Projeto 2 - Segundo Relatório

Instituto de Física de São Carlos
Universidade de São Paulo

Gabriel Lima Alves (12558547)

Introdução à Física Computacional
Prof. Francisco Castilho Alcaraz

Setembro, 2022

Tarefa A

Nessa tarefa o objetivo era testar a geração de números aleatórios calculando alguns momentos da distribuição aleatórios. Essa distribuição é dada por:

$$< x^n >, \text{ para n} = 1,\ 2,\ 3,\ 4.$$

O algoritmo utilizado para esses cálculos pode ser visto abaixo, nele os números aleatórios gerados são reais e variam dentro do seguinte intervalo $]0, 1[$. Além disso, o parâmetro m define o números de números gerados para cada distribuição. A saída do algoritmo também pode ser vista abaixo, analisando a saida é possível verificar que ela é equivalente ao que era esperado.

```fortran
c      program tarefa-1

       Parameter(n = 4)
       Parameter(m = 1000)
       resp = 0.e0
       rr = rand(iseed)

       do i = 1,n
         resp = 0
         do j = 1,m
             rr = rand()
             resp = resp + rr**i
         end do
         write(*,3) i, resp/m
3        format('<rr**', i0,'>: ', e0.10)
       end do
       end
```

Algoritmo 1: Código para resolução da tarefa A

Figura 1: testes do código realizados

Tarefa B1

Na tarefa B1, foi calculado um número *it_ and* de andarilhos deslocando se *it_ n* passos, com as probabilidades p e q iguais a $\frac{1}{2}$ (para mudar o número de andarilhos ou passos deve se alterar o valor das variáveis diretamente no algoritmo). Depois, foi calculado a posição média ($< x >$) e a posição quadrática média ($< x^2 >$) que é impresso no terminal. Além disso, também é gerado um historiograma da distribuição dos andarilhos em função de sua posição

No algoritmo, para o calculo das posições dos andarilhos são usados números aleatórios que são manipulados de modo que estes variam entre os valores $\{1, 2\}$. Já no calculo do histograma, é encontrado o andarilhos com a menor posição e o com a maior posição e partindo desse intervalo são feitas janelas que tem tamanho *rint*. O algoritmo pode ser visto abaixo.

```fortran
c      program tarefa-2

       Parameter(it_n = 10000)
       Parameter(it_and = 1000)
       dimension imatm(it_n)
       dimension ip(2)
       Parameter(ip = (/-1,1/))
       Parameter(ient = 10)

       rmedx = 0e0
       rmedx2 = 0e0
       r = rand(iseed)
       do i=1,it_and
         n = 0
         do k=1,it_n
             r = rand()*2
             j = int((r+1)/2)+1
             n = n + ip(j)
         end do
         imatm(i) = n
         rmedx = rmedx + n
         rmedx2 = rmedx2 + n*n
       end do

       write(*,*) '<x>: ', (rmedx/it_and), '<x^2>',(rmedx2/it_and)

       min = imatm(1)
       max = min
       do i = 2,it_and
         if (.NOT. (min < imatm(i))) then
             min = imatm(i)
         end if
         if (.NOT. (max > imatm(i))) then
             max = imatm(i)
         end if
       end do

       min = min -1
       max = max +1

       amp = max - min
       jan = 10
       rint = amp/jan
       aux = min
       open(unit=ient,file='saida-1-12558547.dat')
```

```fortran
      do k = 1,jan
        rvalcolum = 0e0
        do i = 1,it_and
            if (imatm(i)>=aux .and. imatm(i)< aux+rint) then
                rvalcolum = rvalcolum + 1
            end if
        end do
        write(ient,*) aux, rvalcolum
        aux = aux + rint
      end do
      close(ient)

      end
```

Algoritmo 2: Código para resolução da tarefa B1

A saída do algoritmo está abaixo, bem como o gráfico gerado a partir do histograma cuja curva se aproximou de uma distribuição normal.

