delta x$ and $t_k = k\Delta t$, where $\Delta x, \Delta t > 0$ are small and j and k are integers. The goal is to calculate $u_{j,k+1} \approx u(x_j, t_{k+1})$ and ensure that this approximation converges to the exact solution as $\Delta x, \Delta t \to 0$. Use explicit finite differences to discretize the PDE and write the difference equation for $u_{j,k+1}$ by

(a) using a forward difference approximation to discretize $\frac{\partial u}{\partial x}$.

(b) using a backward difference approximation to discretize $\frac{\partial u}{\partial x}$.

Now assume the initial condition $f(x) = \cos(\pi x/\Delta x)$ and show that the scheme in part (a) is unstable no matter how small Δx and Δt are. Find the condition that would ensure the scheme in part (b) becomes stable.

Solution. If we apply an explicit finite difference scheme to discretize the PDE, then we use the forward difference to approximate $\frac{\partial u}{\partial t} = \frac{1}{\Delta t}(u_{j,k+1} - u_{j,k}) + O(\Delta t)$.

(a) To obtain a forward difference approximation to discretize $\frac{\partial u}{\partial x}$, we consider

$$\frac{\partial u}{\partial x} = \frac{1}{\Delta x}(u_{j+1,k} - u_{j,k}) + O(\Delta x)$$

Therefore, the given differential equation can be approximately expressed as the difference equation

$$\begin{aligned} & \frac{1}{\Delta t}(u_{j,k+1} - u_{j,k}) + \frac{c}{\Delta x}(u_{j+1,k} - u_{j,k}) = 0 \\ \Rightarrow \quad & u_{j,k+1} - u_{j,k} + \rho u_{j+1,k} - \rho u_{j,k} = 0, \qquad \text{where } \rho = c\frac{\Delta t}{\Delta x} \\ \Rightarrow \quad & u_{j,k+1} = -\rho u_{j+1,k} + (1+\rho)u_{j,k} \end{aligned}$$

where the approximation error is of magnitude $O(\Delta x) + O(\Delta t)$.

Now, to perform the stability analysis, we consider the initial condition as $f(x) = \cos(\pi x/\Delta x)$. This implies,

$$u_{j,0} = f(x_j) = f(j\Delta x) = \cos(\pi j) = \begin{cases} 1 & \text{if } j \text{ is even,} \\ (-1) & \text{if } j \text{ is odd} \end{cases}$$

Putting it back into the difference equation

$$u_{j,k+1} = -\rho u_{j+1,k} + (1+\rho)u_{j,k} \tag{1}$$

we obtain the expression of $u_{j,1}$ as,

$$u_{j,1} = \begin{cases} 1+2\rho & \text{if } j \text{ is even,} \\ -(1+2\rho) & \text{if } j \text{ is odd} \end{cases}$$

To show that

$$u_{j,k} = \begin{cases} (1+2\rho)^k & \text{if } j \text{ is even,} \\ -(1+2\rho)^k & \text{if } j \text{ is odd} \end{cases}$$

we proceed by principle of mathematical induction. Assuming the expression for $u_{j,k}$, to obtain the expression for $u_{j,k+1}$, we note that, if j is even, then $u_{j,k+1} = -\rho[-(1+2\rho)^k] + (1+\rho)(1+2\rho)^k = (1+2\rho)^{(k+1)}$. If j is odd, $u_{j,k+1} = -\rho(1+2\rho)^k + (1+\rho)[-(1+2\rho)^k] = -(1+2\rho)^{(k+1)}$. This completes the inductive step. Therefore, it follows that, $\mid u_{j,k} \mid = \mid 1+2\rho \mid^k \to \infty$ as $k \to \infty$ (since, $\rho > 0$). Hence, the initial value, even if bounded, the solution of the PDE obtained by this scheme turns out to be unbounded. This shows that the scheme is unstable.

(b) To obtain a backward difference approximation to discretize $\frac{\partial u}{\partial x}$, we consider

$$\frac{\partial u}{\partial x} = \frac{1}{\Delta x}(u_{j,k} - u_{j-1,k}) + O(\Delta x)$$

Therefore, we obtain the difference equation as

$$\begin{aligned} & \frac{1}{\Delta t}(u_{j,k+1} - u_{j,k}) + \frac{c}{\Delta x}(u_{j,k} - u_{j-1,k}) = 0 \\ \Rightarrow \quad & u_{j,k+1} - u_{j,k} + \rho u_{j,k} - \rho u_{j-1,k} = 0, \qquad \text{where } \rho = c\frac{\Delta t}{\Delta x} \\ \Rightarrow \quad & u_{j,k+1} = (1-\rho)u_{j,k} + \rho u_{j-1,k} \end{aligned}$$

Denoting

$$\boldsymbol{U}_k = \begin{bmatrix} u_{1,k} \\ u_{2,k} \\ \vdots \\ u_{n,k} \end{bmatrix}$$

we can rewrite the difference equation in matrix form as

$$\boldsymbol{U}_{k+1} = \begin{bmatrix} (1-\rho) & 0 & 0 & \dots & 0 \\ \rho & (1-\rho) & 0 & \dots & 0 \\ 0 & \rho & (1-\rho) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & (1-\rho) \end{bmatrix} \boldsymbol{U}_k$$

