UNIVERSIDADE DE SÃO PAULO
Instituto de Física de São Carlos

Projeto 4:Modelo de Crescimento

Anderson Araujo de Oliveira 11371311

2020

Conteúdo

1 Tarefa-1

1.1 Teoria

Nesse projeto iremos ver como é o desenvolvimento de modelos de crescimento, usamos as regras de evolução de automato que envolve as três vizinhanças, sendo $[A_{i-1}, A_i, A_{i+1}]$ (o anterior, o atual e o próximo), onde esses autômatos só podem ter 0 ou 1 do binário, assim, conseguimos 2^3 ou 8 possibilidades, a informação que os vizinhos tem decidi qual vai ser o estado do automato. A tabela abaixo mostra as configurações possíveis e o seu valor em decimal.

A_{i-1}	A_i	A_{i+1}	D
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Tabela 1: Caption

Podemos ver que temos oito tipos de condições que os vizinhos podem ter, para ver qual vai ser o estado do autômato. Assim, temos 2^8 ou 256 condições ou regras, podemos transformar esse numero binário de forma que é possível saber quais dessas condições vão ser colocadas, para cada valor inserido no código, assim o modelo irá crescer de modo diferente.A tabela 3 mostra como séria isso para algumas regras , vamos usar a régua binária para ajudar na tabela 2.

000	001	010	011	100	101	110	111
1	2	4	8	16	32	64	128

Tabela 2: Régua de binário

Vamos encontrar as respectivas regra e sua condições.

000	001	010	011	100	101	110	111	
0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	0	0	21
1	0	0	0	0	0	1	0	65

Tabela 3: Caption

1.2 Resultados

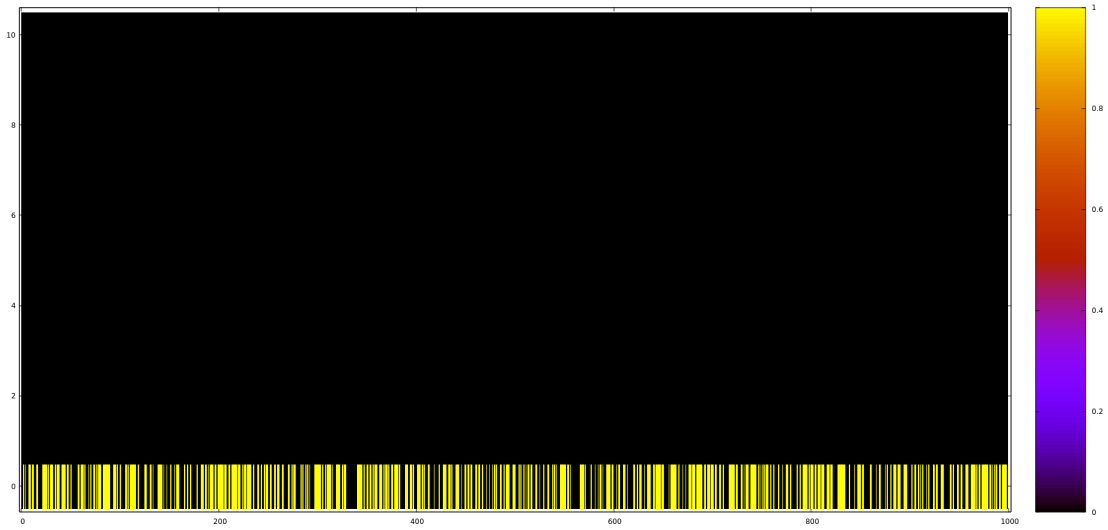

Figura 1: Regra 0

A regra 0, devia se esperar que todos os valores tendem a informar 0.

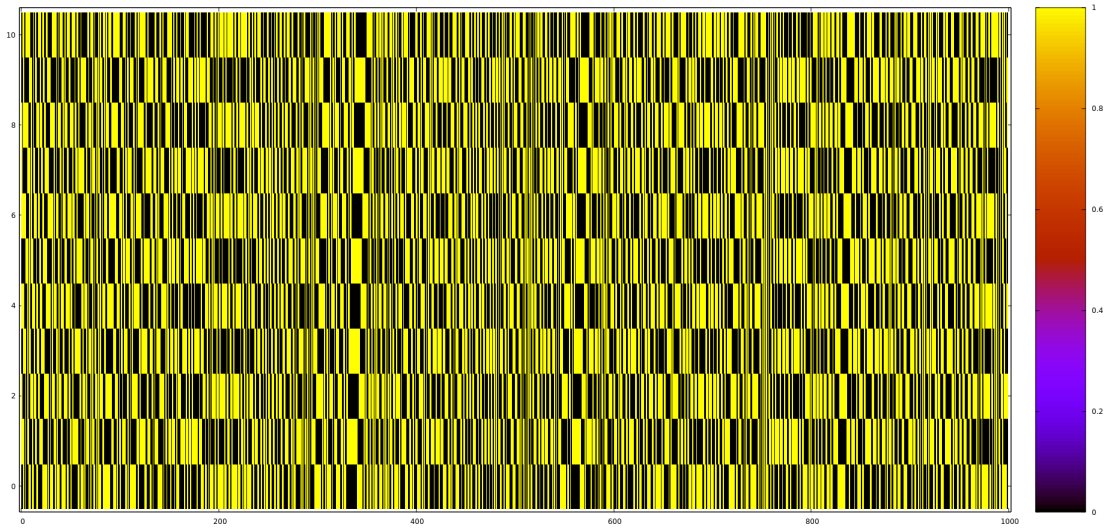

Figura 2: Regra 12

Regra 12, Vemos que existe um certo padrão para cada interação que fica repetindo.

