

Universidade de São Paulo

Instituto de Física de São Carlos

Projeto 1

Pedro Calligaris Delbem 5255417

Professor: Francisco Alcaraz

Setembro de 2023

Sumário

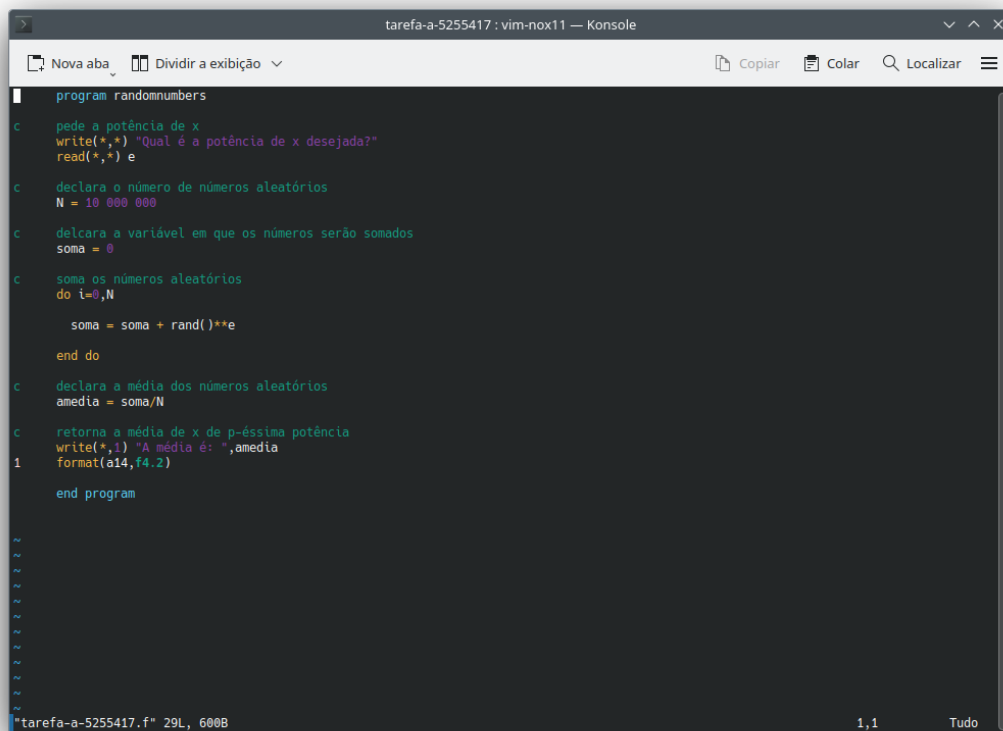
1	Tarefa A	2
2	Tarefa B	4
2.1	B.1	4
2.2	B.2	7
3	Tarefa C	13
4	Tarefa D	22

1 Tarefa A

Tarefa: Escrever um código em FORTRAN77 que calcule, dado um N, a média da distribuição pseudoaleatória gerada pela função rand() intrínseca da linguagem. Ou seja, calcular

$$\langle x^N \rangle \quad (1)$$

Código escrito:



```
program randomnumbers
c  pede a potência de x
write(*,*) "Qual é a potência de x desejada?"
read(*,*) e

c  declara o número de números aleatórios
N = 10 000 000

c  declara a variável em que os números serão somados
soma = 0

c  soma os números aleatórios
do i=0,N
    soma = soma + rand()**e
end do

c  declara a média dos números aleatórios
amedia = soma/N

c  retorna a média de x de p-éssima potência
write(*,1) "A média é: ",amedia
1 format(a14,f4.2)

end program
```

Descrição:

O código pede ao usuário qual será a potência dos números médios e salva o valor fornecido na variável "e".

Em seguida, declara "N" como 10 milhões que será a quantidade de números aleatórios gerados, além de iniciar "soma" - que será a soma dos mesmos - como 0.

Após isso, inicia um loop indo de 0 até "N" onde adiciona-se à "soma" um número aleatório - gerado por rand() - elevado à variável "e".

Por fim, divide-se a soma total por "N", calculando a média - salvando em "amedia" - desejada. Assim, imprime-se o resultado na tela com mensagem "A média

: amedia”, utilizando format para imprimir apenas duas casas decimais e controlar o espaçamento entre a variável e o texto.

Saídas: É esperado que calcular tal média seja equivalente à calcular a integral de x^N de 0 até 1. Assim compara-se a saída para $N = 1, 2, 3, 4$ com o resultados das integrais.

N=1:

```
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-a-5255417> ./tarefa-a-5255417.exe
Qual é a potência de x desejada?
1
A média é: 0.50
```

Como a integral de x é x^2 calculando de 0 a 1, obtêm-se 0.5 que corresponde com o valor obtido.

N=2:

```
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-a-5255417> ./tarefa-a-5255417.exe
Qual é a potência de x desejada?
2
A média é: 0.33
```

Como a integral de x^2 é x^3 calculando de 0 a 1, obtêm-se $1/3$ que corresponde com o valor obtido.

N=3:

```
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-a-5255417> ./tarefa-a-5255417.exe
Qual é a potência de x desejada?
3
A média é: 0.25
```

Como a integral de x^3 é x^4 calculando de 0 a 1, obtêm-se 0.25 que corresponde com o valor obtido.

N=4:

```
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-a-5255417> ./tarefa-a-5255417.exe
Qual é a potência de x desejada?
4
A média é: 0.20
```

Como a integral de x^4 é x^5 calculando de 0 a 1, obtêm-se 0.2 que corresponde com o valor obtido.

2 Tarefa B

Tarefa: Escrever um código em FORTRAN77 que utilizando a função `rand()` intrínseca da linguagem, determine a distribuição de N andarilhos aleatórios - após 1000 passos -, onde caso `rand()` retorne um valor maior que p o andarilho dará uma passo a direita e caso seja menor que p dará um passo a esquerda. O código deve retornar $\langle x \rangle$ e $\langle x^2 \rangle$ dos andarilhos, além de um histograma da suas posições finais.

Deve ser feito um código para $p=1/2$ e um outro código onde p é fornecido pelo usuário.