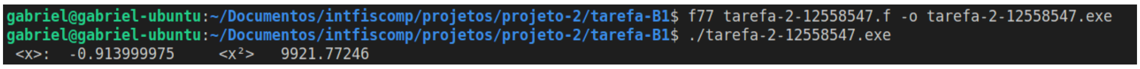


Figura 2: testes do código realizados

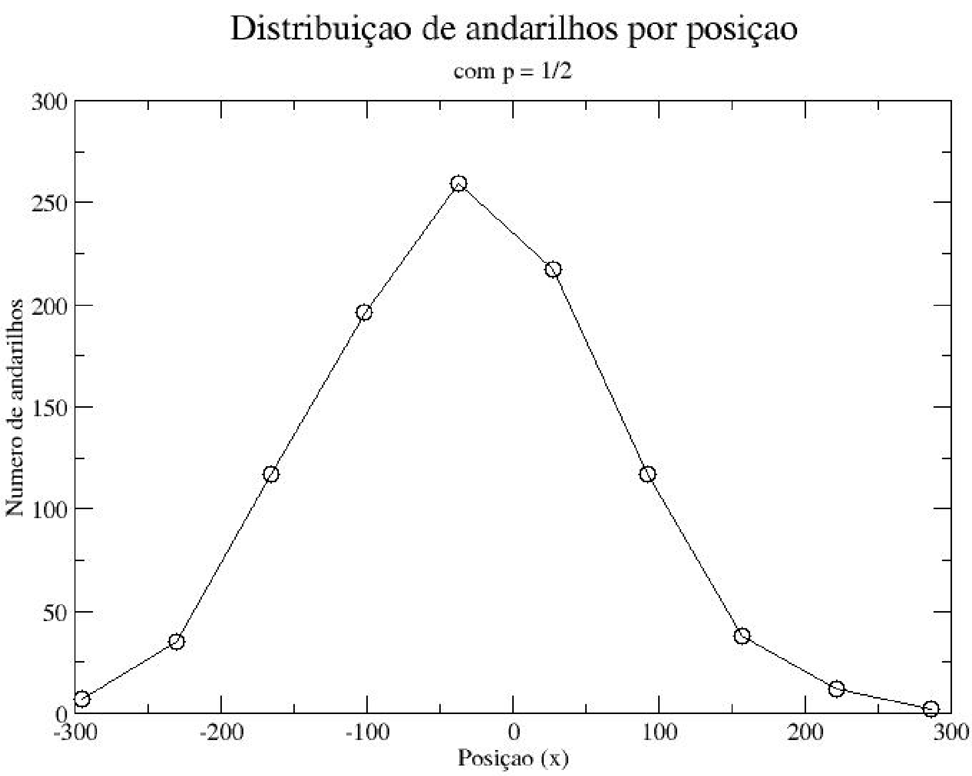


Figura 3: gráfico plotado no Xmgrace

Tarefa B2

Nessa tarefa o algoritmo utilizado é basicamente igual ao anterior, a diferença é que foi adicionado um parâmetro p cujo valor deve ser o inverso da probabilidade p que se pretende calcular. Ou seja, quando o parâmetro p vale 3 a probabilidade do andarilho dar um passo para direita é igual a $p = \frac{1}{3}$ e para esquerda é igual $q = \frac{2}{3}$, pois p+q = 1.

Assim, utilizando o parâmetro $p = 3$ no lugar da constante 2 é possível fazer os números aleatórios variarem entre os valores $\{1, 1, 2\}$, de modo que a chance de andar para esquerda, representado pelo valor 1, é $\frac{2}{3}$. Essa mesma logica é aplicada para p = 4 e 5, nos quais os números aleatórios iriam variar, respectivamente, $\{1, 1, 1, 2\}$ e $\{1, 1, 1, 1, 2\}$.

A seguir, está o algoritmo utilizado, bem como os gráficos e a saída do algoritmo para p = 3,4 e 5; para outros valores de p deve se variar o parâmetro p no algoritmo.

```
c      program tarefa-3

       Parameter(it_n = 10000)
       Parameter(it_and = 1000)
       dimension imatm(it_n)
       dimension ip(2)
       Parameter(ip = (/-1,1/))
       Parameter(p = 3)
       Parameter(ient = 10)

       rmedx = 0e0
       rmedx2 = 0e0
       r = rand(iseed)
       do i=1,it_and
         n = 0
         do k=1,it_n
             r = rand()*p
             j = int((r+1)/p)+1
             n = n + ip(j)
         end do
         imatm(i) = n
         rmedx = rmedx + n
         rmedx2 = rmedx2 + n*n
       end do

       write(*,*) '<x>: ', (rmedx/it_and), '<x^2>',(rmedx2/it_and)

       min = imatm(1)
       max = min
       do i = 2,it_and
         if (.NOT. (min < imatm(i))) then
             min = imatm(i)
         end if
         if (.NOT. (max > imatm(i))) then
             max = imatm(i)
         end if
```

```fortran
      end do

      min = min -1
      max = max +1

      amp = max - min
      jan = 10
      rint = amp/jan
      aux = min
      open(unit=ient,file='saida-2-12558547.dat')
      do k = 1,jan
        rvalcolum = 0e0
        do i = 1,it_and
            if (imatm(i)>=aux .and. imatm(i)< aux+rint) then
                rvalcolum = rvalcolum + 1
            end if
        end do
        write(ient,*) aux, rvalcolum
        aux = aux + rint
      end do
      close(ient)

      end

```

Algoritmo 3: Código para resolução da tarefa B2

```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-B2$ f77 tarefa-3-12558547.f -o tarefa-3-12558547.exe
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-B2$ ./tarefa-3-12558547.exe
 <x>:    -3331.26392      <x²>    11105442.0
```

