**NSEG-5984 Monte Carlo Methods for Particle Transport**

Nagendra Krishnamurthy

Homework -1

**Problem 1:**

Flowchart:

Start

iter = 0

Generate η1 and η2

n1 = INT(6 xη1)

n2 = INT(6 xη2)

s = n1 + n2

count(s) = count(s) + 1

iter < iterMax?

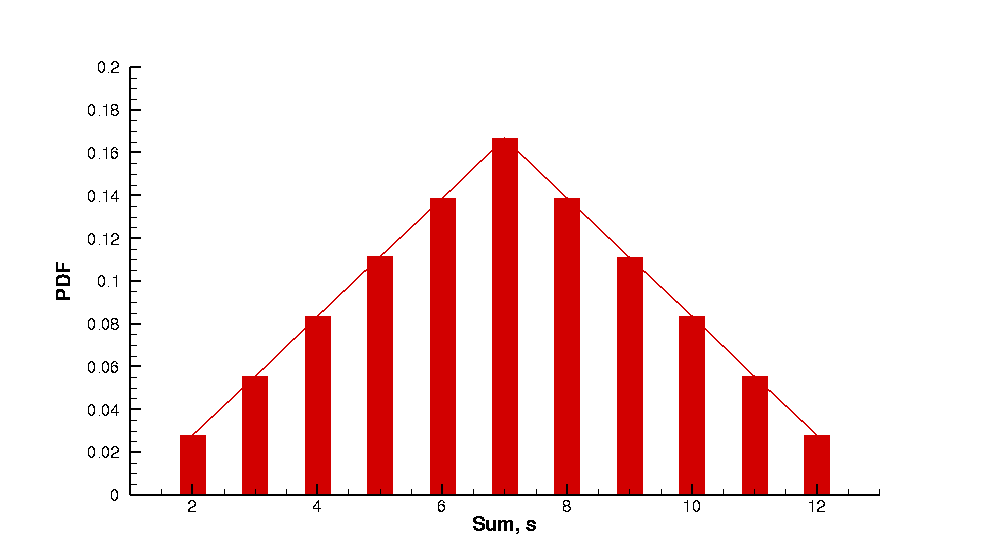
no

yes

pdf(s) = pdf(s)/iterMax

Stop

The PDF obtained is provided in the following plot. The code was executed for maxIter = 1000000.



Source code:

PROGRAM sumDice

INTEGER :: iter, maxIter, n1, n2, sum, unitPdfFile

REAL :: ranNum\_1, ranNum\_2, pdfBin(2:12)

CHARACTER(80) :: namePdfFile

! Start

PRINT\*, 'Enter the maximum number of iterations:'

READ\*, maxIter

CALL random\_seed()

iter = 0

DO WHILE (iter .le. maxIter)

! Increment iteration number

iter = iter + 1

! Generate two random numbers

CALL random\_number(ranNum\_1)

CALL random\_number(ranNum\_2)

! Compute values for n1 and n1 based on the random numbers

n1 = INT(6.\*ranNum\_1) + 1

n2 = INT(6.\*ranNum\_2) + 1

! Computer sum of the two die values

sum = n1 + n2

! Bin the sum for computation of pdf

pdfBin(sum) = pdfBin(sum) + 1

END DO

! Computer pdf by dividing the bin by total number of iterations

pdfBin(:) = pdfBin(:)/maxIter

unitPdfFile = 101

namePdfFile = 'pdf.dat'

OPEN (UNIT = unitPdfFile, FILE = namePdfFile, STATUS = 'replace', &

POSITION = 'rewind', FORM = 'formatted', ACTION = 'write')

DO i = 2, 12

WRITE(unitPdfFile,501) i, pdfBin(i)

END DO

501 FORMAT (i3.3, 1X, 1(pe12.5, 1X))

END PROGRAM sumDice

**Problem 2:**

yes

no

Start

Iter = 0

Create list of cards

i = 0; iter = iter + 1

Generate ηi

Find card number

ci = INT(52 – i + 1) + 1

Append card to handList

Remove card from cardList

Are all chosen cards of same suit?

yes

success = success + 1

Stop

i < 5?

