בס"ד

שיטות נומריות למתמטיקה פיננסית, פרופ' ג'רמי שיף, 88-636

מבחן מועד א', סמסטר א' תשע"ב

זמן המבחן: שעתיים וחצי.
מותר להשתמש בכל חומר עזר ובמחשב כיס.
יש לענות על 4 מתוך 5 השאלות. שאלה 5 היא חובה. ניקוד כל השאלות שווה.
יש לנמק היטב כל תשובה.

1. המשתנים הסטוכסטיים S, V מקיימים את המשוואות הדפרנציאליות הסטוכסטיות

$$\begin{aligned} dS &= S(rdt + \sqrt{V}dW_1) \\ dV &= a(V_0 - V)dt + b\sqrt{V}(\rho\, dW_1 + \sqrt{1-\rho^2}dW_2) \end{aligned}$$

כאשר r, a, V_0, b, ρ הם קבועים חיוביים עם $\rho < 1$ ו- W_1, W_2 הם תהליכי וינר בלתי תלויים.

(א) כתוב את שיטת אויילר-מרוימה למשוואות אלה.

(ב) איך ניתן להשתמש בשיטת מילשטיין לשפר את הקירוב למשוואה השנייה ?

(ג) הסבר למה במקרה $a = b = 0$ המשוואה הראשונה הופכת למשוואת GBM הסט-נדרטית. איך היית מציע לנצל עובדה זו לשפר את הקירוב למשוואה הראשונה ? (גם במקרה ש- a, b אינם אפס.)

(א)

$$\begin{aligned} S_{n+1} &= S_n + S_n(rh + \sqrt{hV_n}Z_{1,n}) \\ V_{n+1} &= V_n + ah(V_0 - V_n) + b\sqrt{hV_n}(\rho\ Z_{1,n} + \sqrt{1-\rho^2}Z_{2,n}) \end{aligned}$$

כאן $Z_{1,n}, Z_{2,n}$ הם משתנים מקריים נורמליים סטמדרטיים בלתי תלויים

(ב) הצירוף הלינארי $W = \rho W_1 + \sqrt{1-\rho^2}W_2$ הוא תהליך וינר סטנדרטי וניתן לכתוב את המשוואה השנייה

$$dV = a(V_0 - V)dt + b\sqrt{V}dW$$

למשוואה

$$dX = A(X)dt + B(X)dW$$

שיטת מילשטיין היא

$$X_{n+1} = X_n + A(X_n)h + B(X_n)\sqrt{h}Z_n + \frac{1}{2}hBB'(X_n)(Z_n^2 - 1)$$

למשוואה שלנו $B(V) = b\sqrt{V}$ ולכן $BB'(V) = \frac{1}{2}b^2$. ולכן שיטת מילשטיין היא

$$V_{n+1} = V_n + a(V_0 - V_n)h + b\sqrt{hV_n}Z_n + \frac{1}{4}hb^2(Z_n^2 - 1)$$

כאשר $Z_n = \rho\ Z_{1,n} + \sqrt{1-\rho^2}Z_{2,n}$.

(ג) במקרה $a=b=0$ המשוואה השנייה היא $dV=0$ ולכן V הוא קבוע והמשוואה הראשונה היא GBM הסטדנרטי עם נדיפות $\sigma=\sqrt{V}$. במקרה הזה פותרים את המשוואה הראשונה על ידי הנוסחה

$$S_{n+1}=S_n e^{\left(r-\frac{1}{2}\sigma^2\right)h+\sqrt{h}\sigma Z_{1,n}}$$

לכן הייתי מציע לפתור את המשוואה הראשונה באופן כללי על ידי הרקורסיה

$$S_{n+1}=S_n e^{\left(r-\frac{1}{2}V_n\right)h+\sqrt{hV_n}Z_{1,n}}$$

(עוד יותר טוב מזה - במקום V_n בנוסחה זו להשתמש ב- $\frac{1}{2}(V_n+V_{n+1})$.)

---

2. המחירים של שתי מניות $S_1(t),S_2(t)$ מקיימים את המשוואות

$$\begin{aligned} dS_1 &= S_1(r_1dt+\sigma_{11}dW_1+\sigma_{12}dW_2) \\ dS_2 &= S_2(r_2dt+\sigma_{21}dW_1+\sigma_{22}dW_2) \end{aligned}$$

כאשר $r_1,r_2,\sigma_{11},\sigma_{12},\sigma_{21},\sigma_{22}$ הם קבועים חיוביים, עם $r_1>r_2$, ו- W_1,W_2 הם תהליכי וינר בלתי תלויים. רוצים לתמחר 2 אופציות על מניות אלה. הראשונה תשלם $|S_1(T)-S_2(T)|$ בעת הפקיעה T. השנייה תשלם $\max_{0\le t\le T}|S_1(t)-S_2(t)|$ בעת הפקיעה.