Figura 4: saída do algoritmo para p = 1/3

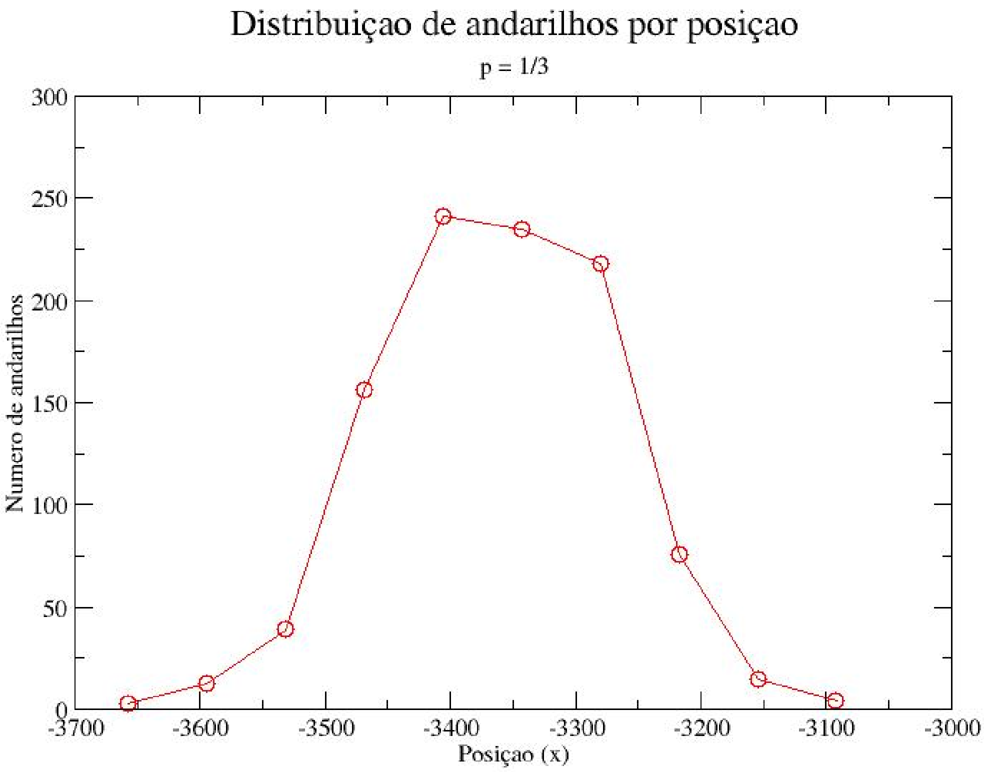


Figura 5: gráfico plotado no Xmgrace para p = 1/3

```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-B2$ f77 tarefa-3-12558547.f -o tarefa-3-12558547.exe
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-B2$ ./tarefa-3-12558547.exe
 <x>:    -4999.39600      <x²>    25001368.0
```

Figura 6: saída do algoritmo para p = 1/4

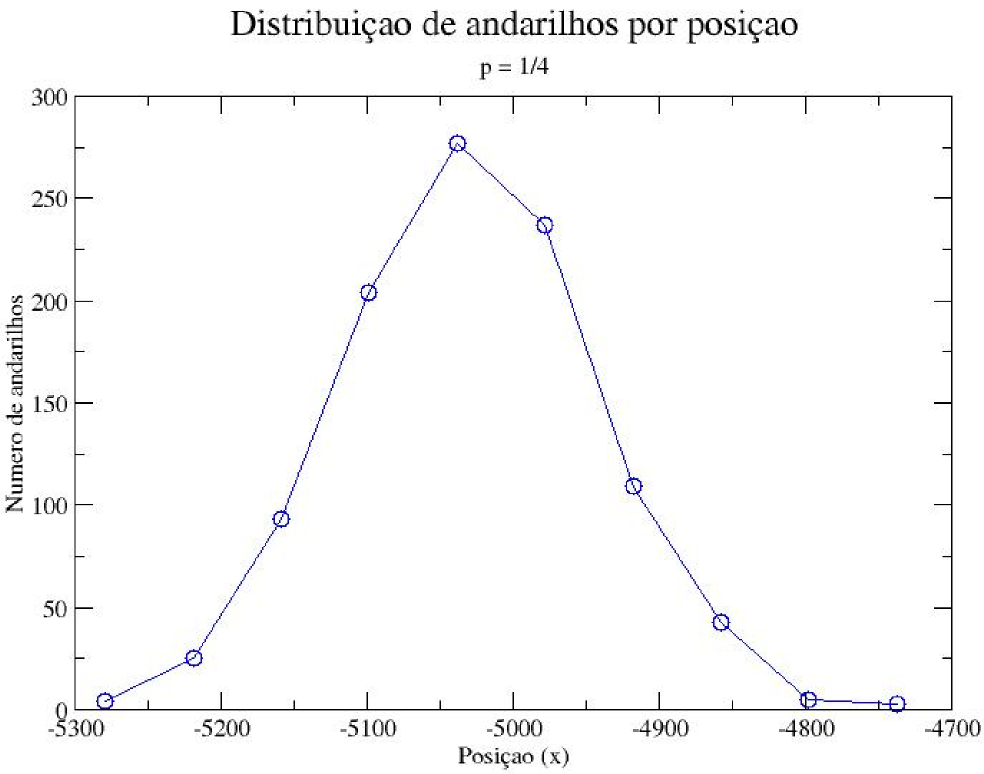
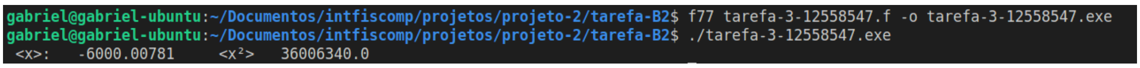


Figura 7: gráfico plotado no Xmgrace para p = 1/4



```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-B2$ f77 tarefa-3-12558547.f -o tarefa-3-12558547.exe
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-B2$ ./tarefa-3-12558547.exe
 <x>:    -6000.00781      <x²>    36006340.0
```