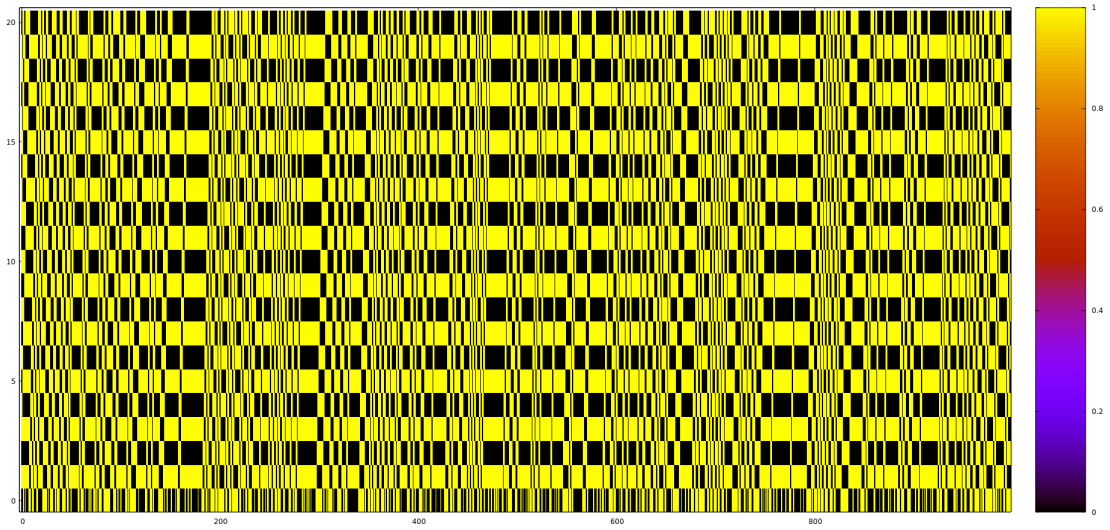

Figura 3: Regra 51

Regra 51, Vemos que existe um padrão diferente, porém, esse padrão fica repetindo igual a regra 12.

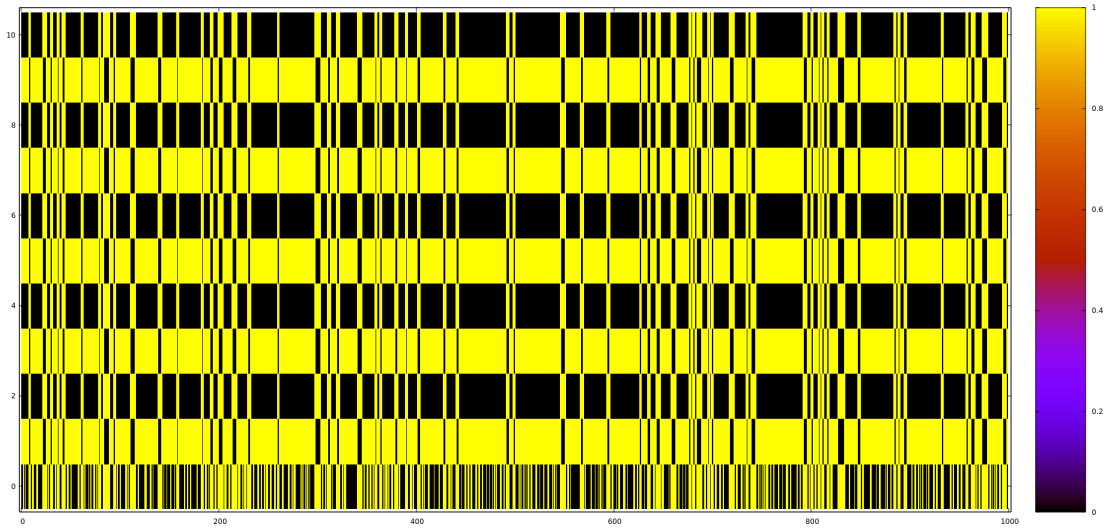

Figura 4: Regra 86

Regra 86, Vemos que existe um padrão diferente das regras anteriores, porém, esse padrão fica repetindo igual a regra 12 e a 51.

Figura 5: Regra 232

Nessa regra, após, a segunda interação todos os sítios ficam ocupado por um 1.

Figura 6: Regra 254

Essa regra segue o mesmo padrão que a regra 234.