(א) הסבר בקצרה איך היית משתמש בשיטת מונטה-קרלו לתמחר את האופציה הראשונה.

(ב) הסבר בקצרה איך היית משתמש בשיטת מונטה-קרלו לתמחר את האופציה השנייה. למה, לדעתך, הקירוב לאופציה זו יהיה פחות טוב ?

(ג) אופציה שלישית מעניקה את הזכות לממש בכל זמן t מ-0 עד T, ולקבל באותו זמן $|S_1(t)-S_2(t)|$. מחזיק האופציה החליט לממש את האופציה לפני עת הפקיעה בזמן הראשון t שבו $S_2(t)>S_1(t)e^{(r_1-r_2)(T-t)}$, ואם לא קיים זמן כזה להחזיק עד עת הפקיעה. הסבר בקצרה איך היית משתמש בשיטת מונטה-קרלו למצוא את הערך של אופציה זו למחזיק זה.

(א) הייתי בונה דגימות של $S_1(T),S_2(T)$ על ידי הנוסחאות

$$\begin{aligned} S_1(T) &= S_1(0)e^{\left(r_1-\frac{1}{2}\sigma_1^2\right)T+\sqrt{T}(\sigma_{11}Z_1+\sigma_{12}Z_2)} \\ S_2(T) &- S_2(0)e^{\left(r_1-\frac{1}{2}\sigma_2^2\right)T+\sqrt{T}(\sigma_{21}Z_1+\sigma_{22}Z_2)} \end{aligned}$$

כאשר $\sigma_1^2=\sigma_{11}^2+\sigma_{12}^2$, $\sigma_2^2=\sigma_{21}^2+\sigma_{22}^2$ ו- Z_1,Z_2 הם משתנים מקריים נורמליים סטנדרטים בלתי תלויים. מאלה ניתן לבנות דגימות של $|S_1(T)-S_2(T)|$ ולחשב אומדנים לתוחלת ולשונות של $e^{-rT}|S_1(T)-S_2(T)|$. התוחלת היא אומדן למחיר, עם טעות מסדר גודל של $\frac{\sigma}{\sqrt{M}}$ כאשר σ היא סטיית התקן ו- M הוא מספר הדגימות.

(ב) לאופציה השנייה לא מספיק לדעת את הערכים של $S_1(T),S_2(T)$ כדי לתמחר את האופציה. יש צורך לדעת גם בזמנים באמצע הקטע. דוגמים את המחירים בזמנים $h,2h,\ldots,Nh=T$. כל ש- N יותר גדול נקבל מחיר מדוייק יותר, אבל זמן הריצה גם יהיה ארוך יותר. הסימולציה של האופציה הראשונה תהיה יותר מדוייקת, היות ויש פחות גורמים סטוכסטיים רלוונטיים.

(ג) מאוד דומה לסעיף הקודם. בכל סימולציה יש צורך להחליט האם לממש את האופציה ואם כן מתי - אין טעם להמשיך את הסימולציה אחרי זמן המימוש. אם מממשים לפני זמן הפקיעה יש לשים גורם היוון מתאים: e^{-rt} במקום e^{-rT}. אפשר לבנות אסטרטגיות שונות לדגום את הערך בהנחה שכן מממשים בעזרת ה- Brownian bridge.

---

3. המשתנים הסטוכסטיים S, V מקיימים את המשוואות הדפרנציאליות הסטוכסטיות

$$\begin{aligned} dS &= S(rdt + \sqrt{V}dW_1) \\ dV &= a(V_0 - V)dt + b\sqrt{V}(\rho\, dW_1 + \sqrt{1-\rho^2}dW_2) \end{aligned}$$

כאשר r, a, V_0, b, ρ הם קבועים חיוביים עם $\rho < 1$ ו- W_1, W_2 הם תהליכי וינר בלתי תלויים. הוכח שהתוחלת

$$u(x,y,t) = \mathbf{E}[e^{-rT}(S(T)-K)_+ | S(t) = x, V(t) = y]$$

(K, T קבועים) מקיימת את המד"ח

$$u_t + rxu_x + a(V_0 - y)u_y + \frac{1}{2}x^2yu_{xx} + b\rho xyu_{xy} + \frac{1}{2}b^2yu_{yy} = ru$$

איך היית מיישם את שיטת אויילר למשוואה זו ? (אין צורך להתייחס לתנאיי שפה, אבל יש להתייחס לפחות לתחום הרלוונטי של הקואורדינטות).