Figura 8: saída do algoritmo para p = 1/5

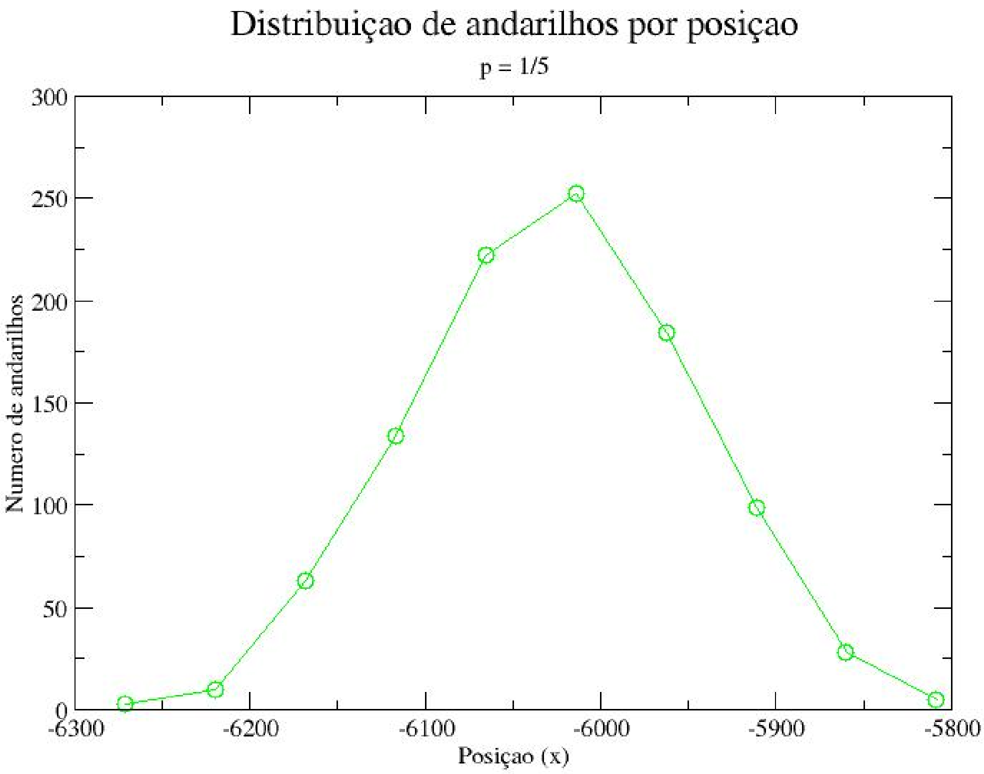


Figura 9: gráfico plotado no Xmgrace para p = 1/5

Tarefa C

Na tarefa C foi feito uma generalização dos algoritmos dos itens anterior para duas dimensões. Assim, foi calculado um número it_and de andarilhos deslocando se it_n passos nos sentidos norte, sul, leste e oeste. Para cada sentido a probabilidade do andarilho ir no sentido é igual a $\frac{1}{4}$ (para mudar o número de andarilhos ou passos deve se alterar o valor das variáveis diretamente no algoritmo).

Depois, foi calculado as coordenadas x e y médias e em seguida a distancia desse ponto médio à orgiem (0,0), ($< r >$). Além disso foi calculado o Δ^2 que é definido como: $\Delta^2 =< r^2 > - < r >^2$. Por ultimo, também foi gerado um gráfico da posição final de cada andarilho no espaço.

No algoritmo, para o calculo das posições dos andarilhos são usados números aleatórios que são manipulados de modo que estes variam entre os valores $\{1, 2\}$. Assim, utilizando os números aleatórios duas vezes é possível chegar na probabilidade $\frac{1}{4}$, pois haverá $\frac{1}{2}$ de chance do andarilho caminhas na direção norte/sul ou leste/oeste e mais $\frac{1}{2}$ de chance dele andar no sentido (norte - sul) ou (leste - oeste). O algoritmo pode ser visto abaixo, bem como os gráficos e saídas para $it\_n = 10, 100, 10^3, 10^4, 10^5$, (não foi feito o calculo para $it\_n = 10^6$ pois é gerado um erro).

```
c       program tarefa-4

        Parameter(it_n = 10)
```

```
      Parameter(it_and = 1000)
      dimension rmatm_xy(it_and,2)
      dimension ip(2)
      Parameter(ip = (/-1,1/))
      Parameter(p = 2)
      Parameter(ient = 10)

      rx = 0
      ry = 0
      rx2 = 0
      ry2 = 0
      r = rand(iseed)
      do i=1,it_and
        rmatm_xy(i,1) = 0
        rmatm_xy(i,2) = 0
        do k=1,it_n
            r = rand()*p
            j = int((r+1)/p)+1
            r = rand()*p
            m = int((r+1)/p)+1
            rmatm_xy(i,j) = rmatm_xy(i,j) + ip(m)
        end do
        rx = rmatm_xy(i,1) +rx
        ry = rmatm_xy(i,2) +ry
        rx2 = rmatm_xy(i,1)**2 +rx2
        ry2 = rmatm_xy(i,2)**2 +ry2
      end do
      rmedx = (rx/it_and)**2 +(ry/it_and)**2
      rmedx2 = (sqrt(rx2/it_and)**2 + (ry2/it_and)**2) - rmedx
      rmedx = sqrt(rmedx)

      write(*,*) '<r>: ', rmedx, '<²>: ', rmedx2

      open(unit=ient,file='saida-3-12558547.dat')
      do i = 1, it_and
        write(ient,*) rmatm_xy(i,1), rmatm_xy(i,2)
      end do
      close(ient)

      end