Note that, the eigenvalues of the above matrix are all equal to $(1-\rho)$, hence, if we denote the matrix as $\boldsymbol{A}$, then the eigenvalues of $\boldsymbol{A}^k$ are all equal to $(1-\rho)^k$. Therefore, the scheme is stable if

$$\begin{aligned} & \lim_{k\to\infty} (1-\rho)^k = 0 \\ \Rightarrow \quad & |1-\rho| < 1 \\ \Rightarrow \quad & 0 < \rho < 2 \\ \Rightarrow \quad & c(\Delta t) < 2(\Delta x) \end{aligned}$$

Therefore, the condition for stablity of the scheme is

$$\boxed{c(\Delta t) < 2(\Delta x)}$$

Problem 3. A fortiori, the up-and-out call value function $U(S,t)$ solves a backward boundary value problem (BVP), consisting of the backward partial differential equation

$$\frac{\partial U(S,t)}{\partial t} + \frac{\sigma^2(S,t)S^2}{2}\frac{\partial^2 U(S,t)}{\partial S^2} + [r(t) - q(t)]S\frac{\partial U(S,t)}{\partial S} - r(t)U(S,t) = 0$$

subject to the following boundary conditions:

$$\begin{aligned} U(S,T_0) &= (S-K_0)^+, \quad S \in [0,B] \\ \lim_{S\downarrow 0} U(S,t) &= 0, \quad t \in [0,T_0] \\ \lim_{S\uparrow B} U(S,t) &= 0, \quad t \in [0,T_0] \end{aligned}$$

Solve the PDE numerically for the following set of parameters: spot price $S_0 = \$100$, strike price $K = \{105, 115\}$, up barrier $B = 125$, risk-free interest rate $r = 4.75\%$, dividend rate $q = 1.75\%$, maturity $T = 1$ year, and volatility $\sigma = \{15\%, 30\%, 50\%\}$ using

- Uniform mesh points on S
- Uniform mesh points on ξ where

$$\xi = \frac{\sinh^{-1}((S-B)/\alpha) - c_2}{c_1 - c_2}$$

where

$$c_1 = \sinh^{-1}\left((S_{\max} - B)/\alpha\right), \qquad c_2 = \sinh^{-1}\left((S_{\min} - B)/\alpha\right), \qquad \alpha = \frac{(S_{\max} - S_{\min})}{20}$$

Compare approximated values (premium and delta) with the closed form values.

Solution. 1. **Uniform Mesh points:** Let us consider uniform mesh points on S for the discretization. Define $\tau = T_0 - t$, then the backward partial differential equation converts to the forward partial differential equation as

$$-\frac{\partial U(S,\tau)}{\partial \tau} + \frac{\sigma^2(S,\tau)S^2}{2}\frac{\partial^2 U(S,\tau)}{\partial S^2} + [r(\tau) - q(\tau)]S\frac{\partial U(S,\tau)}{\partial S} - r(\tau)U(S,\tau) = 0 \qquad (2)$$

Now, let $\tau_k = (k-1)\Delta\tau$ and $S_j = S_{\min} + (j-1)\Delta S$ for $j = 1, 2, \ldots; k = 1, 2, \ldots$. Consequently, the discretized approximation of $U(S_j, \tau_k), \sigma(S_j, \tau_k)$ is denoted as $U_{j,k}$ and $\sigma_{j,k}$. With these notations, the difference equation for implicit scheme turns out to be as follows,

$$
\begin{aligned}
&\frac{U_{j,k} - U_{j,k+1}}{\Delta\tau} + \frac{\sigma_{j,k+1}^2 S_j^2}{2} \frac{U_{j-1,k+1} - 2U_{j,k+1} + U_{j+1,k+1}}{(\Delta S)^2} + \\
&\qquad (r-q)S_j \frac{U_{j+1,k+1} - U_{j-1,k+1}}{2\Delta S} - rU_{j,k+1} = 0 \\
\Rightarrow \quad & U_{j,k+1}\left(\frac{1}{\Delta\tau} + \frac{\sigma_{j,k+1}^2 S_j^2}{(\Delta S)^2} + r\right) + U_{j-1,k+1}\left(-\frac{\sigma_{j,k+1}^2 S_j^2}{2(\Delta S)^2} + \frac{(r-q)S_j}{2\Delta S}\right) + \\
&\qquad U_{j+1,k+1}\left(-\frac{\sigma_{j,k+1}^2 S_j^2}{2(\Delta S)^2} - \frac{(r-q)S_j}{2\Delta S}\right) = \frac{U_{j,k}}{\Delta\tau} \\
\Rightarrow \quad & l_{j,k+1}U_{j-1,k+1} + d_{j,k+1}U_{j,k+1} + u_{j,k+1}U_{j+1,k+1} = U_{j,k}
\end{aligned}
$$

where $l_{j,k+1}, d_{j,k+1}$ and $u_{j,k+1}$ are the quantities it is replacing. Therefore, if we denote $\boldsymbol{U}_k = (U_{2,k}, \ldots U_{N,k})^\intercal$, then we can solve the PDE numerically by solving the following system of equation for each timestep,

$$
\begin{bmatrix}
d_{1,k+1} & u_{1,k+1} & 0 & 0 & \ldots \\
l_{2,k+1} & d_{2,k+1} & u_{2,k+1} & 0 & \ldots \\
0 & l_{3,k+1} & d_{3,k+1} & u_{3,k+1} & \ldots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{bmatrix} \boldsymbol{U}_{k+1} = \boldsymbol{U}_k
$$

with $\boldsymbol{U}_0 = U(S_j, T_0) = (S_j - K)^+$.