1.3 Código

```
1 program analise
2   implicit integer*16(a-h,o-z)
3   integer*16::soma,dois
4   !separando em dois vetores um para manter as condi  es at  processar
     todos os automato
5   !outro para salvar as altera  es
```

```
6   integer, dimension(10000) ::celulas1,celulas2,r
7   write(*,*)"digite a quantidade de intera oes"
8   read(*,*)int
9   write(*,*)'Digite a quantidade de automatos'
10  read(*,*)l
11  write(*,*)'Digite para qual modelo iremos analisar(1 randomico, 2 tudo
     zero e 3 tudo um)'
12  read(*,*)aut
13  write(*,*)'Qual regra ir  usar'
14  read(*,*)regra
15  b=7
16  dois=2
17  soma=0
18  do while(b>=0)!extraindo a regra
19    if (regra>=2**b)then
20      r(b+1)=1
21      b=b-1
22    else if(regra<2**b)then
23       r(b+1)=0
24       b=b-1
25    endif
26  enddo
27
28
29  !gerando o valor dos automatos iniciais
30  if(aut==1)then
31    do i=1,l!automatos aleatorios
32      if (rand()>=0.5)then
33        celulas1(i)=1
34      else
35        celulas1(i)=0
36      endif
37    enddo
38  else if(aut==2)then
39    do i=1,l!todos 0
40      celulas1(i)=0
41    enddo
42  else if(aut==3)then
43    do i=1,l!todos 1
44      celulas1(i)=1
45    enddo
46  endif
47
48
49  !passando os dados para um arquivo
50  write(2,*)(celulas1(a),a=1,l)
51
52  !regra da epidemia
53  do j=1,int!do que ir  contar o tempo
54
55    do i=1,l!do que ir  fazer a logica para regra da epidemia
56
57      if (0==(i-1))then!condi  es para caso o chega na borda
```

```
58          if (r(8)==1 .and. celulas1(l)==1 .and. celulas1(i)==1 .and.
    celulas1(i+1)==1)then
59            celulas2(i)=1
60          else if(r(7)==1 .and. celulas1(l)==1 .and. celulas1(i)==1 .and.
    celulas1(i+1)==0)  then
61            celulas2(i)=1
62          else if(r(6)==1 .and. celulas1(l)==1 .and. celulas1(i)==0 .and.
    celulas1(i+1)==1)  then
63            celulas2(i)=1
64          else if(r(5)==1 .and. celulas1(l)==1 .and. celulas1(i)==0 .and.
    celulas1(i+1)==0)  then
65            celulas2(i)=1
66          else if(r(4)==1 .and. celulas1(l)==0 .and. celulas1(i)==1 .and.
    celulas1(i+1)==1)  then
67            celulas2(i)=1
68          else if(r(3)==1 .and. celulas1(l)==0 .and. celulas1(i)==1 .and.
    celulas1(i+1)==0)  then
69            celulas2(i)=1
70          else if(r(2)==1 .and. celulas1(l)==0 .and. celulas1(i)==0 .and.
    celulas1(i+1)==1)  then
71            celulas2(i)=1
72          else if(r(1)==1 .and. celulas1(l)==0 .and. celulas1(i)==0 .and.
    celulas1(i+1)==0)  then
73            celulas2(i)=1
74          else
75            celulas2(i)=0
76          endif
77
78        else if ((l+1)==(i+1))then!condi  es para caso o chega na borda
79          if (r(8)==1 .and. celulas1(i-1)==1 .and. celulas1(i)==1 .and.
    celulas1(1)==1)then
80            celulas2(i)=1
81          else if(r(7)==1 .and. celulas1(i-1)==1 .and. celulas1(i)==1 .and.
    celulas1(1)==0)  then
82            celulas2(i)=1
83          else if(r(6)==1 .and. celulas1(i-1)==1 .and. celulas1(i)==0 .and.
    celulas1(1)==1)  then
84            celulas2(i)=1
85          else if(r(5)==1 .and. celulas1(i-1)==1 .and. celulas1(i)==0 .and.
    celulas1(1)==0)  then
86            celulas2(i)=1
87          else if(r(4)==1 .and. celulas1(i-1)==0 .and. celulas1(i)==1 .and.
    celulas1(1)==1)  then
88            celulas2(i)=1
89          else if(r(3)==1 .and. celulas1(i-1)==0 .and. celulas1(i)==1 .and.
    celulas1(1)==0)  then
90            celulas2(i)=1
91          else if(r(2)==1 .and. celulas1(i-1)==0 .and. celulas1(i)==0 .and.
    celulas1(1)==1)  then
92            celulas2(i)=1
93          else if(r(1)==1 .and. celulas1(i-1)==0 .and. celulas1(i)==0 .and.
    celulas1(1)==0)  then
94            celulas2(i)=1
95          else
```

```
96            celulas2(i)=0
97          endif
98
99        else
100         if (r(8)==1 .and. celulas1(i-1)==1 .and. celulas1(i)==1 .and.
    celulas1(i+1)==1) then
101           celulas2(i)=1
102         else if(r(7)==1 .and. celulas1(i-1)==1 .and. celulas1(i)==1 .and.
    celulas1(i+1)==0)  then
103           celulas2(i)=1
104         else if(r(6)==1 .and. celulas1(i-1)==1 .and. celulas1(i)==0 .and.
    celulas1(i+1)==1)  then
105           celulas2(i)=1
106         else if(r(5)==1 .and. celulas1(i-1)==1 .and. celulas1(i)==0 .and.
    celulas1(i+1)==0)  then
107           celulas2(i)=1
108         else if(r(4)==1 .and. celulas1(i-1)==0 .and. celulas1(i)==1 .and.
    celulas1(i+1)==1)  then
109           celulas2(i)=1
110         else if(r(3)==1 .and. celulas1(i-1)==0 .and. celulas1(i)==1 .and.
    celulas1(i+1)==0)  then
111           celulas2(i)=1
112         else if(r(2)==1 .and. celulas1(i-1)==0 .and. celulas1(i)==0 .and.
    celulas1(i+1)==1)  then
113           celulas2(i)=1
114         else if(r(1)==1 .and. celulas1(i-1)==0 .and. celulas1(i)==0 .and.
    celulas1(i+1)==0)  then
115           celulas2(i)=1
116         else
117           celulas2(i)=0
118         endif
119       endif
120     enddo
121
122       do i=1,l!passando todos os valores alterados para o vetor do tempo
    atual
123         celulas1(i)=celulas2(i)
124       enddo
125
126       !passando os dados para um arquivo
127       write(2,*)(celulas1(a),a=1,l)
128   enddo
129 end program
```

2 Tarefa-2

2.1 Teoria

A maioria dos fenômenos de crescimento, envolvem um certo grau de aleatoriedade, sendo crescimento de cristais , tumores flocos de neve, partidos políticos, Urbanização e outros. Esse tipo de modelo que iremos estudar o crescimento tende aumenta através de sua superfície,

um desse modelo é de Eden que simula o crescimento do tumores.

Figura 7: Modelo de Eden

Outro tipo de modelo que iremos estudar aqui é o de crescimento de blocos de neve , fuligem, descargas elétricas, colônia de corais e incêndio em flores, o modelo é chamado de DLA('diffusion limited agregation' ou 'agregação limitada por difusão'). Fractal é um objeto geométrico que pode ser dividido em partes como se fosse composto por um número grande de pequenos blocos, uma de suas propriedades é auto-semelhança e a complexidade infinita.

A auto-semelhança é a simetria através das escalas. Consiste em cada pequena porção do fractal poder ser vista como uma réplica de todo o fractal numa escala menor.

A complexidade infinita prende-se com o facto de o processo gerador dos fractais ser recursivo, tendo um número infinito de iterações.

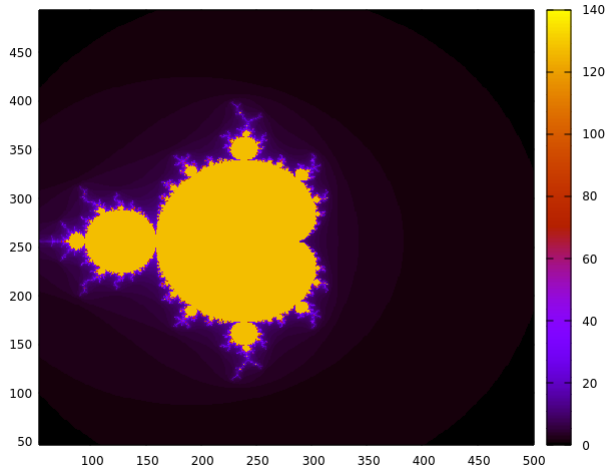

Figura 8: Fractal de Mandelbrot

Neste modelo teremos uma particular solta a uma certa distância da partícula primordial(ou a primeira partícula), a partícula solta vai se movimentar de forma browniana, quando esta partícula acerta agregado é adicionada a ele, a figura 9 pode se comparar a um fractal como vemos.