2.1 B.1

(Caso $p=1/2$)

Código escrito:

```
tarefa-b-5255417: vim-nox11 — Konsole
Nova aba  Dividir a exibição  Copiar  Colar  Localizar  ☰

program andarilho bebado
c  cria os vetores que armazenarão a posição e passos dos andarilhos
dimension thisto(-1000:1000), ipx(0:1)
c  limpa o vetor que armazena a posição dos andarilhos
thisto(-1000:1000) = 0
c  define o número de andarilhos
m = 500000
c  define as probabilidades
ap = 0.5
c  define o número de passos
n = 1000
c  define as somas
soma1 = 0
soma2 = 0
c  define os passos possíveis
ipx(0) = 1
ipx(1) = -1
c  define o loop dos andarilhos
do i=1,m
c  define a posição do andarilho
ix = 0
c  define o loop de cada andarilho
do j=1,n
c  faz a divisão inteira de i por ap
int = rand(0)/ap
c  adiciona o passo de acordo com o valor retornado por ipx()
ix = ix + ipx(int)
end do
thisto(ix) = ihisto(ix) + 1
soma1 = soma1 + ix
soma2 = soma2 + (ix)**2
end do
c  aloca a memória para salvar os dados do histograma
open(unit=1,file='histograma')
c  inicia o loop para salvar as informações no histograma
do i=-1000,1000
j = ihisto(i)
write(1,*) i,j
end do
c  fecha a unidade de memória
close(1)
c  define as médias
amedia1 = soma1/m
amedia2 = soma2/m
write(*,*) "As médias de potência 1 e 2 são:",amedia1,amedia2
end program
~
~
"tarefa-b1-5255417.f" 69L, 1579B 1,1 Tudo
```

Descrição:

O código defini um vetor indexado de -1000 até 1000 - que armazenará a quantidade de andarilhos em cada posição e um vetor com duas posições (0 e 1) - que determinará a direção do passo.

Em seguida, é definido o número de andarilhos ($m = 500\,000$), a probabilidade de andar para a direita ($ap = 0.5$), o número de passos ($n = 1000$) e as somas das posições ($soma1 = 0$, $soma2 = 0$).

Após isso, o código defini $ipx(0) = 1$ e $ipx(1) = -1$ que definirá mais tarde a direção da cada passo.

Ademais, o programa inicia um loop de 1 até m - ou seja, uma iteração para cada andarilho. Definindo, como 0, a posição inicial ($ix=0$) inicia-se um loop individual de cada andarilho indo de 1 até n (número de passos) no qual cada iteração definirá a direção do movimento da partícula.

Ao salvar a divisão inteira de $\text{rand}(0)$ por ap em int , obtêm-se 1 caso $\text{rand}(0)$ seja maior que ap e 0, caso contrário. Assim ao somar $ipx(int)$ em ix obtêm-se o efeito de um passo à esquerda ou à direita.

Ao sair, do loop de cada andarilho, é adicionado 1 em $ihisto(ix)$, de modo a contabilizar quantos andarilhos ficaram em cada posição ix .

No fim do loop, adiciona-se ix à $soma1$, para se contabilizar a posição média. E soma-se ix^2 para se contabilizar a posição média quadrada.

Por fim, o código aloca uma unidade de memória com o nome 'histograma-5255417' para salvar o histograma das posições das partículas. Assim, utilizando um loop de -1000 até 1000, impimi-se na arquivo a posição e o número de partículas correspondente. Fecha-se a memória.

Dividindo $soma1$ por m e $soma2$ por m , obtêm-se a média e a média quadrada das posições, imprimindo-as na tela.

Saídas:

Rodando o código obte-se a seguinte saída no terminal:

```
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-b-5255417> ./tarefa-b1-5255417.exe
As médias de potência 1 e 2 são: -9.99199972E-03  997.967468
```

Que são resultados razoáveis, pois não há qualquer razão para que a posição seja mais ou menos à direita, de modo que algo próximo de 0 é o esperado. Além disso, a posição média quadrada deve ser positiva e se aproximar do número de passos dados, uma vez que um passo positivo não anula um negativo - nesta conta.

Plotando o histograma com Xmgrace, obteve-se o seguinte gráfico:

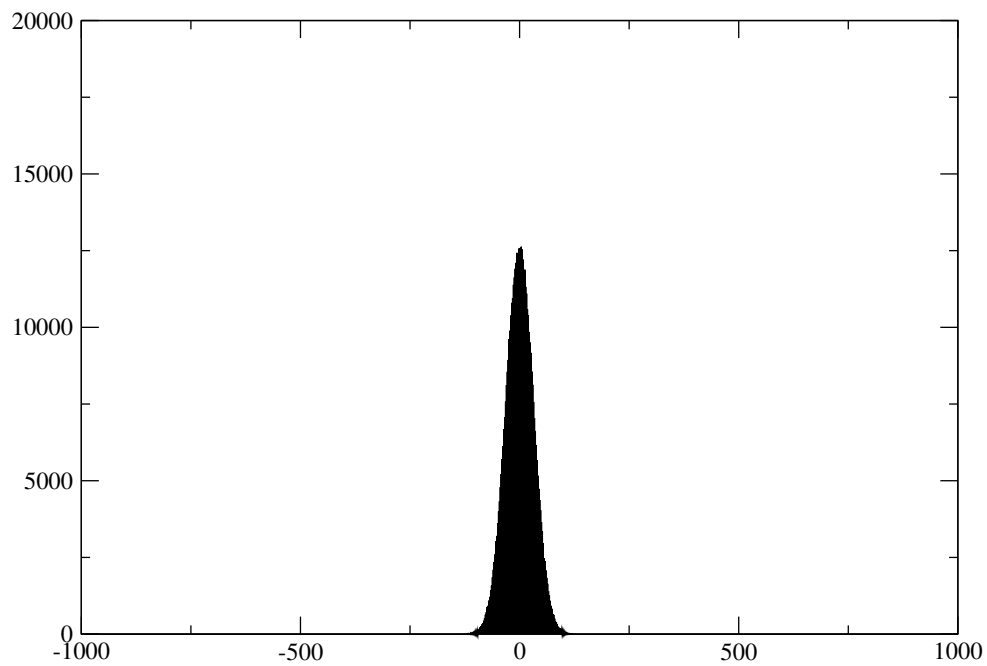


Gráfico: Número de partículas X Posição

Que representa uma distribuição razoável para uma movimentação completamente aleatória.

2.2 B.2

(p fornecido pelo usuário)

Código escrito:


```
tarefa-b-5255417: vim-nox11 — Konsole
Nova aba  Dividir a exibição  Copiar  Colar  Localizar  ☰

program andarilho bebado na rampa
c  cria os vetores que armazenarão a posição e passos dos andarilhos
c  e a variável que define o nome do arquivo de saída
dimension ihisto(-1000:1000), ipx(0:10000)
character*30 name

c  limpa o vetor que armazena a posição dos andarilhos
ihisto(-1000:1000) = 0

c  define o número de andarilhos
m = 500000

c  define as probabilidades
write(*,*) "Qual é o valor de p desejado, onde p = 1/x?"
read(*,*) x
ap = 1.0e0/x

c  define o nome do arquivo
name = 'histograma--5255417'

c  define o número de passos
n = 1000

c  define as somas
soma1 = 0
soma2 = 0

c  define os passos possíveis
ipx(0) = 1
ipx(1:10000) = -1

c  define o loop dos andarilhos
do i=1,m

c  define a posição do andarilho
ix = 0
c  define o loop de cada andarilho
do j=1,n

c  faz a divisão inteira de i por ap
int = rand(0)/ap

c  adiciona o passo de acordo com o valor retornado por ipx()
ix = ix + ipx(int)

end do

ihisto(ix) = ihisto(ix) + 1
soma1 = soma1 + ix
soma2 = soma2 + (ix)**2

end do

c  aloca a memória para salvar os dados do histograma
open(unit=1,file=name)

c  inicia o loop para salvar as informações no histograma
do i=-1000,1000

j = ihisto(i)
write(1,*) i,j

end do

c  fecha a unidade de memória
close(1)

c  define as médias
amedia1 = soma1/m
amedia2 = soma2/m

write(*,*) "As médias de potência 1 e 2 são:",amedia1,amedia2

end program
"tarefa-b2-5255417.f" 75L, 1710B 1,1 Tudo
```