```

Algoritmo 4: Código para resolução da tarefa C

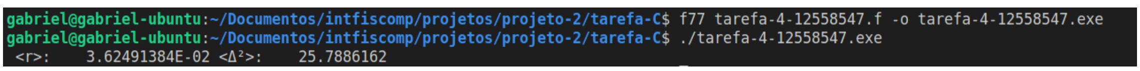


Figura 10: saída do algoritmo para n = 10

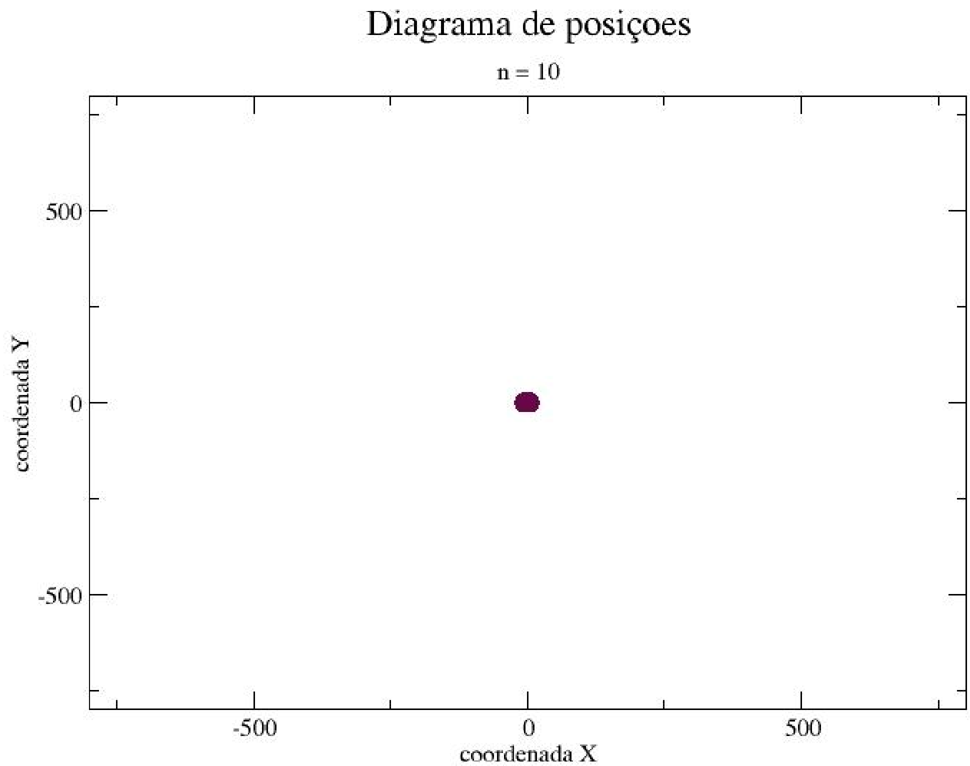


Figura 11: gráfico plotado no Xmgrace para para n = 10

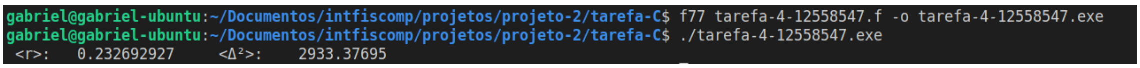


Figura 12: saída do algoritmo para n = 10^2

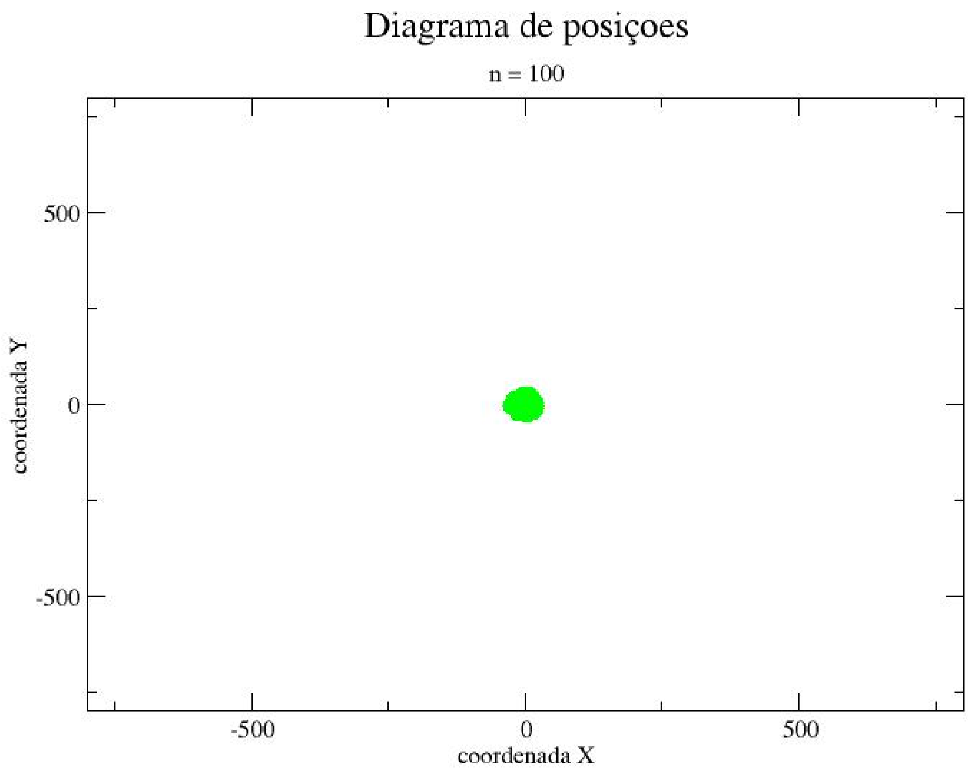
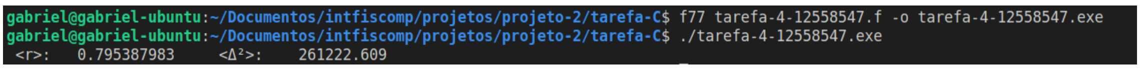


