Kristoffer Karlsson, kristoffer.karlsson@miun.se
Magnus Eriksson, magnus.eriksson@miun.se

Laboration 2: Stokastiska metoder – Monte Carlo

Denna laboration ger er möjlighet att bekanta er med MATLAB:s möjligheter för att utföra Monte Carlo-simuleringar. Var beredd att visa och förklara all programkod, alla diagram och alla numeriska resultat.

Uppgift 1

I MATLAB kan man generera slumptal med de två inbyggda funktionerna rand och randn. De skapar slumptal ur 2 olika stokastiska fördelningar, nämligen *likformig fördelning* och *normalfördelning*. Det finns även andra fördelningar, men detta är de 2 viktigaste. Det finns 2 viktiga begrepp när det gäller normalfördelning, nämligen *väntevärde* och *varians.* Hur dessa begrepp påverkar fördelningarna kommer att åskådliggöras när ni gör uppgifterna. Ladda ner m-filerna DemoNormalf.m och DemoLikformf.m från Moodle. De innehåller MATLAB-funktionen som demonstrerar normalfördelning respektive likforming fördelning. Båda ritar dels upp en plot med samtliga slumptal som skapats, dels ett histogram med 100 staplar. Stapelns höjd visar antalet slumptal inom det intervall som stapeln täcker.

a) Använd DemoNormalf för att studera normalfördelningen. Börja med 100 slumptal, öka sedan successivt upp till 1000000 slumptal, d.v.s. kör N=100, 1000, 10000, 100000, 1000000 (använd *defaultvärden* på medelvärde och varians). Läs hjälptexten i m-filen för att förstå hur funktionen ska användas.

b) Kör med 10000 slumptal men ange flera medelvärden, ni kan köra med 10 och 100, och en fix varians (kör variansen 1). Titta på de 2 figurerna som ni skapat för de olika medelvärdena. *Vad har medelvärdet för effekt på slumptalen?*

c) Utför samma sak som i b), men använd ett fixt medelvärde (kör medelvärdet 0) och använd istället flera värden på variansen, ni kan köra med 1, 10 och 50. Titta på de 2 figurerna som ni skapat för de olika varianserna. *Vad har variansen för effekt på slumptalen?*

Kristoffer Karlsson, kristoffer.karlsson@miun.se
Klas Gustavsson, klas.gustavsson@miun.se

Nu ska ni nästan upprepa a)-c) men istället för normalfördelning använda likformig fördelning. MATLAB:s funktion för likformig fördelning, rand, genererar slumptal på intervallet [0,1]. Vill man ha ett annat intervall, får man multiplicera med en faktor, t.ex. 2*rand(n,1) ger n st likformigt fördelade slumptal på intervallet [0,2], som är lagrade i en array som har storleken $n \times 1$.

d) Använd funktionen DemoLikformf för att studera den likformiga fördelningen på intervallet [0,1] (Använd hjälptexten för att förstå hur funktionen ska användas). Börja med 100 slumptal och öka sedan successivt upp till 1000000 slumptal. Ändra även intervallet. *Hur ser kurvan ut för olika antal slumptal och för olika intervall?*

Uppgift 2

Man kan ställa sig frågan: ”Vad blir medelvärdet om vi kastar en tärning med 6 sidor numrerade {1, 2, 3, 4, 5, 6}?”
Det har vi räknat ut tidigare och vi fick medelvärdet $\mu = 3.5$. Men nu ska vi istället räkna ut det med Monte Carlo. Vad vi gör är att simulera n st tärningskast, lagra utfallen i en array y och räkna ut medelvärdet med mean(y). Man kan simulera tärningskast genom att slumpa *heltal* i intervallet [1,6] där alla tal har lika stor sannolikhet, d.v.s. det blir en likformig fördelning, som i detta fall blir en *diskret* likformig fördelning.

För att konstruera den diskreta likformiga fördelning på intervallet som behövs här kan man använda rand i MATLAB på följande sätt:

```
y = floor(1+6*rand(1,n));
```

där n är antalet slumptal och floor avrundar nedåt till närmaste heltal. y innehåller då n st slumpmässiga heltal i intervallet [1,6].