The following R code implements the algorithm.

```r
# uniform mesh grid on S
upoutPriceType1 <- function(K, B, sigma, param_list) {
    delS <- param_list[["delS"]]
    delT <- param_list[["delT"]]
    S0 <- param_list[["S0"]]
    r <- param_list[["r"]]
    q <- param_list[["q"]]

    Smin <- 0
    Smax <- B
    N <- (Smax - Smin)/delS
    S <- Smin + 1:(N-1) * delS
    U <- pmax((S - K), 0)

    maturity <- 1
    nT <- maturity / delT
```

```
    for (i in 1:nT) {
        A_mat <- matrix(0, nrow = (N-1), ncol = (N-1))

        # fill up the rows
        for (j in 1:(N-1)) {
            if (j > 1) {
                A_mat[j,(j-1)] <- (-sigma^2*S[j]^2/(2 * delS^2) + (r-q)*S[j]/(2 * delS))*delT
            }
            if (j < (N-1)) {
                A_mat[j,(j+1)] <- (-sigma^2*S[j]^2/(2 * delS^2) - (r-q)*S[j]/(2 * delS))*delT
            }
            A_mat[j,j] <- (1/delT + sigma^2*S[j]^2/delS^2 + r)*delT
        }

        # update U vector by implicit scheme
        U <- solve(A_mat, U)
    }

    # the price corresponds to U(S0, 0)
    S0index <- (S0 - Smin)/delS
    return(U[S0index])
}
```

2. **Uniform mesh points on ξ:** Note that, given the parametrization of ξ, we have $S(\xi) = B + \alpha \sinh(c_1\xi + c_2(1-\xi))$. Letting $\overline{U}(\xi,\tau) = U(S,t)$, the chain rule of differentiation yields,

$$\begin{aligned}
\frac{\partial U}{\partial S} &= \frac{\partial \overline{U}}{\partial \xi}\frac{1}{S'(\xi)} \\
\frac{\partial^2 U}{\partial^2 S} &= \frac{\partial^2 \overline{U}}{\partial \xi^2}\frac{1}{(S'(\xi))^2} - \frac{\partial \overline{U}}{\partial \xi}\frac{S''(\xi)}{(S'(\xi))^3} \\
\frac{\partial U}{\partial t} &= -\frac{\partial \overline{U}}{\partial \tau}
\end{aligned}$$

Now, we know that,

$$\begin{aligned}
S'(\xi) &= \alpha(c_1 - c_2)\cosh(c_1\xi + c_2(1-\xi)) \\
S''(\xi) &= \alpha(c_1 - c_2)^2\sinh(c_1\xi + c_2(1-\xi))
\end{aligned}$$

Putting these back into the PDE in question and applying the change of coordinate using the chain rule of differentiation, we obtain the new PDE in transformed coordinate as

$$
\begin{aligned}
& -\frac{\partial \overline{U}}{\partial \tau} + \frac{\sigma^2(\xi,\tau)S^2}{2}\left(\frac{\partial^2 \overline{U}}{\partial \xi^2}\frac{1}{(S'(\xi))^2} - \frac{\partial \overline{U}}{\partial \xi}\frac{S''(\xi)}{(S'(\xi))^3}\right) + (r(\tau)-q(\tau))S\frac{\partial \overline{U}}{\partial \xi}\frac{1}{S'(\xi)} - r(t)U = 0 \\
\Rightarrow \quad & -\frac{\partial \overline{U}}{\partial \tau} + \frac{\partial \overline{U}}{\partial \xi}\left((r-q)\frac{S}{S'(\xi)} - \frac{\sigma^2 S^2 S''(\xi)}{2(S'(\xi))^3}\right) + \frac{\partial^2 \overline{U}}{\partial \xi^2}\left(\frac{\sigma^2 S^2}{2(S'(\xi))^2}\right) - r(\tau)\overline{U}(\tau) = 0
\end{aligned}
$$

Now, applying discretization to ξ-grid, similar to before, we have the difference equation,