Figura 13: gráfico plotado no Xmgrace para para n $= 10^2$



```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-C$ f77 tarefa-4-12558547.f -o tarefa-4-12558547.exe
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-C$ ./tarefa-4-12558547.exe
 <r>:   0.795387983      <Δ²>:    261222.609
```

Figura 14: saída do algoritmo para n $= 10^3$

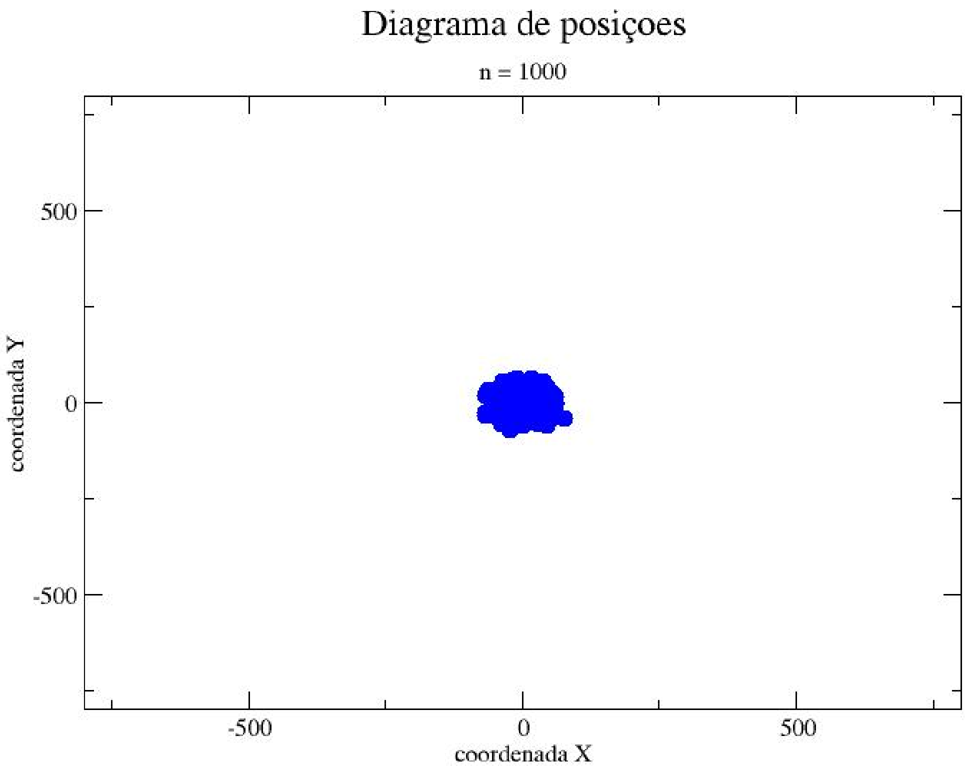


Figura 15: gráfico plotado no Xmgrace para para n = 10^3

```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-C$ f77 tarefa-4-12558547.f -o tarefa-4-12558547.exe
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-C$ ./tarefa-4-12558547.exe
 <r>:    1.22244430      <Δ²>:    25881040.0
```