Kristoffer Karlsson, kristoffer.karlsson@miun.se
Klas Gustavsson, klas.gustavsson@miun.se

a) Generera med koden arrayerna y1, y2 och y3 med n1=10^3, n2=10^4 och n3=10^5, samt beräkna medelvärdet för var och en av arrayerna, medel_y1=mean(y1), medel_y2=mean(y2) och medel_y3=mean(y3). *Stämmer det att skillnaden mellan beräknat medelvärde och det exakta bli mindre ju fler slumptal som används?*

b) Eftersom man får nya slumptal från rand varje gång man anropar funktionen, kommer medelvärdena som beräknas skilja sig något varje gång man utför beräkningen. Undersök detta genom att beräkna medelvärdena för 10 st kast per medelvärde som upprepas 10^3, 10^4 och 10^5 gånger. Ni kan använda m-filen tarning_upprepa.m. m-filen kommer ge 3 st plottade histogram. Placera de 3 figurerna bredvid varandra och jämför. *Påverkas denna fördelning i stort av de olika värdena på upprepning?*

c) Testa nu att kasta lite fler tärningskast. Kör 200 kast per medelvärde och upprepa 10^5 gånger. *Vad skiljer denna fördelningsfunktion mot fördelningsfunktionen i b)? Varför?*

Vad du sett ovan är vad man inom statistiken kallar den *centrala gränsvärdessatsen*. Satsen säger i korthet att om man beräknar medelvärdet av ett stort antal slumptal, X_i, för en viss sannolikhetsfunktion, $f_X(x)$, kommer medelvärdet, $\bar{X}$, fördelas enligt en normalfördelning, d.v.s. att $\bar{X} \in N\left(\mu, \frac{\sigma}{\sqrt{n}}\right)$. Detta gäller oavsett vilken sannolikhetsfunktion som använts.

Sannolikhetsfunktionen, $f(x)$, för en normalfördelning är:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (1)$$

där μ är förväntansvärdet och σ är standardavvikelsen.

Vi har beräknat medelvärden av tärningskasten, $\bar{x}_n$, enligt:

$$\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (2)$$

Variansen för medelvärdet, $V(\bar{x}_n)$, blir:

$$V(\bar{x}_n) = V\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) = \frac{1}{n}V(x) = \frac{35}{12n} \qquad (3)$$

där talet $\frac{35}{12} \approx 2.9$ är variansen för en tärning, som har visats på föreläsning.

Kristoffer Karlsson, kristoffer.karlsson@miun.se
Klas Gustavsson, klas.gustavsson@miun.se

Standardavvikelsen, σ_n, blir då:

$$\sigma_n = \sqrt{V(\bar{x}_n)} = \sqrt{\frac{35}{12n}} \qquad (4)$$

och väntevärdet är $\mu = 3.5$.

d) Kör nu m-filen tarning_upprepa.m med n=200 och N=10^5 igen. Skriv sedan hold nedanför loopen, vilket gör att vi kan fortsätta att plotta i samma fönster utan att histogrammet försvinner. Skapa en array x med samma gränser som i histogrammet och beräkna sedan värden för sannolikhetsfunktionen i (1) (kalla arrayen f i MATLAB) där standardavvikelse och väntevärde är givet enligt ovan. Använd kommandot plot(x,1000*f) och visa labbhandledaren. *Varför var vi tvungna att multiplicera f med 1000*?

e) Simulera en fusktärning med valfri diskret sannolikhetsfunktion p(*i*), där *i*=1, 2,...6, representerad numeriskt av vektorn p. Normera först vektorn så att sum(p) är 1, och skapa en vektor F=cumsum(p), som representerar den samplade kumulativa fördelningsfunktionen (cdf:en). I en loop, skapa likformigt fördelade slumptal u (mellan 0 och 1) och transformera dem enligt metoden *inverse transform sampling* (googla på det), exempelvis genom att beräkna sum(u<F)+1 eller göra en case-switch-sats. Plotta histogram utan medelvärdesbildning (motsvarande n=1) och jämför med p. Diskutera hur metoden skulle kunna användas för att generera en kontinuerlig stokastisk variabel med valfri täthetsfunktion f(*y*) och sampelvektor *y* mellan *a* och *b*.

Uppgift 3

Ett användningsområde för Monte Carlo-metoder är beräkning av integraler av hög dimensionalitet. Orsaken är att vid höga dimensioner blir kvadraturbaserade metoder såsom trapetsregeln eller Simpsons regel alltför beräkningstunga.

I den här uppgiften ska vi lösa en integral som i 1-D skrivs:

$$\int_{-L}^{L} e^{-x^2} dx \qquad (5)$$

Integranden i (5) är ett exempel på normalfördelning som ni arbetade med i föregående uppgift. Integralen i (5) är också välkänd för att sakna primitiv funktion vilket gör att det inte finns någon trivial lösning.