Descrição:

O código defini um vetor indexado de -1000 até 1000 - que armazenará a quantidade de andarilhos em cada posição, um vetor indexado de 0 até 10000 - que determinará a direção do passo e um character*30 que guardará o nome do arquivo de saída

Em seguida, é definido o número de andarilhos ($m = 1000$). E defini "name='histograma-5255417'" (que foram renomeados de acordo com o p correspondente, i.e., 'histograma-3-5255417' se $p = 1/3$).

Após isso o programa pede ao usuário qual será o x se a probabilidade de dar um passo à direita for $1/x$ e salva a probabilidade ($ap = 1/x$) e também salva as somas das posições ($soma1 = 0$, $soma2 = 0$).

Após isso, o código defini $ipx(0) = 1$ e $ipx(1:10000) = -1$ que definirá mais tarde a direção da cada passo.

Ademais, o programa inicia um loop de 1 até m - ou seja, uma iteração para cada andarilho. Definindo, como 0, a posição inicial ($ix=0$) inicia-se um loop individual de cada andarilho indo de 1 até n (número de passos) no qual cada iteração definirá a direção do movimento da partícula.

Ao salvar a divisão inteira de $rand(0)$ por ap em int , obtêm-se um número maior ou igual a 1 caso $rand(0)$ seja maior que ap e 0, caso contrário. Assim ao somar $ipx(int)$ em ix obtêm-se o efeito de um passo à esquerda ou à direita.

Ao sair, do loop de cada andarilho, é adicionado 1 em $ihisto(ix)$, de modo a contabilizar quantos andarilhos ficaram em cada posição ix .

No fim do loop, adiciona-se ix à $soma1$, para se contabilizar a posição média. E soma-se ix^2 para se contabilizar a posição média quadrada.

Por fim, o código aloca uma unidade de memória com o nome "name" para salvar o histograma das posições das partículas. Assim, utilizando um loop de -1000 até 1000, impime-se no arquivo a posição e o número de partículas correspondente. Fecha-se a memória.

Dividindo $soma1$ por m e $soma2$ por m , obtêm-se a média e a média quadrada das posições, imprimindo-as na tela.

Saídas:

Os gráficos obtidos para $p = 1/3, 1/4$ e $1/5$ foram:

($p = 1/3$)

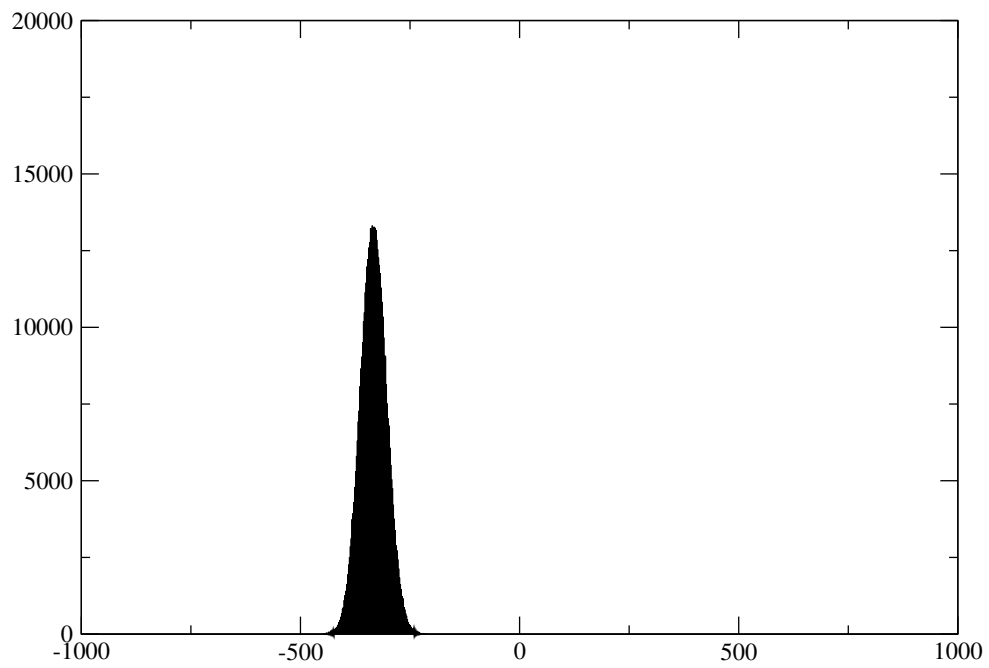


Gráfico: Número de partículas X Posição

($p = 1/4$)

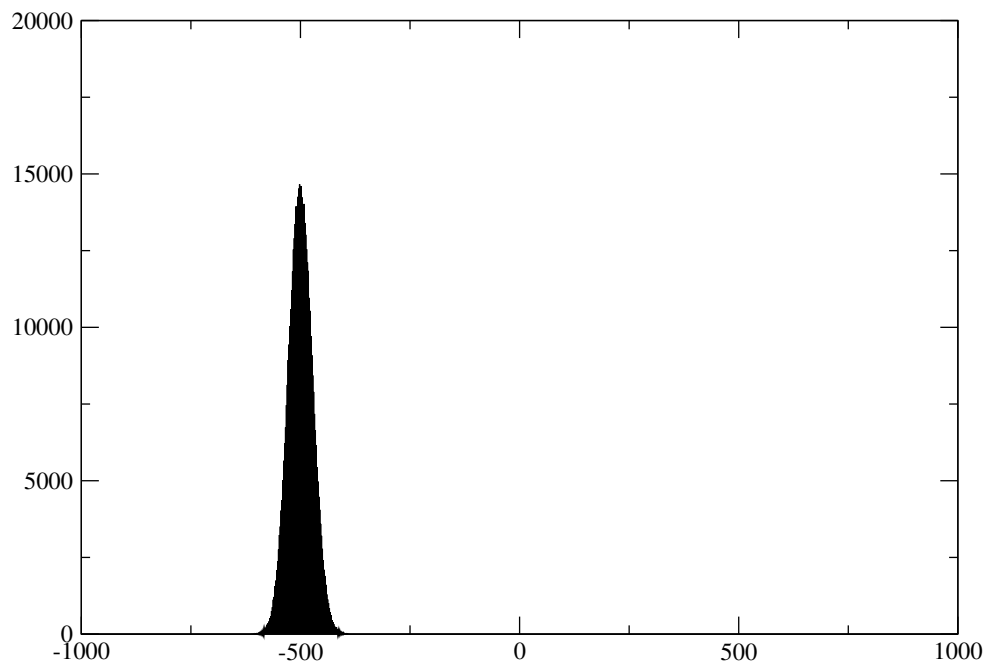


Gráfico: Número de partículas X Posição

($p = 1/5$)