Figura 16: saída do algoritmo para n = 10^4

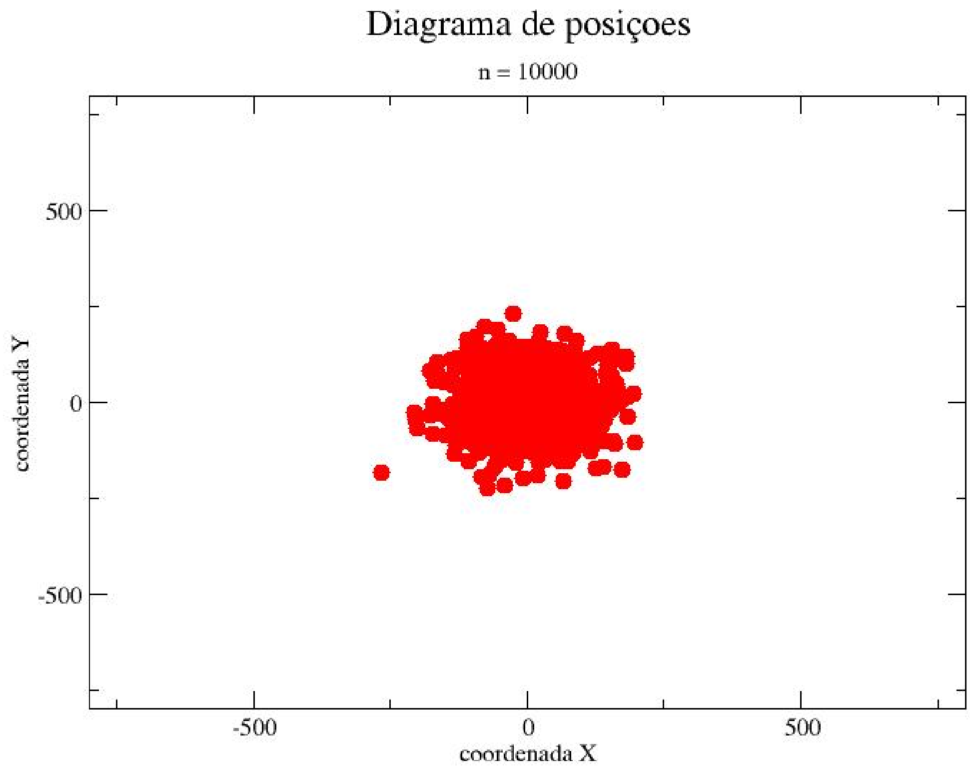


Figura 17: gráfico plotado no Xmgrace para para n $= 10^4$

```
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-C$ f77 tarefa-4-12558547.f -o tarefa-4-12558547.exe
gabriel@gabriel-ubuntu:~/Documentos/intfiscomp/projetos/projeto-2/tarefa-C$ ./tarefa-4-12558547.exe
 <r>:     6.69403601      <Δ²>:     2.53313280E+09
```

Figura 18: saída do algoritmo para n $= 10^5$

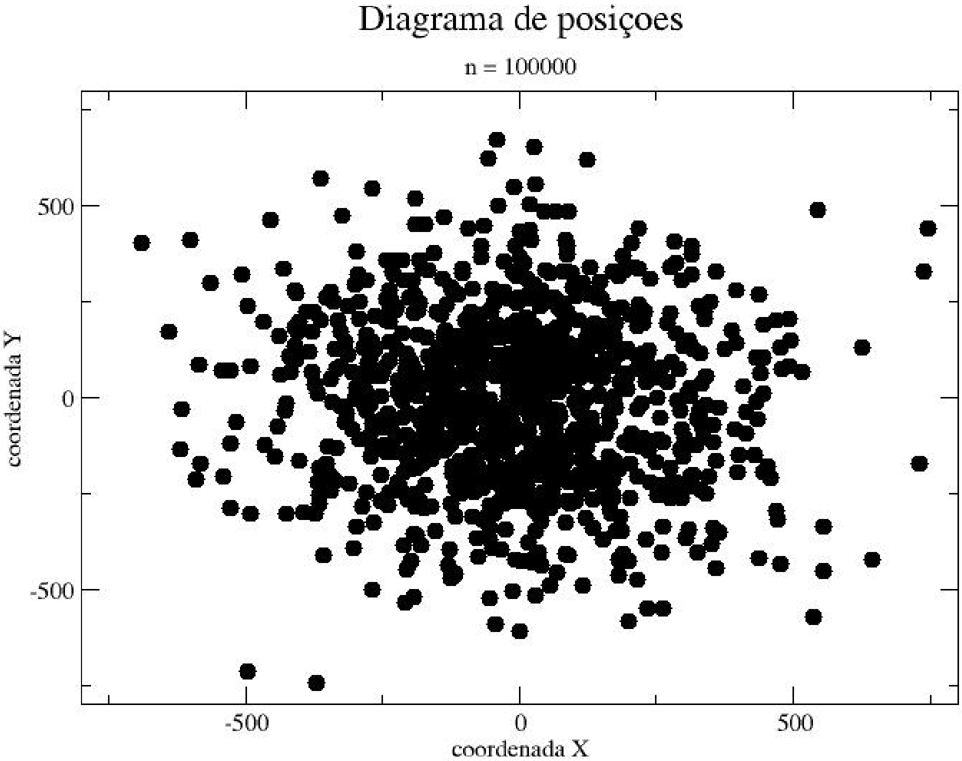


Figura 19: gráfico plotado no Xmgrace para para n = 10^5

Tarefa D

Nesta tarefa, o objetivo é calcular o aumento da entropia a medida que o número de passos dados pelos andarilhos aumentam. O algoritmo utilizado para essa tarefa é similar ao anterior, um plano bidimensional com iguais chances do andarilho dar um passo para cada sentido (para mudar o número de andarilhos ou passos deve se alterar o valor das variáveis diretamente no algoritmo).

Para o calculo da entropia foi utilizado a seguinte formula:

$$S = -\sum_{i}^{N} P_i ln(P_i)$$

onde P_i é a probabilidade de se encontrar o sistema em um certo "micro-estado"i.