$$
\begin{aligned}
& \frac{\overline{U}_{j,k} - \overline{U}_{j,k+1}}{\Delta\tau} + \frac{\overline{U}_{j+1,k+1} - \overline{U}_{j-1,k+1}}{2\Delta\xi}\left((r-q)\frac{S(\xi_j)}{S'(\xi_j)} - \frac{\sigma^2_{j,k+1}S(\xi_j)^2 S''(\xi_j)}{2(S'(\xi_j))^3}\right) + \\
& \qquad \frac{\overline{U}_{j-1,k+1} - 2\overline{U}_{j,k+1} + \overline{U}_{j+1,k+1}}{(\Delta\xi)^2}\frac{\sigma^2_{j,k+1}S(\xi_j)^2}{2(S'(\xi_j))^2} - r\overline{U}_{j,k+1} = 0 \\
\Rightarrow \quad & \frac{\overline{U}_{j,k} - \overline{U}_{j,k+1}}{\Delta\tau} + \frac{\overline{U}_{j+1,k+1} - \overline{U}_{j-1,k+1}}{2\Delta\xi}\alpha_{j,k} + \frac{\overline{U}_{j-1,k+1} - 2\overline{U}_{j,k+1} + \overline{U}_{j+1,k+1}}{(\Delta\xi)^2}\beta_{j,k} - r\overline{U}_{j,k+1} = 0 \\
\Rightarrow \quad & \frac{\overline{U}_{j,k}}{\Delta\tau} = \overline{U}_{j-1,k+1}\left(\frac{\alpha_{j,k}}{2\Delta\xi} - \frac{\beta_{j,k}}{(\Delta\xi)^2}\right) + \overline{U}_{j,k+1}\left(r + \frac{1}{\Delta\tau} + \frac{2\beta_{j,k}}{(\Delta\xi)^2}\right) + \overline{U}_{j+1,k+1}\left(-\frac{\alpha_{j,k}}{2\Delta\xi} - \frac{\beta_{j,k}}{(\Delta\xi)^2}\right)
\end{aligned}
$$

where $\xi_j = (j-1)\Delta\xi$, such that $\xi_0 = 0, \xi_N = 1$, since $S(\xi = 0) = S_{\min}$ and $S(\xi = 1) = S_{\max}$. Again, we can solve the PDE numerically by solving the above system of equations as each timepoint τ_k.

The following R code implements this explicit discretization algorithm with uniform mesh grid on ξ-plane.

```r
# uniform mesh grid on xi
upoutPriceType2 <- function(K, B, sigma, param_list) {
    delS <- param_list[["delS"]]
    delxi <- param_list[["delxi"]]
    delT <- param_list[["delT"]]
    S0 <- param_list[["S0"]]
    r <- param_list[["r"]]
    q <- param_list[["q"]]

    Smin <- 0
    Smax <- B
    alpha <- (Smax - Smin)/20
    c1 <- asinh((Smax - B)/alpha)
```

```
    c2 <- asinh((Smin - B)/alpha)
    N <- 1/delxi
    xi <- 1:(N-1) * delxi  # uniform grid on xi
    S <- B + alpha * sinh(c1 * xi + c2 * (1-xi))
    Sprime <- alpha * (c1-c2) * cosh(c1 * xi + c2 * (1-xi))  # dS/dxi
    Sdprime <- alpha * (c1 - c2)^2 * sinh(c1 * xi + c2 * (1-xi))  # d2S/
   dxi2
    U <- pmax((S - K), 0)

    maturity <- 1
    nT <- maturity / delT
    for (i in 1:nT) {
        A_mat <- matrix(0, nrow = (N-1), ncol = (N-1))
        # fill up the rows
        for (j in 1:(N-1)) {
            alphajk <- (r-q)*S[j]/Sprime[j] - (sigma^2*S[j]^2*Sdprime[j]/(2
    * Sprime[j]^3))
            betajk <- S[j]^2 * sigma^2 / (2 * Sprime[j]^2)

            if (j > 1) {
                A_mat[j,(j-1)] <- delT * (alphajk/(2 * delxi) - betajk /
   delxi^2 )
            }
            if (j < (N-1)) {
                A_mat[j,(j+1)] <- delT * (-alphajk/(2*delxi) - betajk /
   delxi^2)
            }
            A_mat[j,j] <- delT * (r + 1/delT + 2 * betajk/delxi^2)
        }

        # update U vector by implicit scheme
        U <- solve(A_mat, U)
    }

    # the price corresponds to U(S0, 0)
    S0index <- round((asinh((S0-B)/alpha) - c2)/((c1 - c2) * delxi))
    return(U[S0index])
}
```

Since the strike price K is less than the barrier B, the closed form solution to the price of the

up-and-out barrier option is $c_{uo} = A - B + C - D$, where

$$
\begin{aligned}
A &= S_0 e^{(b-r)T}\Phi(x_1) - Ke^{-rT}\Phi(x_1 - \sigma\sqrt{T}) \\
B &= S_0 e^{(b-r)T}\Phi(x_2) - Ke^{-rT}\Phi(x_2 - \sigma\sqrt{T}) \\
C &= S_0 e^{(b-r)T}(B/S_0)^{2(\mu+1)}\Phi(-y_1) - Ke^{-rT}(B/S_0)^{2\mu}\Phi(-y_1 + \sigma\sqrt{T}) \\
D &= S_0 e^{(b-r)T}(B/S_0)^{2(\mu+1)}\Phi(-y_2) - Ke^{-rT}(B/S_0)^{2\mu}\Phi(-y_2 + \sigma\sqrt{T})
\end{aligned}
$$

where

$$
\begin{aligned}
x_1 &= \frac{\log(S_0/K)}{\sigma\sqrt{T}} + (1+2\mu)\sigma\sqrt{T} \\
x_2 &= \frac{\log(S_0/B)}{\sigma\sqrt{T}} + (1+2\mu)\sigma\sqrt{T} \\
y_1 &= \frac{\log(B^2/S_0K)}{\sigma\sqrt{T}} + (1+2\mu)\sigma\sqrt{T} \\
y_2 &= \frac{\log(B/S_0)}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T} \\
\mu &\frac{b - \sigma^2/2}{\sigma^2} \\
\lambda &= \sqrt{\mu^2 + 2r/\sigma^2} \\
b &= (r - q)
\end{aligned}
$$