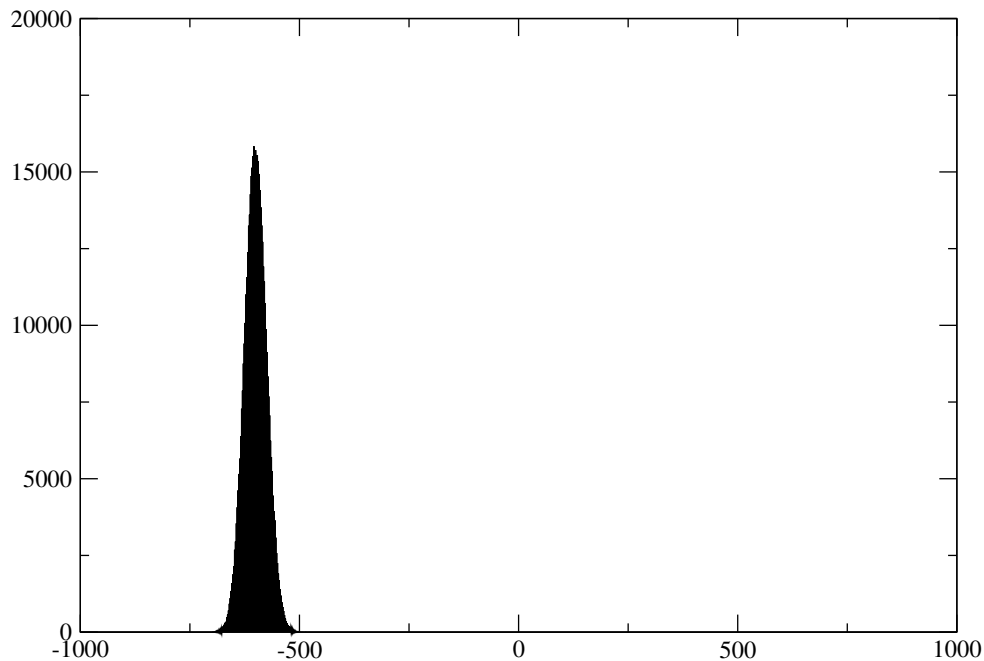


Gráfico: Número de partículas X Posição

Analisando os gráficos, percebe-se que os mesmos correspondem com o esperado, tendo em vista que quanto menor o p , mais as partículas tenderão a ir para esquerda o que claramente nos gráficos.

Analisando as saídas no terminal:

```
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-b-5255417> ./tarefa-b2-5255417.exe
Qual é o valor de desejado, onde  $p = 1/x$ ?
3
As médias de potência 1 e 2 são: -333.333405      112080.383
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-b-5255417> ./tarefa-b2-5255417.exe
Qual é o valor de desejado, onde  $p = 1/x$ ?
4
As médias de potência 1 e 2 são: -499.986389      251089.922
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-b-5255417> ./tarefa-b2-5255417.exe
Qual é o valor de desejado, onde  $p = 1/x$ ?
5
As médias de potência 1 e 2 são: -600.008972      360676.625
```

Percebe-se que quanto menor o p , menor o raio médio - ou seja - mais a esquerda, em média, as partículas ficam - tal qual esperado. Nota-se também que a distância média quadrada também cresce com a diminuição do p , isso se deve pois quando há uma preferência para qualquer lado as partículas terão maior probabilidade de ficar longe da origem.

3 Tarefa C

Tarefa: Escrever um código em FORTRAN77 que utilizando a função rand() intrínseca da linguagem, determine a distribuição de N andarilhos aleatórios - após N passos -, onde cada andarilho tem uma probabilidade de 1/4 de ir para cima, para baixo, para direita ou para esquerda. O código deve retornar $\langle \vec{r} \rangle$ e

$$\Delta^2 = \langle \vec{r}^2 \rangle - \langle \vec{r} \rangle \cdot \langle \vec{r} \rangle$$

dos andarilhos, além de um gráfico da suas posições finais para $N = 10, 100, 1000, 10^4, 10^5$ e 10^6 .

Código escrito:

```
tarefa-c-5255417: vim-nox11 — Konsole
Nova aba  Dividir a exibição  Copiar  Colar  Localizar  ☰

program andarilho bebado 2d
c  cria os vetores que armazenarão a posição e passos dos andarilhos
c  e a variável que define o nome do arquivo de saída
dimension ihisto(-10000:10000,-10000:10000), ipx(0:3)
character*30 name
c  define 'n' como inteiro*8 para poder receber valores maiores de passos
integer*8 n

c  limpa o vetor que armazena a posição dos andarilhos
ihisto(-1000:1000,-1000:1000) = 0

c  define o número de andarilhos
m = 1000

c  define as probabilidades
ap = 0.250e0

c  define o número de passos
write(*,*) "Qual é o número de passos desejado?"
read(*,*) n

c  define o nome do arquivo de saída
name = 'histograma--5255417'

c  define as somas
somax = 0
somay = 0
somax2 = 0
somay2 = 0

c  aloca a memória para salvar os dados do histograma
open(unit=1,file=name)

c  define o loop dos andarilhos
do i=1,m

c  define a posição do andarilho
ix = 0
iy = 0

c  define os passos possíveis
ipx(0) = 0
ipx(1) = 0
ipx(2) = 0
ipx(3) = 0

c  define o loop de cada andarilho
do j=1,n

c  faz a divisão inteira de rand por ap
int = rand(0)/ap

c  incrementa um passo na direção correspondente a "int"
ipx(int) = ipx(int) + 1

end do

c  define as coordenadas finais de acordo com quantos passos
foram dados em cada direção
ix = ipx(0) - ipx(1)
iy = ipx(2) - ipx(3)

ihisto(ix,iy) = ihisto(ix,iy) + 1
write(1,*) ix,iy

s = ix**2 + iy**2
somax = somax + ix
somay = somay + iy
somax2 = somax2 + ix**2
somay2 = somay2 + iy**2

end do

c  fecha a unidade de memória
close(1)

c  define as médias
amediax = somax/m
amediaiy = somay/m
amedia2 = (somax2 + somay2 - (somax**2 + somay**2))/m

write(*,*) "<=> = ",amediax,"i + ",amediaiy,"j"
write(*,*) "e o delta ao quadrado é:",amedia2

end program

1,1  Tudo
```

Descrição: O código defini uma matriz indexada de -10 000 até 10 000 nas suas duas dimensões - que armazenará a quantidade de andarilhos em cada posição e um vetor com duas posições (0 e 3) - que determinará a direção do passo. Além de declaram N como integer*8 para suportar um valor grande de passos.