No algoritmo o "micro-estado"i foi definido como um quadrado de área $iaux \cdot iaux$ em que o espaço total foi dividido. Assim, a probabilidade de cada P_i foi definido como:

$$P_i = \frac{\text{Quantidade de andarilhos no quadrado}}{Quantidadetotaldeandarilhos}$$

Dessa forma, a partir dessa relação foi possível calcular a variação da entropia em função do aumento de passos. O algoritmo utilizado para esses cálculos pode ser visto abaixo, bem como o gráfico da entropia.

```
c      program tarefa-5
      parameter (it_and = 1000)
      parameter (it_n = 1000)
      dimension imatm_xy(it_and, 2)
      dimension ip(2)
      Parameter(ip = (/-1,1/))
      Parameter(ient = 10)
      Parameter(p = 2)

      rnd = rand(iseed)

      do j = 1, it_and
         imatm_xy(j, 1) = 0
         imatm_xy(j, 2) = 0
      end do

      open(unit=ient,file='saida-5-12558547.dat')
      i = 0
      do while(i<it_n)
        i = i +1

         do j=1,it_and
            r = rand()*p
            k = int((r+1)/p)+1
            r = rand()*p
            m = int((r+1)/p)+1
            imatm_xy(j,k) = imatm_xy(j,k) + ip(m)
        end do

         ixmin = imatm_xy(1, 1)
         ixmax = ixmin
         iymin = imatm_xy(1, 2)
         iymax = iymin

         do j = 2, it_and
            if (.NOT. (ixmin < imatm_xy(j, 1))) then
               ixmin = imatm_xy(j, 1)
            end if
            if (.NOT. (ixmax > imatm_xy(j, 1))) then
               ixmax = imatm_xy(j, 1)
            end if
            if (.NOT. (iymin < imatm_xy(j, 2))) then
               iymin = imatm_xy(j, 2)
            end if
            if (.NOT. (iymax > imatm_xy(j,2))) then
               iymax = imatm_xy(j, 2)
            end if
         end do
```

```
      entropy = 0e0
      iaux = 5

      do ix = ixmin, ixmax, iaux
         do iy = iymin, iymax, iaux
            count = 0e0
            do j = 1, it_and
               iposX = imatm_xy(j,1)
               iposY = imatm_xy(j,2)
               if(((iposX <= ix + iaux) .and. (iposX >= ix)) .and.
     +                 (iposY <= iy + iaux .and. iposY >= iy)) then
                  count = count + 1
               end if
            end do
            if(count /= 0) then
               prob = count / it_and
               entropy = entropy - prob * log(prob)
            end if
         end do
      end do
      write(ient, *) i, entropy
   end do
   close(ient)
   end
```

Algoritmo 5: Código para resolução da tarefa D

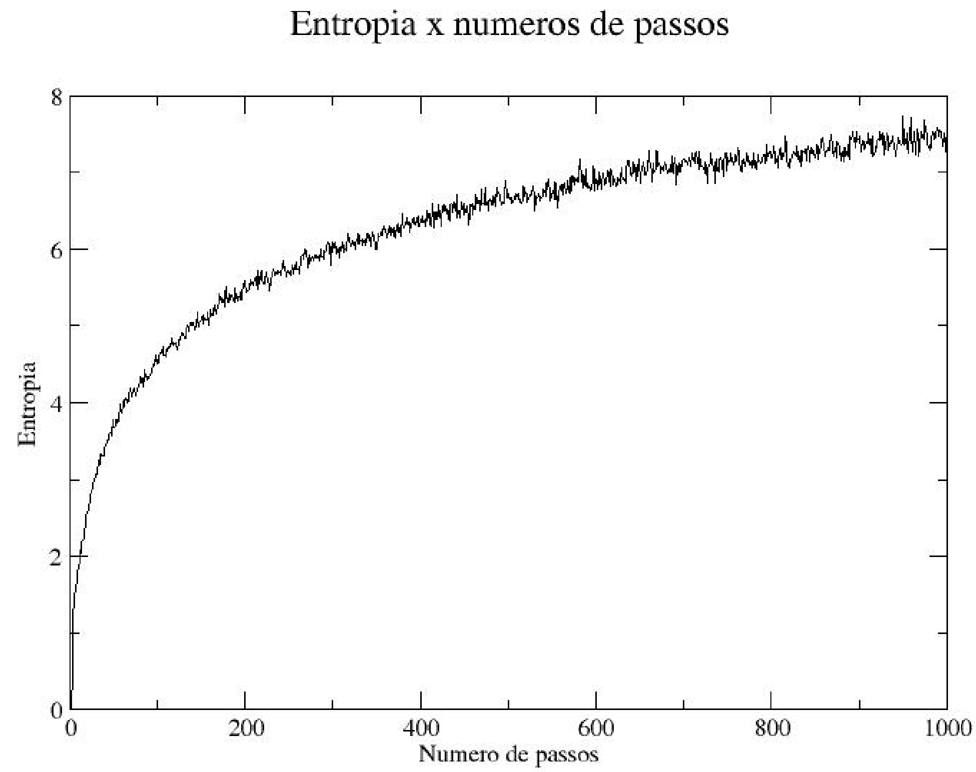


Figura 20: gráfico plotado no Xmgrace