The following R function calculates the closed form price of the up-and-out barrier option.

```r
# closed form price
upoutPriceClosed <- function(K, B, sigma, param_list) {
    S0 <- param_list[["S0"]]
    r <- param_list[["r"]]
    q <- param_list[["q"]]
    maturity <- 1
    sigmat <- sigma * sqrt(maturity)

    b <- (r - q)
    mu <- (b - sigma^2/2)/sigma^2
    lambda <- sqrt(mu^2 + 2*r/sigma^2)
    x1 <- log(S0/K)/sigmat + (1 + 2*mu)*sigmat
    x2 <- log(S0/B)/sigmat + (1+2*mu)*sigmat
    y1 <- log(B^2/(S0 * K))/sigmat + (1 + 2*mu)*sigmat
    y2 <- log(B/S0)/sigmat + (1 + 2*mu)*sigmat

```

```
    A <- S0*exp((b-r)*maturity)*pnorm(x1) - K*exp(-r*maturity)*pnorm(x1 -
   sigmat)
    B_new <- S0*exp((b-r)*maturity)*pnorm(x2) - K*exp(-r*maturity)*pnorm(x2
    - sigmat)
    C <- S0*exp((b-r)*maturity)*((B/S0)^(2*(mu+1)))*pnorm(-y1) - K*exp(-r*
   maturity)*((B/S0)^(2*mu))*pnorm(-y1 + sigmat)
    D <- S0*exp((b-r)*maturity)*((B/S0)^(2*(mu+1)))*pnorm(-y2) - K*exp(-r*
   maturity)*((B/S0)^(2*mu))*pnorm(-y2 + sigmat)

    price <- abs(A - B_new + C - D)

    return(price)
}
```

Similar to above, we also obtain approximate prices from the explicit and Crank Nicolsen scheme as well. The closed form solution of the premium for the given parameters in question along with their approximate values computed by 3 different schemes are described in the following table.

Strike Price (K)	Volatility (σ)	Price of up-and-out-barrier option						
		Closed form	Uniform ξ-grid			Uniform S-grid		
			Explicit	Implicit	Crank Nicolson	Explicit	Implicit	Crank Nicolson
105	15%	1.5877	14310.24	1.5904	1.5883	16913.19	1.7856	1.7214
105	30%	0.4143	8532.44	0.4119	0.4131	9123.29	0.4212	0.4207
105	50%	0.08842	8112.56	0.1033	0.09198	9100.43	0.1039	0.09204
115	15%	0.1527	8239.55	0.1603	0.1513.94	8545.78	0.2274	0.1772
115	30%	0.04182	7635.21	0.03948	0.04156	8040.78	0.05012	0.04785
115	50%	0.01776	6623.49	0.00964	0.01193	7122.64	0.01202	0.01443

As seen from the table, the A_{explicit} matrix in this case turns out to have many eigenvalues more than 1, thus, the explicit scheme is not stable. Therefore, the final solution do not remain bounded, and provide unreasonable approximation of the option premium. In comparison, both implicit and Crank-Nicolson scheme provides an approximation very close to the original closed form solution to the option price, with an error of at most 0.2 approximately. The uniform grid on ξ-plane yields a closer approximation than uniform grid on S-plane.

Problem 4. Let $b(t;\theta,\nu) = \theta t + \sigma W(t)$ be a Brownian motion with constant drift rate θ and volatility σ, where $W(t)$ is a standard Brownian motion. Denote by $\gamma(t;\nu)$ the gamma process with independent gamma increments of mean h and variance νh over non-overlapping intervals of length h.
The three parameter VG process, $X(t;\sigma,\theta,\nu)$ is defined by $X(t;\sigma,\theta,\nu) = b(\gamma(t;\nu);\theta,\sigma)$. The VG risk-neutral process for the stock price is given by

$$S(t) = S(0)e^{(r-q)t+X(t)+wt}$$

where the normalization factor e^{wt} ensures that the $\mathbb{E}_0(S(t)) = S(0)e^{(r-q)t}$. Here,

$$w = \frac{1}{\nu}\ln(1-\sigma^2\nu/2-\theta\nu)$$

By the definition of risk neutrality, the price of a European put option with strike K and maturity T is

$$p(S(0);K,t) = e^{-rT}\mathbb{E}_0((K-S(T))^+)$$

For the following parameters: spot price $S_0 = \$100$, strike price $K = 105$, maturity $T = 1$ year, risk-free interest rate $r = 4.75\%$, continuous dividend rate $q = 1.25\%$, $\sigma = 25\%$, $\nu = 0.5, \theta = -0.3$, price a European put option via

1. FFT technique
2. simulation

and compare.