Em seguida, é definido o número de andarilhos ($m = 1000$), a probabilidade

de andar em cada direção ($ap = 0.25$).

Após isso, o código pede o número de passos e salva em N

Define "name- nome do arquivo de saída como 'histograma-5255417' (que posteriormente será alterado para corresponder ao número de passos) - e as somas das posições ($somax = 0$, $somay = 0$, $somar1 = 0$, $somar2 = 0$).

Aloca memória para um arquivo de nome "name".

Ademais, o programa inicia um loop de 1 até m - ou seja, uma iteração para cada andarilho. Definindo, como (0,0), a posição inicial ($ix=0, iy=0$) inicia-se um loop individual de cada andarilho indo de 1 até n (número de passos) no qual cada iteração definirá a direção do movimento da partícula.

Ao salvar a divisão inteira de $rand(0)$ por ap em int , obtêm-se 0 ($rand(0);0.25$), 1 ($0.25;rand(0);0.5$), 2 ($0.5;rand(0);0.75$), 3 ($0.75;rand(0)$) de modo que ao adicionar a $ipx(int)$ 1, teremos quantos passos serão dados em cada direção

Ao sair, do loop - fazendo $ix = ipx(0)-ipx(1)$ e $iy = ipx(2) - ipx(3)$ - obtemos a posição de cada partícula. E é adicionado 1 em $ihisto(ix,iy)$, de modo a contabilizar quantos andarilhos ficaram em cada posição (ix,iy). Além de salvar ix,iy no arquivo "name", definir s como a soma dos quadrados das coordenadas, adicionar à $somax$, ix e à $somay$, iy além de adicionar à $somar1$ a raiz de s e à $somar2$ s . Fecha-se a unidade de memória.

Por fim, o código calcula $\langle \vec{r} \rangle = (somax/m\vec{i} + somay/m\vec{j})$ e $\Delta^2 = \langle \vec{r}^2 \rangle - \langle \vec{r} \rangle \cdot \langle \vec{r} \rangle$ ($somax2 + somay2$ pelo produto escalar do par ($somax,somay$) - por ele mesmo)/ m) imprimindo-os na tela.

Saídas:

($n= 10$)

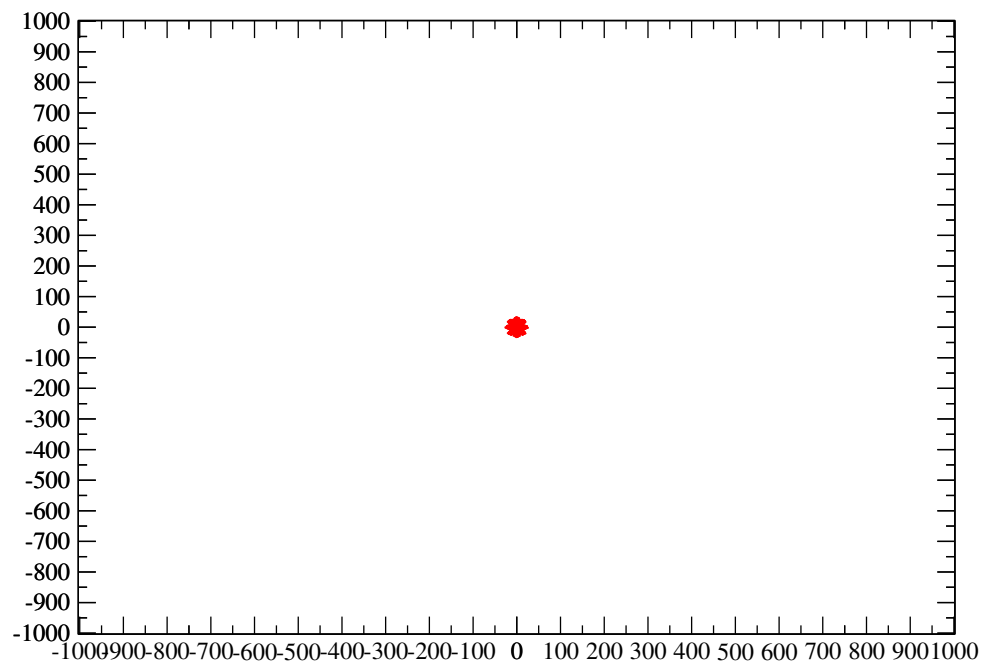


Gráfico: Posição final de cada partícula (x,y)

(n= 100)

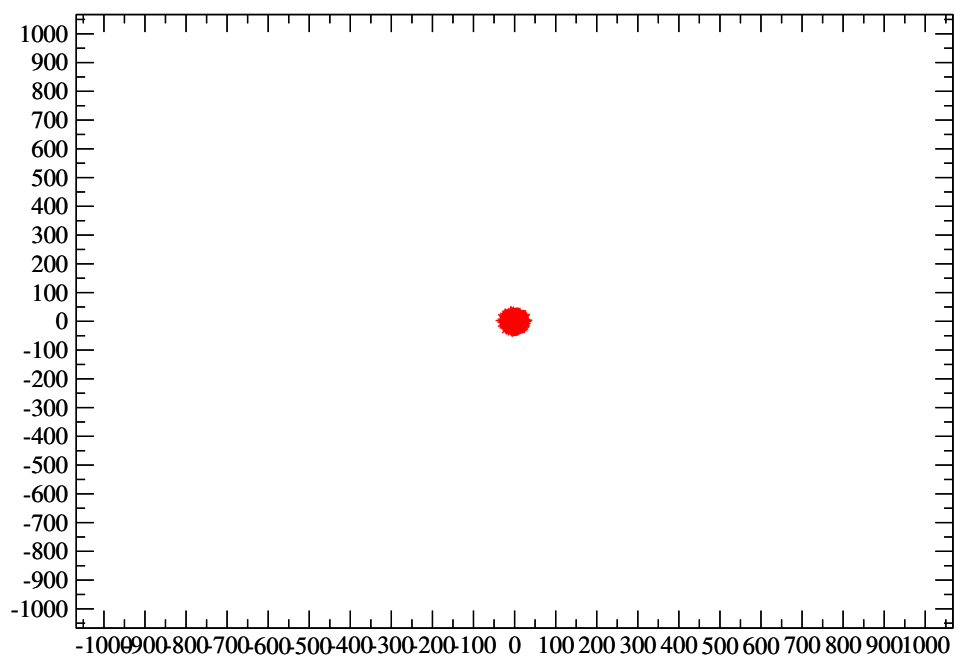


Gráfico: Posição final de cada partícula (x,y)

(n= 1000)