Vi ska lösa integralen med en Monte Carlo metod enligt följande:

$$\frac{2L}{N} \sum_{i=1}^{N} e^{-x_i^2} \qquad (6)$$

där N st värden på x väljs slumpvis i intervallet *[-L, L]* (i högre dimensioner blir det ett intervall per dimension). Vi ser att integralen approximeras med en summa och ju större värde på N desto bättre värde erhåller vi (om $N, L \to \infty$ får vi exakta integralvärdet som är $\sqrt{\pi}$)

Kristoffer Karlsson, kristoffer.karlsson@miun.se
Klas Gustavsson, klas.gustavsson@miun.se

Ladda ner m-filen mcconv.m från Moodle, som vi ska använda för att räkna ut integralen med Monte Carlo för olika antal slumptal N. Som utdata får vi förutom lösningen till integralen, mu, även en vektor med de N som använts, N, samt uppskattning av felet i approximationen (som baseras på standardavvikelsen), err.

a) Använd integralgränserna [-5 5] i samtliga dimensioner. I 1-D blir det t.ex.:

```
[N, mu, err]=mcconv([-5 5], 1000);
```

Testa N=1000, 10000, 100000, 1000000. *Får du konvergens?*
Gör sedan samma för 2-D. *Varför blir det olika svar för 1-D v.s. 2-D?*

b) Nästa uppgift blir att ta reda på beräkningskomplexiteten p, d.v.s. hur felet E varierar som funktion av antalet punkter N.
Man kan göra följande ansats att felet, E, ges enligt:

$$E \approx CN^p \qquad (7)$$

Felet err och antal punkter N är utparametrar till funktionen mcconv och motsvarar E och N i (7). Genom att använda kurvanpassning (linjär regression) på logaritmerade data kan vi uppskatta p och C.

Om vi logaritmerar (7) får vi:

$$\ln(E) \approx \ln(C) + p\ln(N) \qquad (8)$$

vilket är på formen $y = a_0 + a_1 x$ där $y = \ln(E)$, $a_0 = \ln(C)$, $a_1 = p$ och $x = \ln(N)$.

I MATLAB kan vi använda följande kommandon för uppskattningen (i 1-D):

```
[N,mu, err] = mcconv([-5 5]);
a = polyfit(log(N),log(err),1)
```

där exponenten p finns i a(1) och konstanten $\ln(C)$ i a(2) och vi har använt default-värdena på N (se m-filen). För att erhålla säkrare resultat ska vi istället använda värdena N=[10000; 100000; 500000; 1000000] med genomgående integralgränserna [-5 5] i samtliga dimensioner. Fortsätt och undersök p och C även för 2-D integralen. *Hur ändras p beroende på dimensionen? Hur ändras C beroende på dimensionen?*

Kristoffer Karlsson, kristoffer.karlsson@miun.se
Klas Gustavsson, klas.gustavsson@miun.se

c) En alternativ metod för beräkning av integralen är trapetsregeln. Man delar då in intervallet *[-L, L]* med ekvidistanta punkter. *Med 5 punkter likformigt utspridda i varje dimension, hur många punkter skulle det då bli för en 2-D integral? För en 10-D integral? För en 20-D integral?*

Uppgift 4

Blackjack är ett synnerligen populärt kortspel i t.ex. Las Vegas. Om en spelare vill öka sina chanser att vinna finns det ett antal regler att förhålla sig till (”Basic Strategy”). Man kan komma fram till dessa strategier med hjälp av stokastiska metoder/sannolikhetsteori inom matematik, men det är oerhört mycket enklare att designa ett ”**Monte Carlo game**”. Man kan då komma fram till precis samma resultat som de mer avancerade metoderna.

I m-filen `blackjacksim(n)` simuleras `n` st Blackjack-omgångar.

Antag att vi går på Casino Cosmopol i Sundsvall årets samtliga dagar, 365 st. Varje dag spelar vi 100 st Blackjack-omgångar, d.v.s. under en dag kör vi `bs=blackjacksim(100)`, där det skapas en array `bs` med den ackumulerade vinsten eller förlusten för varje omgång och det sista elementet i arrayen innehåller antal kronor du vunnit eller förlorat.

Testa att plotta några omgångar med `plot(blackjacksim(100))`. Man ser att det är väldiga skillnader i resultaten.

Uppgiften är nu att skriva ett m-script som beräknar vinsten/förlusten varje dag under 1 helt år och summerar ihop slutresultatet varje dag för att se om det kanske blir vinst totalt under året.

Tycker du dig se om det blir vinst eller förlust?