נוסחת פיינמן-כץ הרב מימדי: אם המשתנים הסטוכסטיים X_i מקיימים

$$dX_i = a_i(\mathbf{X},t)dt + \sum_{k=1}^{d} b_{ik}(\mathbf{X},t)dW_k$$

אזי התוחלת

$$u(\mathbf{x},t) = \mathbf{E}[e^{-rT}\psi(\mathbf{X},T) | \mathbf{X}(t) = \mathbf{x}]$$

מקיימת את המשוואה

$$u_t + \sum_i a_i(\mathbf{x},t)u_{x_i} + \frac{1}{2}\sum_i\sum_j\sum_k b_{ik}(\mathbf{x},t)b_{jk}(\mathbf{x},t)u_{x_ix_j} = ru$$

כאן

$$\begin{array}{lll} a_1(S,V,t) = rS & b_{11}(S,V,t) = S\sqrt{V} & b_{12} = 0 \\ a_2(S,V,t) = a(V_0 - V) & b_{21}(S,V,t) = b\sqrt{V}\rho & b_{22}(S,V,t) = b\sqrt{V}\sqrt{1-\rho^2} \end{array}$$

רק צריך להציב בנוסחה!

ליישם שיטת אויילר: העניין המרכזי הוא ש- $u(x,y,t)$ היא עכשיו פונקציה של שלושה משתנים x, y, t. ולכן יש צורך לבחור צעדים בכל שלושת הכיוונים: h_1, h_2, k ונעשה קירובים ל- u בצורה

$$u_{i_1,i_2,j} \approx u(i_1h_1, i_2h_2, jk)$$

משתמשים בקירובי הפרשים סופיים הבאים:

$$\begin{aligned}
u_t &\to \frac{u_{i_1,i_2,j+1}-u_{i_1,i_2,j}}{k}\\
u_x &\to \frac{u_{i_1+1,i_2,j}-u_{i_1-1,i_2,j}}{2h_1}\\
u_y &\to \frac{u_{i_1,i_2+1,j}-u_{i_1,i_2-1,j}}{2h_2}\\
u_{xx} &\to \frac{u_{i_1+1,i_2,j}-2u_{i_1,i_2,j}+u_{i_1-1,i_2,j}}{h_1^2}\\
u_{yy} &\to \frac{u_{i_1,i_2+1,j}-2u_{i_1,i_2,j}+u_{i_1,i_2-1,j}}{h_1^2}\\
u_{xy} &\to \frac{u_{i_1+1,i_2+1,j}-u_{i_1+1,i_2-1,j}-u_{i_1-1,i_2+1,j}+u_{i_1-1,i_2-1,j}}{4h_1h_2}
\end{aligned}$$

איפה שמופיע במשוואה x נכתוב i_1h_1, ואיפה שמופיע y נרושם i_2h_2. לגבי התחומים של הקואורדינטות: גם x וגם y (המייצגים את הערכים הנוכחיים של S ו- V בהתאם) יהיו חיוביים, ובעקרון אינם מוגבלים. אבל לצורך הפתרון הנומרי נצטרך לבחור ערכים מקסימליים $x_{\max}$ ו- $y_{\max}$ ונקח

$$h_1=\frac{x_{\max}}{N_1}\ , \qquad h_2=\frac{y_{\max}}{N_2}$$

כאשר N_1 ו- N_2 הם מספרי הצעדים בכיוונים x ו- y, בהתאם.

---

4. רוצים לפתור את בעיית השפה החופשית

$$\begin{aligned}
u_t &= u_{xx}+\frac{2}{x}u_x\ , \qquad 0<x<s(t)\ ,t>0\\
u_x(0,t) &= 0\\
u(s(t),t) &= 0\\
u_x(s(t),t) &= -s'(t)\\
u(x,0) &= 1-x^2\ , \quad s(0)=1
\end{aligned}$$

על ידי החלפת קואורדינטות

$$\begin{cases} x=ys(\tau)\\ t=\tau\end{cases}$$

נסח את הבעייה כבעייה על התחום הקבוע $0<y<1$, והסבר בקצרה איך ניתן לפתור את הבעייה החדשה עם שיטת איילר.