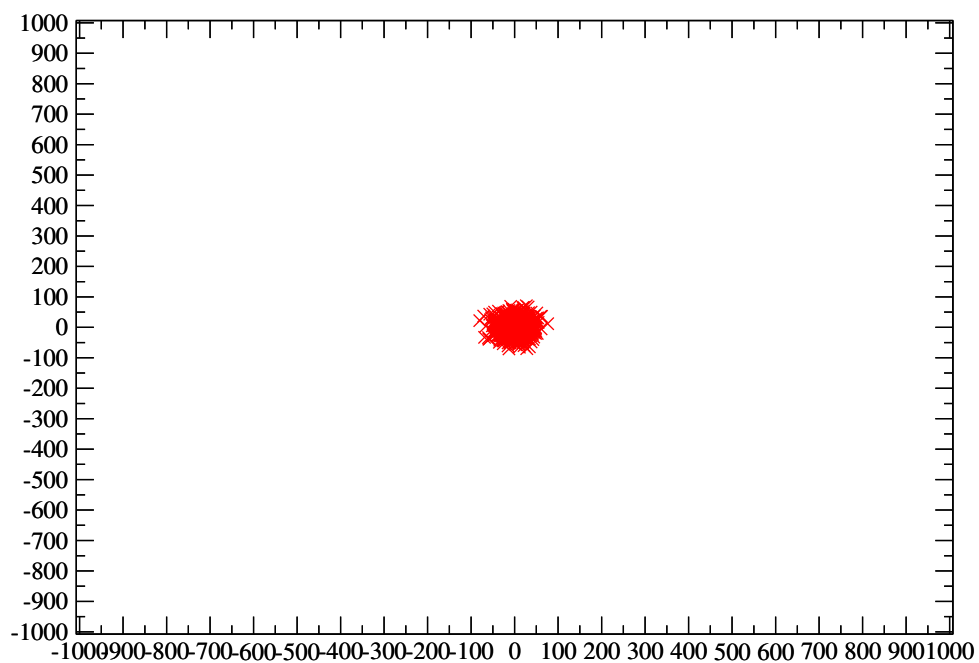


Gráfico: Posição final de cada partícula (x,y)

(n= 10 000)

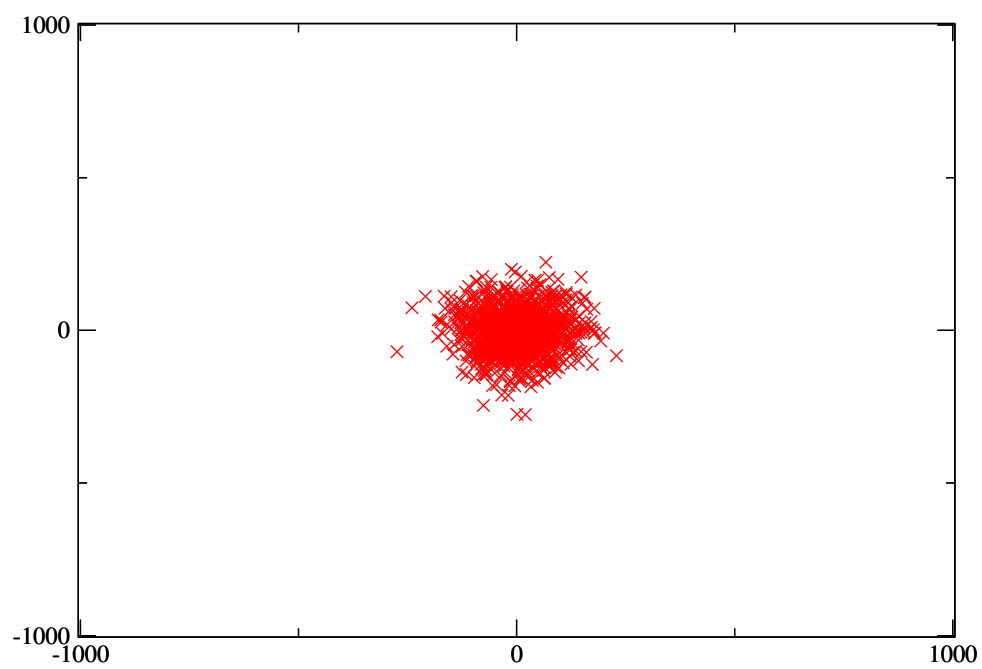


Gráfico: Posição final de cada partícula (x,y)

(n= 100 000)

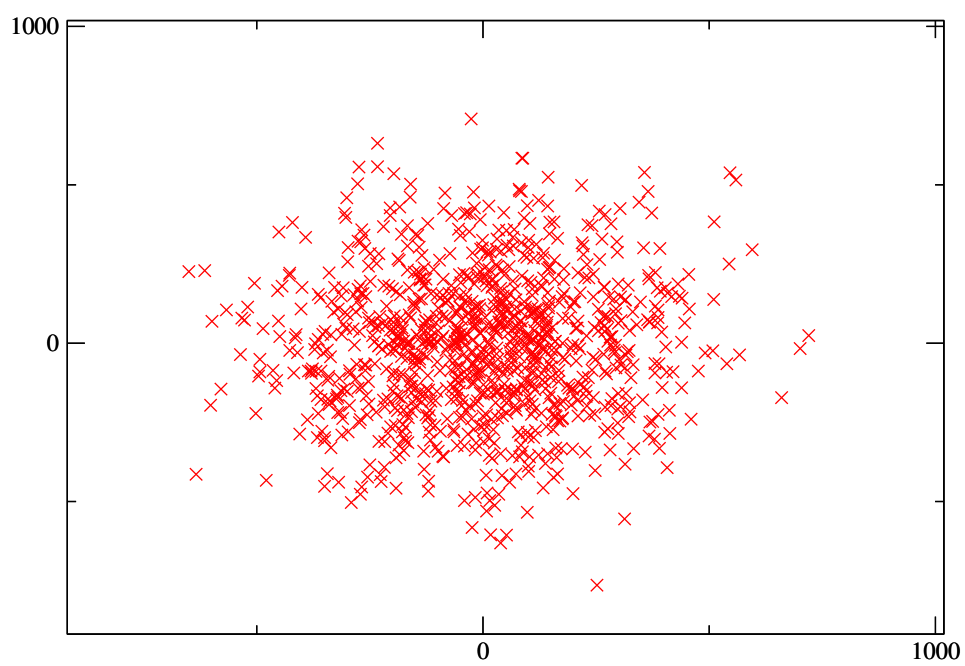


Gráfico: Posição final de cada partícula (x,y)

(n= 1 000 000)