Solution. Since, the VG risk-neutral process for the stock price is given by $S(t) = S(0)\exp((r-q)t + X(t) + wt)$, it follows that the characteristic function of the log of the stock price is given as,

$$\begin{aligned}\psi(u,t) = \mathbb{E}(e^{iu\log(S(t))}) &= \mathbb{E}(e^{iu(\log(S(0))+(r-q)t+X(t)+wt)}) \\ &= \exp(iu(\log(S(0)) + (r-q)t + wt))\mathbb{E}(e^{iuX(t)}) \\ &= \exp(iu(\log(S(0)) + (r-q)t + wt))\phi(u,t)\end{aligned}$$

where $\phi(u,t)$ is the characteristic function of the variance gamma process evaluated at u. Therefore, the algorithm for pricing the European put option when the stock price follows a Variance Gamma

(VG) process, based on FFT based techniques is as follows:

1. Choose $\alpha = -1.5, N = 2^{14}, \eta = 0.1$. Fix $\lambda = 2\pi/N\eta$.

2. Let $\nu_j = (j-1)\eta$ for $j = 1, 2, \ldots N$.

3. Create the vector

$$\boldsymbol{x} = \begin{bmatrix} (\eta/2)\dfrac{e^{-rT}}{(\alpha+i\nu_1)(\alpha+1+i\nu_1)}\exp(-i\nu_1(\log(K)-\lambda N/2))\psi(\nu_1 - i(\alpha+1)) \\ \eta\dfrac{e^{-rT}}{(\alpha+i\nu_2)(\alpha+1+i\nu_2)}\exp(-i\nu_2(\log(K)-\lambda N/2))\psi(\nu_2 - i(\alpha+1)) \\ \vdots \\ \eta\dfrac{e^{-rT}}{(\alpha+i\nu_N)(\alpha+1+i\nu_N)}\exp(-i\nu_N(\log(K)-\lambda N/2))\psi(\nu_N - i(\alpha+1)) \end{bmatrix}$$

4. Obtain $\boldsymbol{y} = \text{FFT}(\boldsymbol{x})$.

5. Let $y_{N/2}$ be the $N/2$-th coordinate of $\boldsymbol{y}$.

6. The final price of the option is $\dfrac{\exp(-\alpha\log(K))}{2\pi}\Re(y_{N/2})$.

The following is the R code that implements this algorithm for the specified strike price, maturity and the other parameters given in the question.

```
FFT_put_price <- function(K, maturity, alpha, param_list) {
    eta <- param_list[["eta"]]
    N <- param_list[["N"]]
    lambda <- (2 * pi)/(N * eta)
    vj <- eta * (1:N - 1)

    # VG process char function of log(S(t))
    char_VG <- function(u, t, param_list) {
        theta <- param_list[["theta"]]
        nu <- param_list[["nu"]]
        sigma <- param_list[["sigma"]]
        S0 <- param_list[["S0"]]
        r <- param_list[["r"]]
        q <- param_list[["q"]]

        phi_u <- (1 - (0+1i)*u*theta*nu + sigma^2 * u^2 * nu/2)
        phi_u <- (1/phi_u)^(t/nu)
        phi_i <- as.complex(1 - theta * nu - sigma^2 * nu/2)
```

```
        phi_i <- (1 / phi_i)^(t/nu)
        ch <- exp((0+1i)*u*(log(S0) + (r-q)*t )) * phi_u / phi_i^((0 + 1i)*u)

        return(ch)
    }

    xvec <- rep(eta, N)
    xvec[1] <- xvec[1]/2
    xvec <- xvec * exp(-param_list[["r"]] * maturity) / ((alpha + (0+1i)*vj) * (
   alpha + (0+1i)*vj + 1))
    xvec <- xvec * exp(-(0+1i)*(log(K) - lambda * N/2)*vj)
    xvec <- xvec * char_VG(vj - (alpha + 1)*(0+1i), maturity, param_list)

    yvec <- fft(xvec)
    p_seq <- log(K) - lambda*N/2 + (0:(N-1))*lambda
    price <- Re(yvec) * exp(-alpha*p_seq)/pi

    price <- price[N/2]

    return(price)
}

param_list <- list("S0" = 100, "r" = 0.0475, "q" = 0.0125, "sigma" = 0.25, "nu"
   = 0.5, "theta" = -0.3, "eta" = 0.1, "N" = 2^14)
FFT_put_price(K = 105, maturity = 1, alpha = -1.5, param_list = param_list)

```

To obtain the price of the put option using the Monte Carlo simulation method, we note that since the European option is path independent, it is enough to simulate the behaviour of the stock price at maturity, instead of simulating the whole path. Therefore, we employ the following algorithm.

1. Fix the total number of resamples as $B \approx 10^6$.
2. For each resample indexed by b for $b = 1, 2, \ldots B$, perform the following.
   (a) Simulate a gamma random variable g_b following the gamma distribution with shape parameter T/ν and scale parameter ν, so that the expectation is T and variance is $T\nu$.
   (b) Simulate a brownian motion $W(g_b)$ conditioned on the sample g. For this, we simulate a random variable $W(g_b)$ following a normal distribution with mean 0 and variance g.
   (c) Define $X_b(T) = \theta g_b + \sigma W(g_b)$.

(d) Define $S_b(T) = S(0)\exp((r-q)T + wT + X_b(T))$.