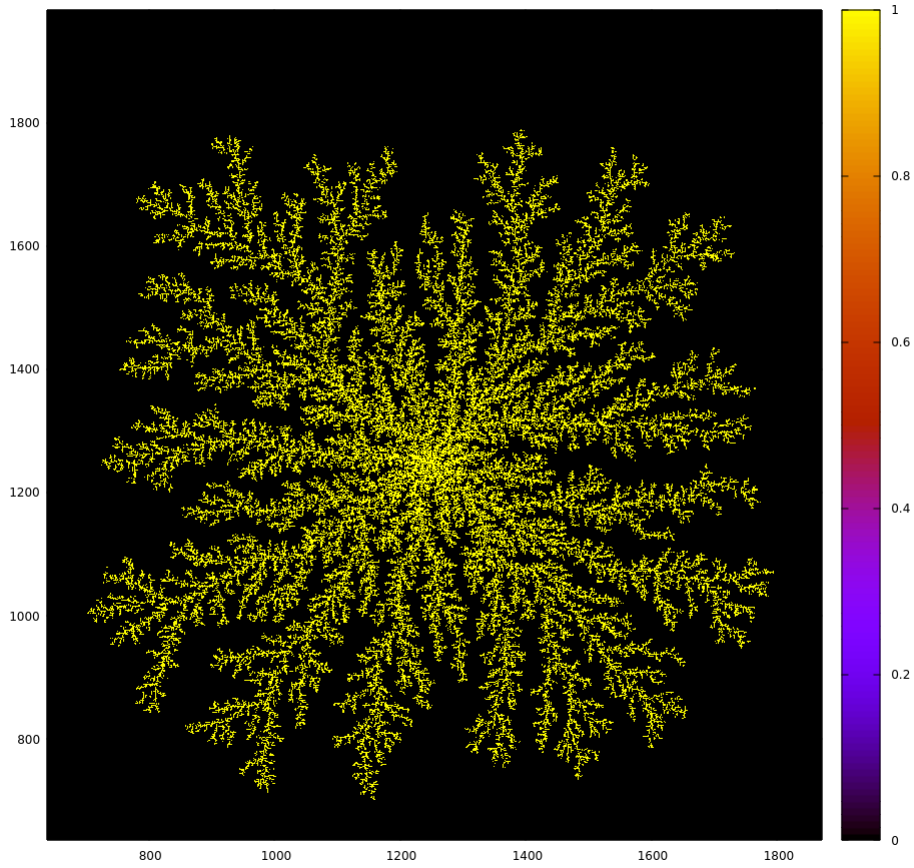

Figura 9: Modelo do DLA com 200000 pontos

Nesse espaço euclideano, onde foi construído nosso fractal, pode ser tirar uma informação que sua dimensão do espaço. A dimensão do fractal representa o grau de ocupação no espaço, que tem a ver com seu grau de irregularidade, podemos obter essa variável, descobrindo o valor do expoente que representa numero de partículas em função do raio.

2.2 Resultado

Agora iremos verificar os resultados que obtivemos, vamos utilizar a teoria que explicamos anteriormente para ver se os resultados são coerentes.

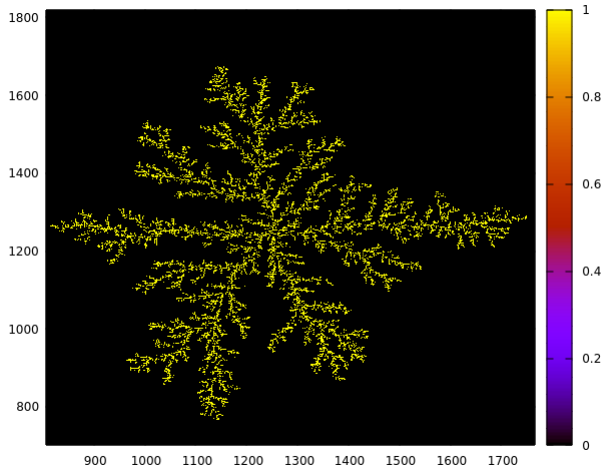

Figura 10: Mostra um fractal com 50000 pontos

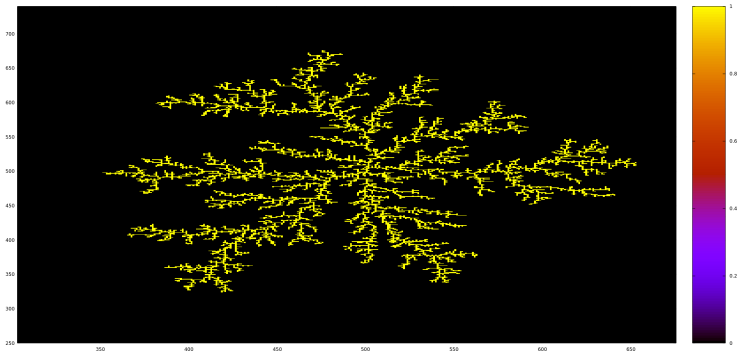

Figura 11: Mostra um fractal com 10000 pontos

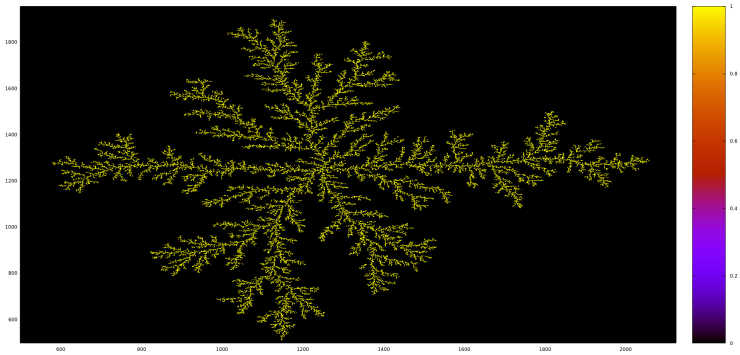

Figura 12: Mostra um fractal com 100000 pontos

O fractais que obtemos nas figuras acima, as suas dimensões fractal e seu erro são informado abaixo.

```
Final set of parameters           Asymptotic Standard Error
=======================           ==========================
a                = 1.91179        +/- 0.007087     (0.3707%)
d                = 1.63849        +/- 0.0006138    (0.03746%)
```