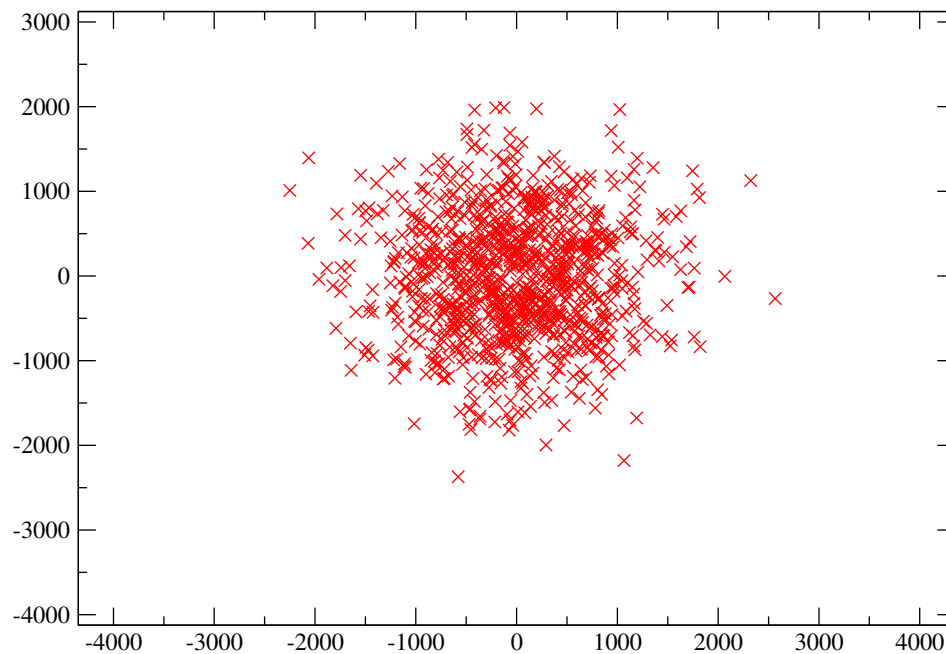


Gráfico: Posição final de cada partícula (x,y)

Tal qual esperado, as partículas se distribuem aleatoriamente, indo mais longe quanto maior o número de passos (note que a escala inicialmente vai de -1000 a 1000 e mas na última figura vai de -3000 a 3000) e ficando mais "desorganizadas" quanto mais passos foram dados.

As saídas no terminal foram:

```

pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-c-5255417> ./tarefa-c-5255417.exe
Qual é o número de passos desejado?
10
<r> = 5.90000004E-02 i + -2.50000004E-02 j
e o delta ao quadrado é: 5.62400007
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-c-5255417> ./tarefa-c-5255417.exe
Qual é o número de passos desejado?
100
<r> = 0.178000003 i + 0.200000003 j
e o delta ao quadrado é: 22.8999996
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-c-5255417> ./tarefa-c-5255417.exe
Qual é o número de passos desejado?
1000
<r> = 0.221000001 i + 0.792999983 j
e o delta ao quadrado é: 361.648010
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-c-5255417> ./tarefa-c-5255417.exe
Qual é o número de passos desejado?
10000
<r> = 0.231999993 i + -1.05999994 j
e o delta ao quadrado é: 8988.49219
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-c-5255417> ./tarefa-c-5255417.exe
Qual é o número de passos desejado?
100000
<r> = -3.61700010 i + -4.18100023 j
e o delta ao quadrado é: 67833.0000
pedro@Pedro-Lenovo:~/Documentos/GitHub/intro-fiscomp/projeto-2-5255417/tarefa-c-5255417> ./tarefa-c-5255417.exe
Qual é o número de passos desejado?
1000000
<r> = -21.5939999 i + -14.1940002 j
e o delta ao quadrado é: 347707.719

```

Note que, tanto o raio médio quanto o delta aumentam, em módulo, com o número de passos. Isso ocorre pois quanto mais passos mais as partículas de distanciam do centro (maior raio médio) e quanto mais passos, maior será o raio quadrado médio que cresce mais rápido que o produto escalar dos raio médios, ou seja, maior o delta.

4 Tarefa D

Tarefa: Escrever um código em FORTRAN77 que calcule a entropia do sistema de partículas - do item C - em função do número de passos.

Código escrito:

```
tarefa-d-5255417: vim-nox11 — Konsole
Nova aba  Dividir a exibição  Copiar  Colar  Localizar  ☰

program entropia dos andarilhos bebados
c  cria os vetores que armazenarão a posição dos andarilhos
dimension thisto(-1000:1000,-1000:1000), tpx(0:3)

c  define o número de andarilhos
n = 1000

c  define as probabilidades
ap = 0.250e0

c  define o número de passos
nmax = 1000

c  aloca memória para salvar os dados da entropia
open(unit=1, file='gráfico-5255417')

c  define as posições máximas e mínimas do sistema
ixmax = 0
ixmin = 0
iymax = 0
iymin = 0

c  calcula a entropia a cada passo
ipassos = 1
do while(ipassos.le.nmax)

c  escreve 0 em todas as coordenadas do plano
thisto(-1000:1000,-1000:1000) = 0

c  define o loop dos andarilhos
do i=1,n

c  define a posição do andarilho
ix = 0
iy = 0

c  define os passos possíveis
ipx(0) = 0
ipx(1) = 0
ipx(2) = 0
ipx(3) = 0

c  define o loop de cada andarilho
do j=0,ipassos

c  faz a divisão inteira de i por ap
int = rand(0)/ap

c  incrementa um passo na direção correspondente a "int"
tpx(int) = tpx(int) + 1

end do

c  define as coordenadas finais de acordo com quantos passos
foram dados em cada direção
ix = ipx(0) - ipx(1)
iy = ipx(2) - ipx(3)

c  atualiza as posições máximas e mínimas
if(ix.gt.ixmax) then
ixmax = ix
end if
if(ix.lt.ixmin) then
ixmin = ix
end if
if(iy.gt.iymax) then
iymax = iy
end if
if(iy.lt.iymin) then
iymin = iy
end if

ihisto(ix,iy) = ihisto(ix,iy) + 1

end do

c  escreve em um arquivo a relação entropia X passos
an = n
write(1,*) ipassos, entropia(ihisto,an,ixmax,ixmin,iymax,iymin)

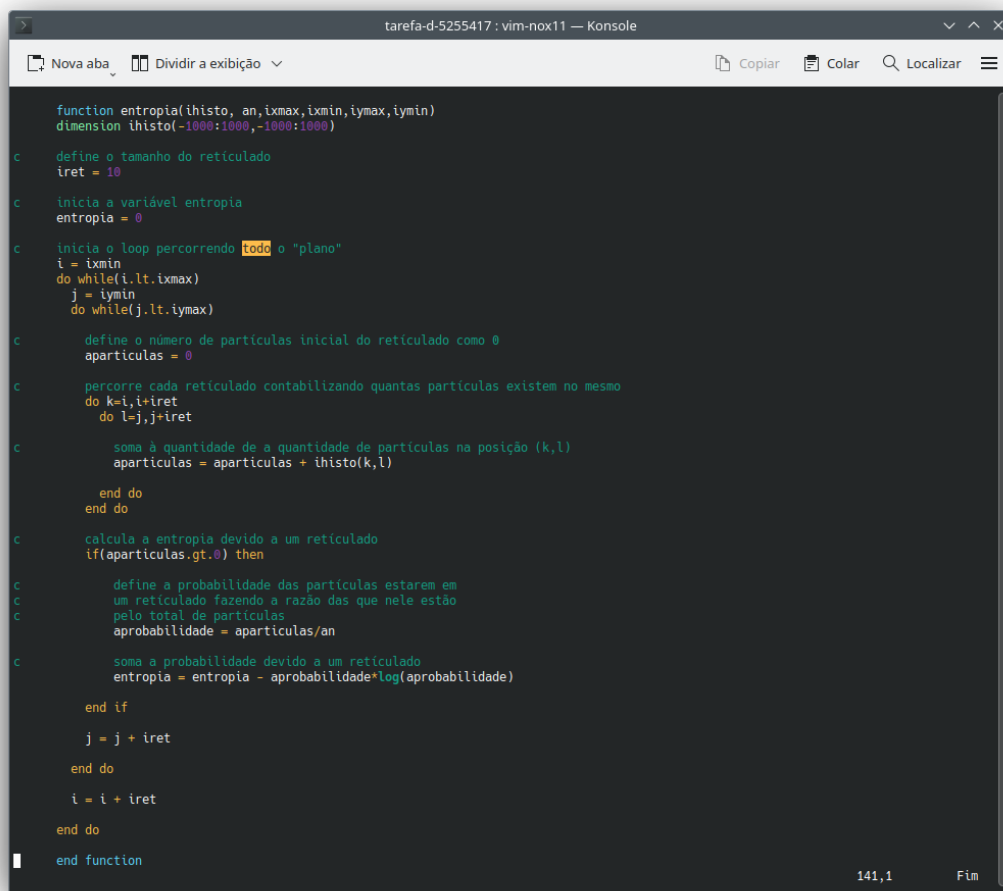
c  incrementa a quantidade de passos
ipassos = ipassos + 1

end do

c  fecha a unidade de memória
close(1)

end program

1,1  Topo
```

```
function entropia(ihisto, an, ixmax, ixmin, iymax, iymmin)
dimension ihisto(-1000:1000,-1000:1000)

c define o tamanho do reticulado
iret = 10

c inicia a variável entropia
entropia = 0

c inicia o loop percorrendo todo o "plano"
i = ixmin
do while(i.lt.ixmax)
j = iymmin
do while(j.lt.iymax)

c define o número de partículas inicial do reticulado como 0
apartículas = 0

c percorre cada reticulado contabilizando quantas partículas existem no mesmo
do k=i,i+iret
do l=j,j+iret

c soma à quantidade de a quantidade de partículas na posição (k,l)
apartículas = apartículas + ihisto(k,l)

end do
end do

c calcula a entropia devido a um reticulado
if(apartículas.gt.0) then

c define a probabilidade das partículas estarem em
c um reticulado fazendo a razão das que nele estão
c pelo total de partículas
aprobabilidade = apartículas/an

c soma a probabilidade devido a um reticulado
entropia = entropia - aprobabilidade*log(aprobabilidade)

end if

j = j + iret
end do
i = i + iret
end do

end function
```