(e) Calculate price of the put option as $P_b(T) = (K - S_b(T))^+$.

3. Finally, estimate the risk neutral price of the put option as $P(T) = \frac{1}{B}\sum_{b=1}^{B} P_b(T)$.

The following is the R code that implements the above algorithm for pricing the European put option using Monte Carlo simulation.

```
simulate_put_price <- function(K, maturity, param_list, seed = 1234) {
    B <- 1e6  # number of resamples to take the average

    S0 <- param_list[["S0"]]
    r <- param_list[["r"]]
    q <- param_list[["q"]]
    sigma <- param_list[["sigma"]]
    nu <- param_list[["nu"]]
    theta <- param_list[["theta"]]

    set.seed(seed)
    gamma_g <- rgamma(B, shape = maturity/nu, scale = nu)
    Wt <- rnorm(B, mean = 0, sd = sqrt(gamma_g))
    Xt <- theta * gamma_g + sigma * Wt
    w <- (1/nu) * log(1 - sigma^2 * nu/2 - theta * nu)
    St <- S0 * exp((r - q) * maturity + Xt + w*maturity)

    values <- ifelse(K > St, (K - St), 0)
    price <- mean(values) * exp(-r*maturity)
    return(price)
}
param_list <- list("S0" = 100, "r" = 0.0475, "q" = 0.0125, "sigma" = 0.25, "nu"
   = 0.5, "theta" = -0.3)
simulate_put_price(K = 105, maturity = 1, param_list = param_list, seed = 1911)
```

The price of the put option obtained from the simulation method with a seed of 1911 **is found to be** \$11.95562**.** In order to compare it with the approximate price obtain via FFT algorithm, we consider different values of the parameters η, N and α to see the effect of its choice in the price determination. The results are summarized in the following table.

We see that the price is fairly robust to the effect of α. As η decreases, the price of the option actually decreases rapidly, but increasing N has an opposite effect that pushes the approximation close to the original price of the option. With the choice of $\eta = 0.2, N = 2^{12}$ and $\alpha = -2$ yields the

	$\eta = 0.2$			$\eta = 0.1$			$\eta = 0.05$		
α	$N = 2^8$	$N = 2^{10}$	$N = 2^{12}$	$N = 2^8$	$N = 2^{10}$	$N = 2^{12}$	$N = 2^8$	$N = 2^{10}$	$N = 2^{12}$
-1.25	7.0854	10.5261	11.5978	4.0823	9.2011	11.1918	1.3035	7.5011	10.4884
-1.5	7.0511	10.4885	11.5591	4.0823	9.2010	11.1918	1.3048	7.5011	10.4884
-2	7.0510	10.4884	11.5592	4.0822	9.2010	11.1918	1.3068	7.5011	10.4884

Table 1: Price of the put option in question obtained via FFT technique for various parameters

best approximation to the put price, namely equal to \$11.5592 which is very close to the true value of the put price obtained by simulation i.e. \$11.9556.

2 Extra Problems

Problem 5. The polar method is an improvement of the Box-Muller method. In this problem, we show that the implied transformation of uniformly distributed random variables on the unit disc yields uniformly distributed random variables on the unit square. Let $B = \{x \in \mathbb{R}^2 : x_1^2 + x_2^2 < 1\}$ denote the unit disc and $S = (0,1) \times (0,1)$ be the unit square. Consider a random variable $V = (V_1, V_2)$ which is uniformly distributed on the unit disc with density

$$f(x) = \begin{cases} \pi^{-1}, & x \in B \\ 0, & \text{else} \end{cases}$$

Let $\phi : B \to S$ be given by

$$\phi(v_1, v_2) = \begin{bmatrix} v_1^2 + v_2^2 \\ H(v_1, v_2) \end{bmatrix}, \qquad (v_1, v_2) \in B$$

with the function H as,

$$H(y_1, y_2) = \begin{cases} \frac{1}{2\pi} \arctan(y_2/y_1), & y_1 > 0, y_2 > 0 \\ \frac{1}{2\pi}(\arctan(y_2/y_1) + \pi), & y_1 < 0 \\ \frac{1}{2\pi}(\arctan(y_2/y_1) + 2\pi), & y_1 > 0, y_2 < 0 \\ 3/4 & y_1 = 0, y_2 < 0, \\ 1/4 & y_1 = 0, y_2 > 0, \\ 1/2 & y_2 = 0 \end{cases}$$

Show that

$$\begin{bmatrix} W_1 \\ W_2 \end{bmatrix} = \phi(V_1, V_2)$$

is uniformly distributed on the unit square S.

Solution. We consider the joint probability that,

$$\begin{aligned} \mathbb{P}(W_1 < u, W_2 < v) &= \mathbb{P}(\phi_1(V_1, V_2) < u, \phi_2(V_1, V_2) < v) \\ &= \mathbb{P}(V_1^2 + V_2^2 < u, H(V_1, V_2) < v) \end{aligned}$$

Let us now consider some sub-cases. Clearly, note that, $\phi : B \to S$, hence, it is meaningful to consider the cases where both $u, v \in [0, 1]$ only.