no

probFlush = success/iterMax

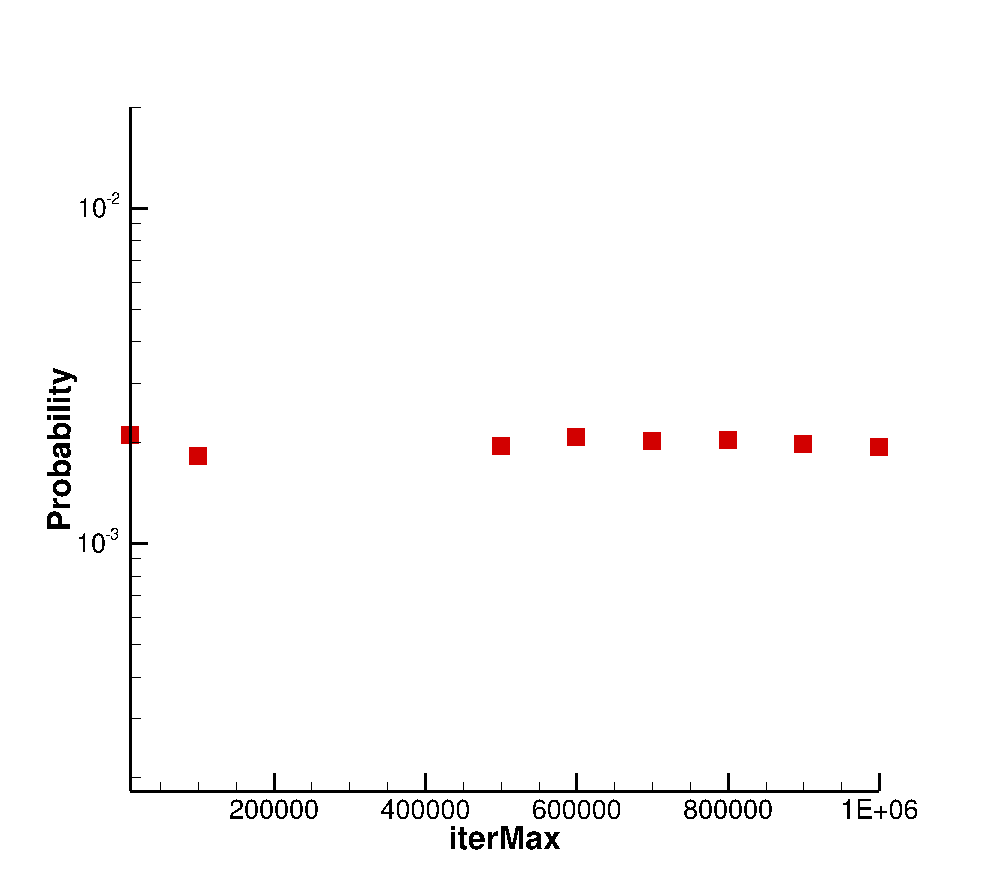
i < 5?

i = i + 1

yes

no

The code was executed for various values of iterMax. From the following plot, it is observed that the probability of getting a flush hand is approximately 0.20x10-2.



Source code:

PROGRAM flushPokerHand

IMPLICIT NONE

TYPE card

INTEGER :: cardNum, cardType

TYPE (card), POINTER :: next

END TYPE card

INTEGER :: iter, maxIter, c1, unitProbFile, i, nCards, &

successCount, iCard

REAL :: ranNum, probFlush

CHARACTER(80) :: nameProbFile

TYPE (card), POINTER :: cardList, cardCurrent, cardLast, handList, &

handCurrent, handLast, cardTemp

! Start

PRINT\*, 'Enter maximum number of iterations'

READ\*, maxIter

successCount = 0

nCards = 5

CALL random\_seed()

ALLOCATE(cardList)

NULLIFY(cardList%next)

cardLast => cardList

ALLOCATE(handList)

NULLIFY(handList%next)

handLast => handList

iter = 0

DO WHILE (iter .lt. maxIter)

300 CONTINUE

CALL deleteList(cardList)

cardLast => cardList

CALL deleteList(handList)

handLast => handList

! Increment iteration number

iter = iter + 1

! Create a list of cards

DO iCard = 1, 52

ALLOCATE(cardLast%next)

NULLIFY(cardLast%next%next)

cardLast => cardLast%next

cardLast%cardNum = iCard

cardLast%cardType = CEILING(iCard/13.0)

END DO

DO i = 1, nCards

! Generate random number

CALL random\_number(ranNum)

! Compute values for c1 based on the random numbers

c1 = INT((52.0-i+1)\*ranNum) + 1

! Find the card

cardCurrent => cardList

DO iCard = 1, c1

cardCurrent => cardCurrent%next

END DO

! Append card to handList

ALLOCATE(handLast%next)

NULLIFY(handLast%next%next)

handLast => handLast%next

handLast%cardNum = cardCurrent%cardNum

handLast%cardType = cardCurrent%cardType

! Remove card from cardList

cardCurrent => cardList

DO iCard = 1, c1-1

cardCurrent => cardCurrent%next

END DO

cardTemp => cardCurrent%next

cardCurrent%next => cardCurrent%next%next

DEALLOCATE(cardTemp)

! Check if all cards in hand are of same suit

handCurrent => handList

DO WHILE (ASSOCIATED(handCurrent%next%next))

IF (handCurrent%next%cardType .NE. &

handCurrent%next%next%cardType) THEN

GOTO 300

END IF

handCurrent => handCurrent%next

END DO

END DO

! Success

successCount = successCount + 1

handCurrent => handList

DO WHILE (ASSOCIATED(handCurrent%next))

handCurrent => handCurrent%next

END DO

END DO

! Compute probability of flush hand

probFlush = 1.0\*successCount/maxIter

unitProbFile = 101

nameProbFile = 'probFlush.dat'

OPEN (UNIT = unitProbFile, FILE = nameProbFile, &

POSITION = 'append', FORM = 'formatted', ACTION = 'write')