על ידי כלל השרשרת:

$$\begin{aligned}
\frac{\partial}{\partial y} &= \frac{\partial x}{\partial y}\frac{\partial}{\partial x}+\frac{\partial t}{\partial y}\frac{\partial}{\partial t}=s(\tau)\frac{\partial}{\partial x}\\
\frac{\partial}{\partial \tau} &= \frac{\partial x}{\partial \tau}\frac{\partial}{\partial x}+\frac{\partial t}{\partial \tau}\frac{\partial}{\partial t}=ys'(\tau)\frac{\partial}{\partial x}+\frac{\partial}{\partial t}
\end{aligned}$$

ולכן בקואורדינטות החדשות המשוואה היא

$$u_\tau-\frac{ys'(\tau)}{s(\tau)}u_y=\frac{1}{s(\tau)^2}u_{yy}+\frac{2}{ys(\tau)^2}u_y$$

והבעייה החדשה היא

$$\begin{aligned} u_\tau &= \frac{1}{s(\tau)^2}u_{yy} + \left(\frac{2}{ys(\tau)^2} + \frac{ys'(\tau)}{s(\tau)}\right)u_y \ , \qquad 0<y<1\ , \tau>0 \\ u_y(0,\tau) &= 0 \\ u(1,\tau) &= 0 \\ u_y(1,\tau) &= -s(\tau)s'(\tau) \\ u(y,0) &= 1-y^2\ , \quad s(0)=1 \end{aligned}$$

שיטת אויילר לבעייה זו: נבצע את הדסקרטיזציה הסטנדרטית לקבל

$$u_{i,j+1} = u_{i,j} + k\left(\frac{1}{s_j^2}\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2} + \left(\frac{2}{ihs_j^2} + \frac{ih(s_{j+1}-s_j)}{ks_j}\right)\frac{u_{i+1,j}-u_{i-1,j}}{2h}\right)$$

עם היינו יודעים את s_{j+1} היינו יכולים להשתמש בזה למצוא את הערכים של $u_{i,j+1}$, עבור $1 \le i \le N-1$, מתוך הערכים של $u_{i,j}$. את הערכים של $u_{0,j+1}$ ושל $u_{N,j+1}$ מוצאים משני תנאי השפה הראשונים: דורשים $u_{N,j+1}=0$ ו- $3u_{0,j+1}-4u_{1,j+1}+u_{2,j+1}=0$. יש צורך לבחור את s_{j+1} כך שגם תנאי השפה האחרונה תתקיים: כלומר כך ש-

$$3u_{N,j+1} - 4u_{N-1,j+1} + u_{N-2,j+1} = -2hs_{j+1}\frac{s_{j+1}-s_j}{k}$$

(כאן כתבתי $u_{N,j+1}$ למרות שזה שווה 0.) את ה"בחירה" הזאת עושים על ידי שיטת ניוטון - עושים ניחוש ראשון $s_{j+1}=s_j$ ומשפרים את הניחוש לפי כלל ניוטון עד שהתנאי שרוצים מתקיים.

---

5. <u>יש לענות או על סעיף (א) או על סעיף (ב)</u>

(א) (המשך של שאלה 1.) כתוב קוד מטלב לחשב את $\mathbf{E}[e^{-rT}(S(T)-K)_+]$ (K,T קבועים). יש להניח ש a,b הם קטנים, ולהעזר גם במה שהצעת בשאלה 1, סעיף (ג), וגם בתשובה הידועה לתוחלת במקרה $a=b=0$ כמשתנה בקרה.

(ב) (קשור לשאלה 4.) כתוב קוד מטלב להפעיל את שיטת Crank-Nicolson לפתור את הבעייה

$$\begin{aligned} u_t &= u_{xx} + \frac{2}{x}u_x \ , \qquad 0<x<1\ , 0<t<1 \\ u_x(0,t) &= 0 \\ u(1,t) &= 0 \\ u(x,0) &= 1-x^2 \end{aligned}$$

יש לתת למשתמש לבחור את מספר נקודות הרשת בכל כיוון וכיוון. ניתן להניח שניתנה פונקציה הפותרת מערכות תלת-אלכסוניות.