Figura 13: A dimensão fractal referente do fractal na figura 10

```
a                = 2.61359        +/- 0.01633      (0.6249%)
d                = 1.55056        +/- 0.001218     (0.07853%)
```

Figura 14: A dimensão fractal referente do fractal na figura 11

```
Final set of parameters           Asymptotic Standard Error
=======================           ==========================
a                = 4.94577        +/- 0.006636     (0.1342%)
d                = 1.47844        +/- 0.0002057    (0.01391%)
```

Figura 15: A dimensão fractal referente do fractal na figura 12

2.3 Código

No código foi utilizado uma matriz de 10000 por 10000, mas o usuário pode diminuir o tamanho, ao inserir os dados, a origem do sistema fica no $(\frac{l}{2}, \frac{l}{2})$ onde l é tamanho da malha ficando no meio dele, fizemos uma aproximação para centro de massa, onde será na origem do sistema,o movimento das partículas no códigom, foi baseado em um projeto anterior de fiscomp 1.

```
1  program t2
2
3    integer*8::a,b,be,n,x,y,irr
4    integer,dimension(5)::ipx,ipy
5    integer, dimension(10000,10000)::frac
6    write(*,*)"digite a quantidade de particulas"
7    read(*,*)be
8    write(*,*)"digite o tamanho do mapa"
9    read(*,*)l
10   !zerar as variav is para caso o c digo  j  foi rodado outro vez
11   r0=0
12   b=1
13   !vetores de movimento
14   ipx(1)=1
15   ipx(2)=-1
16   ipx(3)=0
17   ipx(4)=0
18
19   ipy(1)=0
20   ipy(2)=0
21   ipy(3)=1
22   ipy(4)=-1
23   pi=acos(-1d0)
24   do j=1,l!zerando a matriz
```

```
25      do i=1,l
26        frac(i,j)=0
27      enddo
28    enddo
29    !colocando particula na origem (0,0)     (l/2,l/2)
30    frac(l/2,l/2)=1
31    !do  que ira realizar a movimenta  o do numero de bebados escolhidos
       pelo usuario
32    do while(b<=be)!particulas agregados
33
34      !gerando a coordenada aleat ria
35      r=5+r0
36      teta=rand()
37      x=r*cos(2*pi*teta)+l/2
38      y=r*sin(2*pi*teta)+l/2
39
40      !do para fazer contagem de passos do bebados nos 4 sentidos
41      do while(9>3)
42
43        !condi  es para ver qual caminho bebados v  o fazer
44        irr=4*rand()+1
45        x=x+ipx(irr)
46        y=y+ipy(irr)
47
48        if(x>0 .and. y>0)then
49          if ((frac(x+1,y)+frac(x-1,y)+ frac(x,y+1)+frac(x,y-1))>=1)then!
       procurando uma particula
50            frac(x,y)=1!quando a particula encontra o agregado
51            b=b+1
52            exit
53          endif
54        endif
55
56        if(((x-l/2)**2+(y-l/2)**2)**(0.5)>=1.5*r .or. x<0 .or. y<0)then!
       verificar se a particula foi para longe
57          x=1!para  n  o  dar erro quando for no if do raio
58          y=1
59          exit
60        endif
61      enddo
62
63      if(((x-l/2)**2+(y-l/2)**2)**(0.5)>r0 .and. frac(x,y)==1) then!
       verificar se o novo agregado e a particula mais longe
64        r0=((x-l/2)**2+(y-l/2)**2)**(0.5)
65      endif
66      write(2,*)r,b!salvando o raio e a quantidade de particulas
67    enddo
68
69    do i=1,l!salvando os dados em um arquivo
70        write(1,*)(frac(a,i),a=1,l)
71    enddo
72
73  end program
```

3 Tarefa-3

3.1 Resultados

Essa tarefa iremos procurar a dimensão fractal de um fractal 3d, com isso iremos ter seis graus de liberdade no sistema,portanto, a partícula vai ter acesso a mais lugares.No código adaptamos programa anterior que é 2d para 3d.

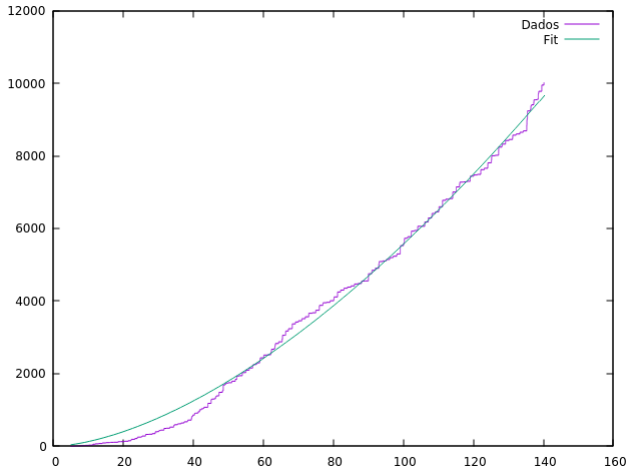

Figura 16: Função do numeros de particulas em função do raio

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
d               = 2.30125          +/- 8.661e-05    (0.003763%)
```

Figura 17: d=df,

Vemos a dimensão fractal se manteve próximo do 2d, portanto, vemos que a irregularidade do fractal ainda é mesmo dos fractais anteriores.

3.2 Código

```
1  program QC
2    integer*8::a,b,be,n,x,y,z,irr
3    integer,dimension(7)::ipx,ipy,ipz
4    integer, dimension(1000,1000,1000)::frac
5    write(*,*)"digite a quantidade de particulas"
6    read(*,*)be
7    write(*,*)"digite o tamanho do mapa"
8    read(*,*)l
9    !zerar as variav is para caso o c digo j  foi rodado outro vez
10   r0=0
11   b=1
12   !vetores para movimento aleatorio
13   ipx(1)=1
14   ipx(2)=-1
```