Descrição:

Efetua uma simulação da movimentação de partículas em 2D - tal qual o item C - mas calculando a entropia a cada passo e salvando a tupla (passos, entropia) em um arquivo de nome 'gráfico-5255417'.

Observe que a simulação é calculada de 1 passo até ipassos, após isso calcula-se a entropia, incrementa-se ipassos e recalcula-se a simulação apartir do passo 1. Esse processo é repetido até que ipassos seja igual o número máximo de passos (nmax = 1000). O programa também calcula as posições máximas e mínimas de x e y a cada interação.

A função utilizada para calcular a entropia recebe uma lista com quantas partículas estão em cada posição, um número real do total de partículas e as posições x e y mínimas e máximas.

Defini o tamanho do reticulado (iret = 10), entropia como 0 e inicia dois loops - encadeados - das posições mínimas até as posições máximas, que definem o valor de i e j que, por sua vez são incrementados por 10 a cada interação.

No começo de cada conjunto de iterações define a quantidade de partículas (apartículas = 0) e então inicia-se dois loops -encadeados - que vão de i até i+10 e de j até j+10. De modo que i e j indexarão a lista com as posições das partículas que retorna o número de partículas na posição (i,j) que é incrementada em apartículas.

Saindo dos loops menores se apartículas for maior que 0 (para evitar o problema do ln indefinido em 0), calcula-se a probabilidade de um partícula estar em um reticulado dividindo apartículas (número de partículas no reticulado) pelo número total de partículas.

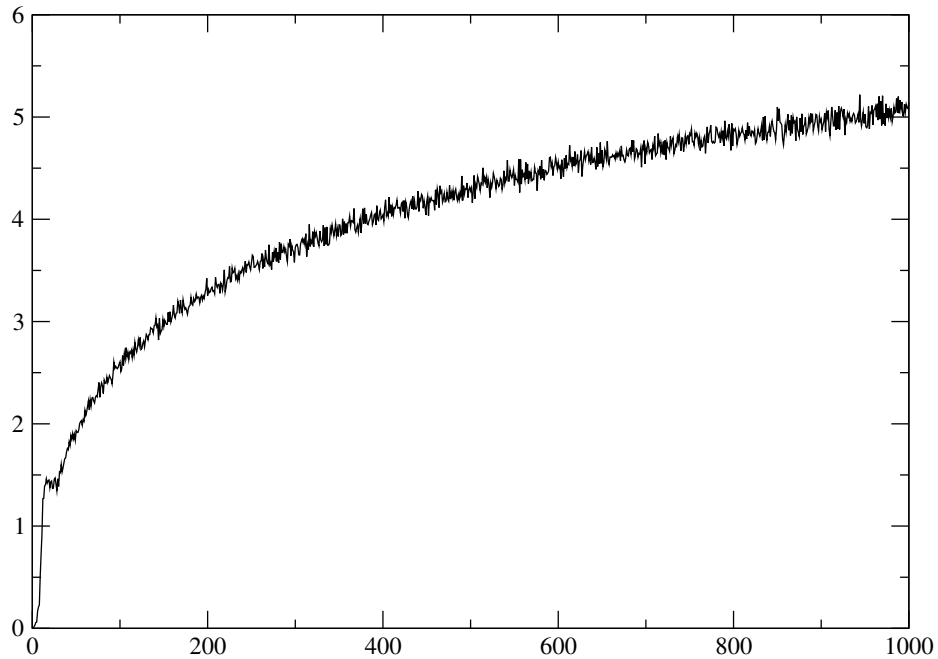
Por fim, utilizando a fórmula (onde P_i é a probabilidade de uma partícula estar em um reticulado)

$$S = \sum -P_i \ln(P_i) \quad (2)$$

incrementa a entropia pela entropia devido a um reticulado.

Ao fim de todos os loops obtêm-se a entropia para um dado número de passos, relação essa que é salva no arquivo de saída.

Saída:



Percebe-se que a entropia cresce muito similar a uma função logaritmica, ou seja, cresce - cada vez mais devagar - em função do aumento do número de passos.

Tal comportamento é justificado tendo em vista que a fórmula para o cálculo da entropia depende do $\ln()$ da probabilidade de uma partícula estar em um determinado reticulado como esta probabilidade é linear com relação ao número de passos, faz sentido que a entropia seja logarítmica.