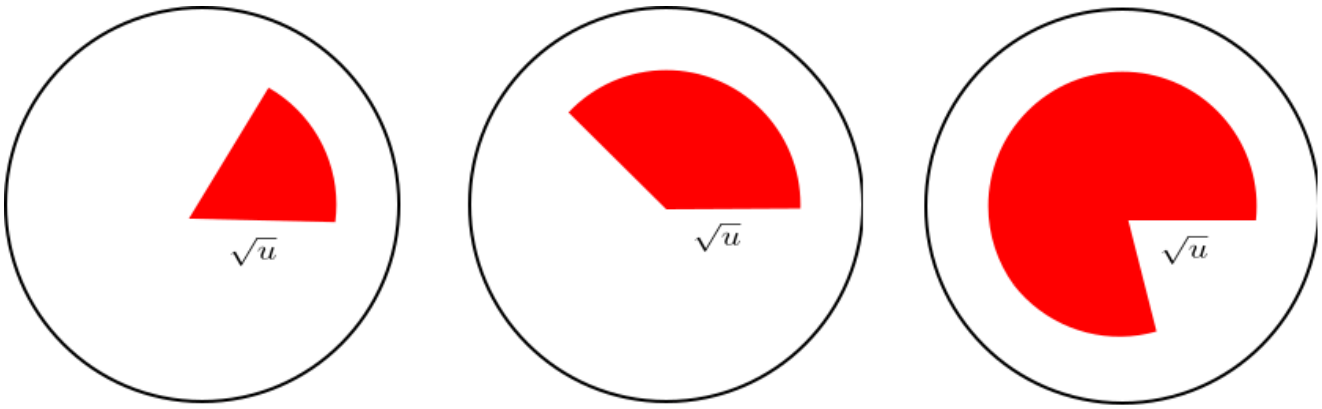

Figure 1: Space where (V_1, V_2) lies for different subcases; [Left] when $0 \leq v \leq 1/4$, [Middle] when $1/4 \leq v \leq 3/4$, [Right] when $3/4 \leq v \leq 1$

1. **Case 1:** $0 \leq v \leq 1/4$. Therefore, $H(V_1, V_2) < v$ simply means, $V_1 > 0, V_2 > 0$, and $\frac{1}{2\pi} \arctan(V_2/V_1) < v$, i.e. $\arctan(V_2/V_1) < 2\pi v$.

   Therefore, the event $\{V_1^2 + V_2^2 < u, \arctan(V_2/V_1) < 2\pi v\}$ implies that the point (V_1, V_2) lies within the circle of radius $\sqrt{u}$ and between the lines $y = 0$ and $y = \tan(2\pi v)x$ (see Figure 1[Left]). Hence,

$$\mathbb{P}(V_1^2 + V_2^2 < u, H(V_1, V_2) < v) = \frac{1}{\pi} \times \pi(\sqrt{u})^2 v = uv$$

2. **Case 2:** $1/4 \leq v \leq 3/4$. Therefore, $H(V_1, V_2) < v$ simply means, either the point (V_1, V_2) lie on the first quadrant, or $V_1 < 0, V_2 > 0$ such that $\frac{1}{2\pi}(\arctan(V_2/V_1) + \pi) < v$, i.e. $\arctan(\mid V_2 \mid / \mid V_1 \mid) > 2\pi(1/2 - v)$. This means, the event $\{V_1^2 + V_2^2 < u, \arctan(V_2/V_1) < 2\pi v\}$ implies that the point (V_1, V_2) lies within the circle of radius $\sqrt{u}$ and between the lines $y = 0$ and $y = \tan(2\pi v)x$ (see Figure 1[Middle]). Hence,

$$\mathbb{P}(V_1^2 + V_2^2 < u, H(V_1, V_2) < v) = \frac{1}{\pi} \times \pi(\sqrt{u})^2 v = uv$$

3. **Case 3:** $3/4 \leq v \leq 1$. Therefore, $H(V_1, V_2) < v$ simply means, either the point (V_1, V_2) lie on 1st, 2nd or 3rd quadrant or $V_1 > 0, V_2 < 0$ such that $\frac{1}{2\pi}(\arctan(V_2/V_1) + 2\pi) < v$, i.e. $\arctan(\mid V_2 \mid / \mid V_1 \mid) > 2\pi(1 - v)$. This means, the event $\{V_1^2 + V_2^2 < u, \arctan(V_2/V_1) < 2\pi v\}$

implies that the point (V_1, V_2) lies within the circle of radius $\sqrt{u}$ and between the lines $y = 0$ and $y = \tan(2\pi v)x$ (see Figure 1[Right]). Hence,

$$\mathbb{P}(V_1^2 + V_2^2 < u, H(V_1, V_2) < v) = \frac{1}{\pi} \times \pi(\sqrt{u})^2 v = uv$$

Therefore, in all such cases, we have

$$\begin{aligned}
\mathbb{P}(W_1 < u, W_2 < v) &= \mathbb{P}(V_1^2 + V_2^2 < u, H(V_1, V_2) < v) \\
&= uv \\
&= \mathbb{P}(W_1 < u)\mathbb{P}(W_2 < v)
\end{aligned}$$

where W_1 and W_2 are independently distributed uniform $(0, 1)$ random variable. This proves that (W_1, W_2) is uniformly distributed on the unit square S.