```
15  ipx(3)=0
16  ipx(4)=0
17  ipx(5)=0
18  ipx(6)=0
19
20  ipy(1)=0
21  ipy(2)=0
22  ipy(3)=1
23  ipy(4)=-1
24  ipy(5)=0
25  ipy(6)=0
26
27  ipz(1)=0
28  ipz(2)=0
29  ipz(3)=0
30  ipz(4)=0
31  ipz(5)=1
32  ipz(6)=-1
33
34  pi=acos(-1d0)
35
36  do j=1,l!zerando a matriz
37    do i=1,l
38      do k=1,l
39        frac(i,j,k)=0
40      enddo
41    enddo
42  enddo
43  frac(l/2,l/2,l/2)=1
44  !do  que ira realizar a movimenta  o do numero de bebados escolhidos
     pelo usuario
45  do while(b<=be)
46    !gerando a coordenada aleat ria atrav s dos angulos
47    r=5+r0
48    teta=rand()
49    phi=rand()
50    x=r*cos(2*pi*teta)*sin(2*pi*rand())+l/2
51    y=r*sin(2*pi*teta)*sin(2*pi*rand())+l/2
52    z=r*cos(2*pi*rand())+l/2
53    !do para fazer contagem de passos do bebados nos 4 sentidos
54
55    do while(9>3)
56      !condi  es para ver qual caminho bebados v o fazer
57      irr=6*rand()+1
58      x=x+ipx(irr)
59      y=y+ipy(irr)
60      z=z+ipz(irr)
61
62      if(x>0 .and. y>0 .and. z>0)then
63        if ((frac(x+1,y,z)+frac(x-1,y,z)+frac(x,y+1,z)+frac(x,y-1,z)+frac(
     x,y,z+1)+frac(x,y,z-1))>=1 )then
64          frac(x,y,z)=1!quando a particula encontra o agregado
65          b=b+1
66          exit
```

```
67          endif
68        endif
69
70
71        if(((x-l/2)**2+(y-l/2)**2+(z-l/2)**2)**(0.5)>=1.5*r .or. x<0 .or. y
      <0 .or. z<0)then!verificar se a particula foi para longe
72          x=1!para n o dar erro
73          y=1
74          z=1
75          exit
76        endif
77
78       enddo
79       if(((x-l/2)**2+(y-l/2)**2+(z-l/2)**2)**(0.5)>r0 .and. frac(x,y,z)==1)
      then!verificar se o novo agregado e a particula mais longe
80        r0=((x-l/2)**2+(y-l/2)**2+(z-l/2)**2)**(0.5)
81       endif
82       !enviando os dados para um arquivo
83       write(2,*)r,b
84     enddo
85     !do j=1,l
86       !  write(1,*)(frac(l/2,j,a),a=1,l)
87      ! enddo
88     !do i=1,l
89      ! do j=1,l
90      !   write(10+i,*)(frac(i,j,a),a=1,l)
91      ! enddo
92     !enddo
93   end program
```

4 Tarefa-4

4.1 Resultados

Nessa tarefa, será colocado uma reta na origem, colocaremos a uma certa distância uma partícula, vai começar a se mover aleatoriamente até encontrar a reta, usando as mesma logica dos códigos anteriores, para como identificar quando ta perto do agregado e ignorar partículas longe.

Para os resultado que são esperados, as imagens que devem ser formada algo parecido com o raio corona.

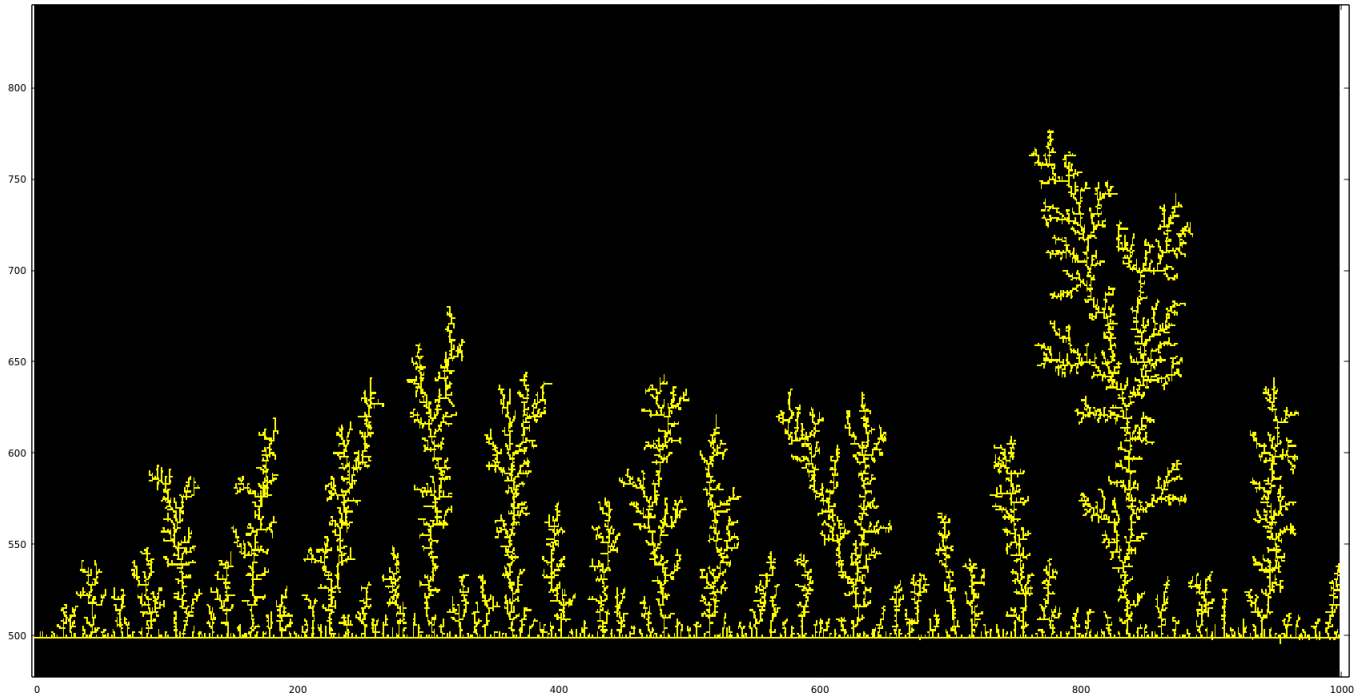

Figura 18: Resultado obtido do programa abaixo

4.2 Código