אופציה (א):

```
% SDE parameters
r = 0.1;
V0 = 0.04;
```

```
a = 0.03;
b = 0.09;
rho = 0.1;
% option parameters
T = 1 ;
K = 1.0 ;
S0 = 0.9 ;
% numerical parameters for small simulation to get correlation
N = 100;      % time subdivision
M = 1000;     % number of simulations
h = T/N;
% random numbers
Z1 = randn(M,N);
Z2 = randn(M,N);
% initialize S and V
S  = S0*ones(M,1);
V  = V0*ones(M,1);
for i=1:N
    V = V + a*h*(V0-V) + b*sqrt(h*V).*(rho*Z1(:,i) + sqrt(1-rho^2)*Z2(:,i)) ;
    S = S.*exp( (r-V/2)*h + sqrt(h*V).*Z1(:,i) );
end
% get value of option
call_value = exp(-r*T) * (S-K).* (S>K);
% value of regular call based on same random numbers
Sfinal = S0 * exp( (r-V0/2)*T + sqrt(h*V0)*sum(Z1,2) );
reg_call_value = exp(-r*T) * (Sfinal-K).* (Sfinal>K);
% correlation of option and regular call
q=cov(call_value,reg_call_value);
c=-q(1,2)/q(2,2);
% the big simulation
M = 10000;     % number of simulations
Z1 = randn(M,N);
Z2 = randn(M,N);
S  = S0*ones(M,1);
V  = V0*ones(M,1);
```

```
for i=1:N
    V = V + a*h*(V0-V) + b*sqrt(h*V).*(rho*Z1(:,i) + sqrt(1-rho^2)*Z2(:,i)) ;
    S = S.*exp( (r-V/2)*h + sqrt(h*V).*Z1(:,i) );
end
% get value of option
call_value = exp(-r*T) * (S-K).* (S>K);
% value of regular call based on same random numbers
Sfinal = S0 * exp( (r-V0/2)*T + sqrt(h*V0)*sum(Z1,2) );
reg_call_value = exp(-r*T) * (Sfinal-K).* (Sfinal>K);
% blsprice on same parameters
bls = blsprice( S0, K, r, T, sqrt(V0), 0);
% corrected option
corrected = call_value + c*(reg_call_value - bls);
% now get some results
ans1 = [ mean(call_value), std(call_value)/sqrt(M) ]
ans2 = [ mean(corrected), std(corrected)/sqrt(M) ]
```

אופציה (ב)

```
function  U = ma5a(N,M)


% solving u_t = u_{xx} + 2/x u_x     0<x<1, 0<t<1
% intial condition u(0,t)=1-x^2
% right boundary condition u(1,t)=0
% left boundary condition u_x(0,t)=0 (otherwise equation not well defined!)


% u will be a matrix of size (M+1)x(N+1) giving the solution of our problem
% on the required grid - each row is the solution at a certain time


U = zeros(M+1,N+1);
h = 1/N;
k = 1/M;
x = (0:N)/N;
% initial condition
U(1,:) = 1-x.^2;
```

```
% The Euler method:
% u_{i,j+1}=u_{i,j} + k/h^2 (u_{i+1,j}(1+1/(i-1))- 2u_{i,j} + (1-1/(i-1))u_{i-1,j})
% (need to explain why "i-1" .... we want i to go from 0 to N but in matlab
%      it goes from 1 to N+1)
% This equation valid only for i from 2 to N !
% CN method:
% u_{i,j+1} - k/2h^2 ( u_{i+1,j+1}(1+1/(i-1)) - 2u_{i,j+1} + (1-1/(i-1))u_{i-1,j+1})
%    = u_{i,j} + k/2h^2 ( u_{i+1,j}(1+1/(i-1)) - 2u_{i,j} + (1-1/(i-1))u_{i-1,j} )
% Again this equation valid for i from 2 to N !
% In this u_{N+1,j+1} can be ignored as it is zero
%         u_{1,j+1}  also does not appear as for i=2, 1-1/(i-1) = 0 !!
% But we must remember that u_{1,j+1} = (4u_{2,j+1}-u_{3,j+1})/3


d=(1+k/h^2)*ones(N-1,1);      % the diagonal
u=-k/2/h^2*(1+1./(1:N-2));    % the upper diagonal
l=-k/2/h^2*(1-1./(2:N-1));    % the lower diagonal


for counter=1:M
    r = (1-k/h^2)*U(counter,2:N) + k/2/h^2*( U(counter,3:(N+1)).* (1+1./(1:N-1)) ...
           + U(counter,1:(N-1)) .* (1-1./(1:N-1)) );
    U(counter+1,2:N)=tridiag(d,u,l,r);
    U(counter+1,1)=(4*U(counter+1,2)-U(counter+1,3))/3;
end


end
```