WRITE(unitProbFile,501) maxIter, probFlush

501 FORMAT (i7.7, 1X, 1(e12.5, 1X))

CONTAINS

SUBROUTINE printList(list)

IMPLICIT NONE

TYPE (card), POINTER, INTENT(IN) :: list

TYPE (card), POINTER :: current

current => list

DO WHILE (ASSOCIATED(current%next))

current => current%next

PRINT\*, current%cardNum, current%cardType

END DO

END SUBROUTINE printList

SUBROUTINE deleteList(list)

IMPLICIT NONE

TYPE (card), POINTER, INTENT(IN) :: list

TYPE (card), POINTER :: current, previous

current => list%next

DO WHILE (ASSOCIATED(current))

previous => current

current => current%next

DEALLOCATE(previous)

END DO

NULLIFY(list%next)

END SUBROUTINE deleteList

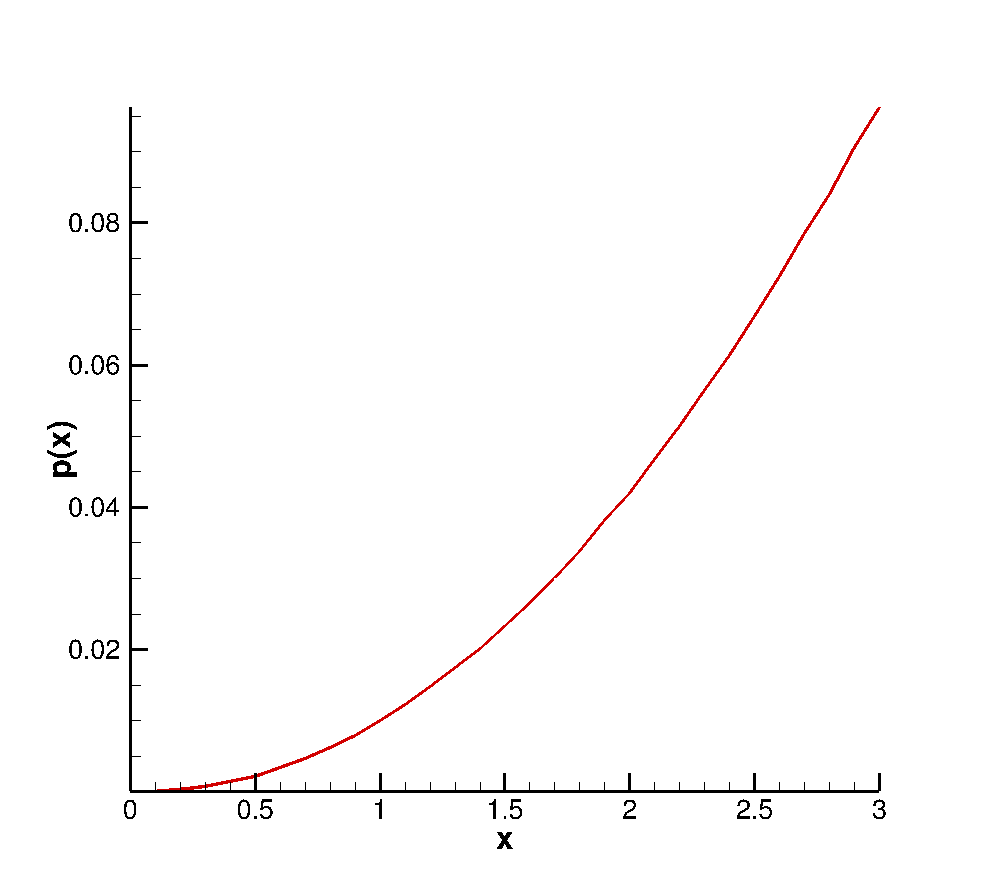
END PROGRAM flushPokerHand

**Problem 3:**

Distribution function:

Probability density function (PDF) will be given by:

FFMC:



Source code:

PROGRAM randDist

IMPLICIT NONE

INTEGER :: iter, bin(30), i, maxIter, unitProbFile

REAL :: power, randNum, x, xBin, pdf(30)

CHARACTER(80) :: nameProbFile

! Start

PRINT\*, 'Enter maximum number of iterations'

READ\*, maxIter

CALL random\_seed()

power = 1./3.

iter = 0

bin(:) = 0

DO WHILE (iter .lt. maxIter)

iter = iter + 1

CALL RANDOM\_NUMBER(randNum)

x = 3.0\*randNum\*\*(power)

xBin = CEILING(x\*10.0)

bin(xBin) = bin(xBin) + 1

END DO

! Compute probability of flush hand

pdf(:) = 1.0\*bin(:)/maxIter

unitProbFile = 101

nameProbFile = 'pdf\_x2.dat'

OPEN (UNIT = unitProbFile, FILE = nameProbFile, &

POSITION = 'rewind', FORM = 'formatted', ACTION = 'write')

DO i = 1, 30

WRITE(unitProbFile,501) i, pdf(i)

END DO

501 FORMAT (i3.3, 1X, 1(e12.5, 1X))

END PROGRAM randDist