```
program t4
  integer*8::a,b,be,n,x,y,irr
  integer,dimension(5)::ipx,ipy
  integer, dimension(10000,10000)::frac

  write(*,*)"digite a quantidade de particulas"
  read(*,*)be
  write(*,*)"digite o tamanho do mapa"
  read(*,*)l
  !zerar as variav is para caso o c digo  j   foi rodado outro vez
  r0=0
  b=1

  ipx(1)=1
  ipx(2)=-1
  ipx(3)=0
  ipx(4)=0


  ipy(1)=0
  ipy(2)=0
  ipy(3)=1
  ipy(4)=-1

  pi=acos(-1d0)

```

```
27  do j=1,l!zerando a matriz
28    do i=1,l
29      if(l/2==j)then
30        frac(i,j)=1
31      else
32        frac(i,j)=0
33      endif
34    enddo
35  enddo
36  do i=1,l!salvando os dados em um arquivo
37      write(3,*)(frac(a,i),a=1,l)
38  enddo
39  !do  que ira realizar a movimenta  o do numero de bebados escolhidos
     pelo usuario
40  do while(b<=be)
41
42    !gerando a coordenada aleat ria da particula
43    r=10+r0
44    y=r+l/2!soltar a partiular na parte superior
45    x=l*rand()
46    !do para fazer contagem de passos do bebados nos 4 sentidos
47
48    do while(9>3)
49      !condi  es para ver qual caminho bebados v  o fazer
50      irr=4*rand()+1
51      x=x+ipx(irr)
52      y=y+ipy(irr)
53
54      !write(*,*)x,y
55      if(x>0 .and. y>0)then
56        if ((frac(x+1,y)+frac(x-1,y)+frac(x,y+1)+frac(x,y-1))>=1)then
57          frac(x,y)=1!quando a particula encontra o agregado
58          b=b+1
59          exit
60        endif
61      endif
62
63      if(abs(y-l/2)>1.5*r .or. x<0 .or. y<0)then!verificar se a particula
     foi para longe
64        x=1!para  n  o  dar erro
65        y=1
66        exit
67      endif
68    enddo
69
70    if(x>0 .and. y>0)then
71      if(abs(y-l/2)>r0 .and. frac(x,y)==1) then!verificar se o novo
     agregado e a particula mais longe
72        r0=abs(y-l/2)
73      endif
74    endif
75
76    write(2,*)r,b
77  enddo
```

```
78
79    do i=1,l!salvando os dados em um arquivo
80        write(1,*)(frac(a,i),a=1,l)
81    enddo
82
83  end program
```

5 Tarefa-5

5.1 Resultados

Nessa parte iremos colocar o agregado para se movimentar, espalhando as partículas aleatoriamente pelo mapa, ocupando aproximadamente um valor do p do espaço, onde agregado irá se movimentar de forma Browniana capturando as partículas em volta, esse sistema que será simulado se parece, com um parasita sugando nutrientes dentro de um corpo.

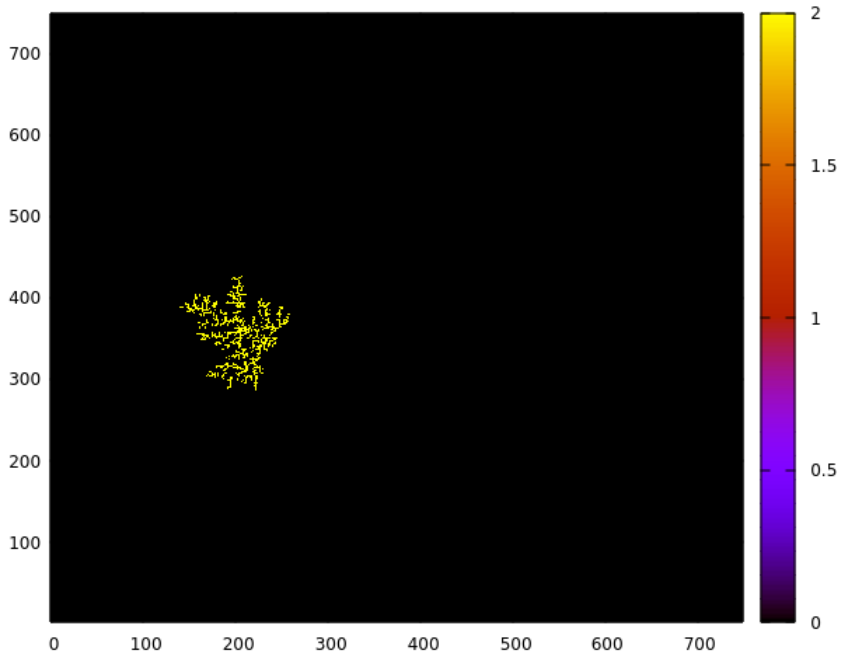

Figura 19: Onde particula parou

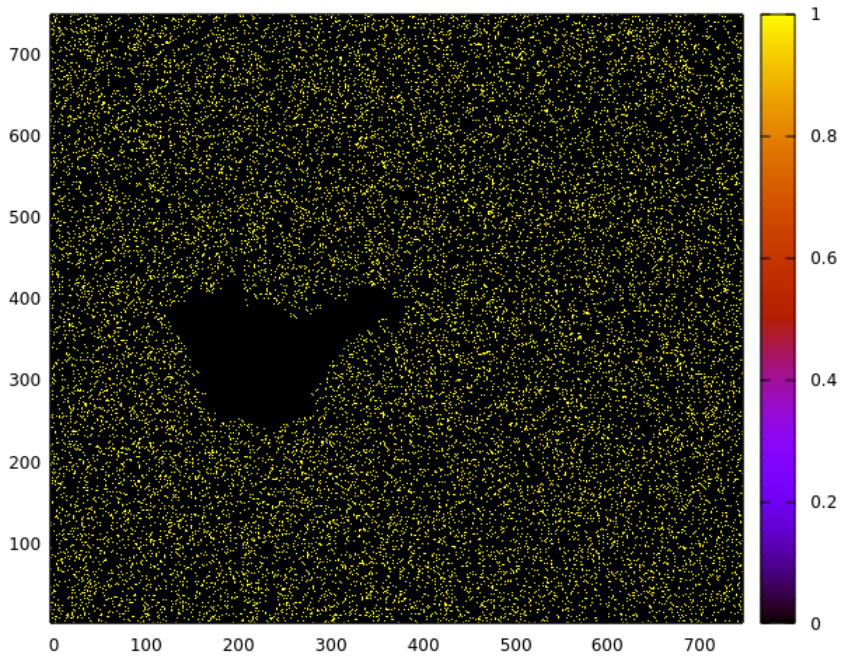

Figura 20: O gráfico mostra onde o agregado passou

Para o fractal que obtemos a dimensão fractal que foi obtido.

```
a                       = 0.687659          +/- 0.00957       (1.392%)
d                       = 1.76966           +/- 0.003099      (0.1751%)
```

Figura 21: A dimensão do fractal

5.2 Código

Nesse código, para o centro de massa fizemos a mesma aproximação do código anterior, colocando a origem como centro, mas como agregado está movendo, a origem do agregado moverá junto com ele,portanto, cada vez que for calculado o raio, a origem vai ser diferente.

```
1  program t5
2    integer*8::a,b,c,be,n,x,y,irr
3    integer,dimension(5)::ipx,ipy
4    integer, dimension(10000,10000)::frac,fracm1,fracm2
5
6
7    write(*,*)"digite a quantidade de particulas"
8    read(*,*)be
9    write(*,*)"digite o tamanho do mapa"
10   read(*,*)l
11
12   !zerar as variav is para caso o c digo  j   foi rodado outro vez
13   r=0
14   b=1
15   entro=0
```

```
16  x_0=l/2
17  y_0=l/2
18
19  do j=1,l!zerando a matriz
20    do i=1,l
21      if(rand()>=0.9)then
22        frac(i,j)=1
23      else
24        frac(i,j)=0
25      endif
26    enddo
27  enddo
28
29  do j=1,l!zerando a matriz
30    do i=1,l
31        fracm1(i,j)=0
32    enddo
33  enddo
34
35  fracm1(l/2,l/2)=2
36   do i=1,l!salvando os dados em um arquivo
37        write(1,*)(fracm1(a,i),a=1,l)
38      enddo
39
40  !do  que ira realizar a movimenta  o do numero de bebados escolhidos
     pelo usuario
41  do while(b<=be)!particulas agregados
42
43    !do para fazer contagem de passos do bebados nos 4 sentidos
44    do while(9>3)
45      !condi  es para ver qual caminho bebados v  o fazer
46
47      ale=rand()
48      if(ale<=0.25)then
49      x_0=x_0+1
50         do j=1,l!reajustando o valores das matrizes
51            do i=1,l
52                fracm2(i+1,j)=fracm1(i,j)
53            enddo
54          enddo
55
56      else if (0.25<ale .and. ale<=0.5)then
57      x_0=x_0-1
58        do j=1,l!reajustando o valores das matrizes
59            do i=1,l
60                if(i-1>0)then!evitar e zero
61                  fracm2(i-1,j)=fracm1(i,j)
62                endif
63            enddo
64          enddo
65
66      else if (0.5<ale .and. ale<=0.75)then
67      y_0=y_0+1
68        do j=1,l!reajustando o valores das matrizes
```

```
69              do i=1,l
70                  fracm2(i,j+1)=fracm1(i,j)
71              enddo
72            enddo
73
74        else if (0.75<ale)then
75
76        y_0=y_0-1
77          do j=1,l!reajustando o valores das matrizes
78              do i=1,l
79                if(j-1>0)then!evitar o zero
80                  fracm2(i,j-1)=fracm1(i,j)
81                endif
82              enddo
83            enddo
84
85        endif
86        do j=1,l!repassando a matriz alterada para matriz original
87            do i=1,l
88              fracm1(i,j)=fracm2(i,j)
89            enddo
90        enddo
91
92
93        do x=1,l
94          do y=1,l
95            if((x-1)*(y-1)>0)then
96              if (((frac(x+1,y)+frac(x-1,y)+frac(x,y+1)+frac(x,y-1))>=1) .
    and. fracm1(x,y)==2)then!procurando uma particula
97                entro=1
98                if(frac(x+1,y)==1 .and. fracm1(x,y)==2)then
99                  fracm1(x+1,y)=2!quando a particula encontra o agregado
100                 frac(x+1,y)=0
101                 b=b+1
102               endif
103
104               if(x-1>0)then
105                 if(frac(x-1,y)==1 .and. fracm1(x,y)==2)then
106                   fracm1(x-1,y)=2!quando a particula encontra o agregado
107                   frac(x-1,y)=0
108                   b=b+1
109                 endif
110               endif
111
112               if(frac(x,y+1)==1 .and. fracm1(x,y)==2)then
113                 fracm1(x,y+1)=2!quando a particula encontra o agregado
114                 frac(x,y+1)=0
115                 b=b+1
116               endif
117
118               if(y-1>0)then
119                 if(frac(x,y-1)==1 .and. fracm1(x,y)==2)then
120                   fracm1(x,y-1)=2!quando a particula encontra o agregado
121                   frac(x,y-1)=0
```

```
                    b=b+1
                  endif
                endif

              endif
            endif
          enddo
        enddo

        do x=1,l
          do y=1,l
            if(fracm1(x,y)==2 .and. sqrt((x-x_0)**2+(y-y_0)**2)>r)then
              r=sqrt((x-x_0)**2+(y-y_0)**2)
            endif
          enddo
        enddo
        write(10,*)r,b

        if(entro==1)then!check
          entro=0
          exit
        endif

      enddo

    enddo

    do i=1,l!salvando os dados em um arquivo
        write(7,*)(frac(a,i),a=1,l)
        write(8,*)(fracm1(a,i),a=1,l)
        write(9,*)(fracm2(a,i),a=1,l)
    enddo


end program
```