Utvikling av kode til masteroppgave

presentasjon

Stian B. Søisdal

2. september 2020

Oversikt

- Mens jeg skrev masteroppgaven min, brukte jeg to programmer.

Oversikt

- Mens jeg skrev masteroppgaven min, brukte jeg to programmer.
- Først program er basert på Program 1. fra FRACTALS EVERYWHERE, MICHAEL F. BARNSLEY

Oversikt

- Mens jeg skrev masteroppgaven min,
  brukte jeg to programmer.
- Først program er basert på Program 1. fra
  FRACTALS EVERYWHERE, MICHAEL F. BARNSLEY
- Jeg går igjenom SystemM2.py
  (P2, P3, P4 og M2 kunne ha blitt laget i et program, men med 3 måneder tidsfrist og masteroppgave som skulle bli skrevet ved siden av, ble det kjapere å skrive dem enkelt vis.)

Oversikt

- Mens jeg skrev masteroppgaven min,
  brukte jeg to programmer.
- Først program er basert på Program 1. fra
  FRACTALS EVERYWHERE, MICHAEL F. BARNSLEY
- Jeg går igjenom SystemM2.py
  (P2, P3, P4 og M2 kunne ha blitt laget i et program, men med 3 måneder tidsfrist og masteroppgave som skulle bli skrevet ved siden av, ble det kjapere å skrive dem enkelt vis.)
- Andre program er basert på Program 2. fra
  FRACTALS EVERYWHERE, MICHAEL F. BARNSLEY

Oversikt

- Mens jeg skrev masteroppgaven min,
  brukte jeg to programmer.
- Først program er basert på Program 1. fra
  FRACTALS EVERYWHERE, MICHAEL F. BARNSLEY
- Jeg går igjenom SystemM2.py
  (P2, P3, P4 og M2 kunne ha blitt laget i et program, men med 3 måneder tidsfrist og masteroppgave som skulle bli skrevet ved siden av, ble det kjapere å skrive dem enkelt vis.)
- Andre program er basert på Program 2. fra
  FRACTALS EVERYWHERE, MICHAEL F. BARNSLEY
- Går igjenom RandomM2.py
  (RandomP3.py er basert på samme kode.)

Algoritme til Program 1

- Ta et set, en ”figur”, A_0.

Algoritme til Program 1

- Ta et set, en ”figur”, A_0.
- Ta også et sett med funksjoner, f_1, f_2,...

Algoritme til Program 1

- Ta et set, en ”figur”, A_0.
- Ta også et sett med funksjoner, f_1, f_2,...
- Hver av funksjonene brukes på setet, så det blir et nyt set som er en kombinasjon av alle avbildingene.

Algoritme til Program 1

- Ta et set, en ”figur”, A_0.
- Ta også et sett med funksjoner, f_1, f_2,...
- Hver av funksjonene brukes på setet, så det blir et nyt set som er en kombinasjon av alle avbildingene.
- $A_1 = f_1(A_0) \cup f_2(A_0) \cup ...$

Algoritme til Program 1

- Ta et set, en ”figur”, A_0.
- Ta også et sett med funksjoner, f_1, f_2,...
- Hver av funksjonene brukes på setet, så det blir et nyt set som er en kombinasjon av alle avbildingene.
- $A_1 = f_1(A_0) \cup f_2(A_0) \cup ...$
- Itererer et viss antall ganger, og plotter resultatet.

Program 1 fra FRACTALS EVERYWHERE, MICHAEL F. BARNSLEY

- The program is written in BASIC.

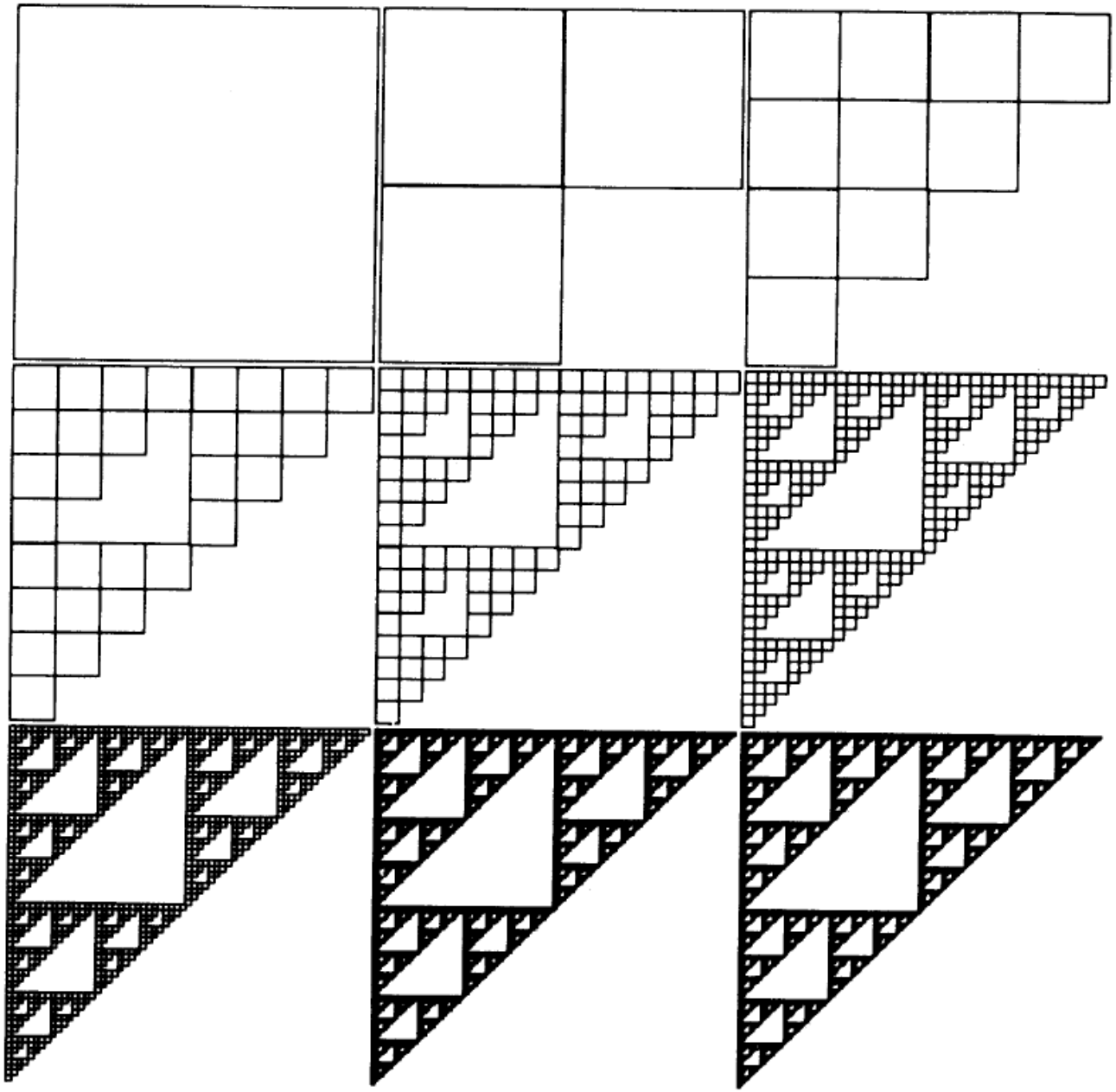
Table III.1. IFS code for a Sierpinski triangle.

w	a	b	c	d	e	f	p
1	0.5	0	0	0.5	1	1	0.33
2	0.5	0	0	0.5	1	50	0.33
3	0.5	0	0	0.5	50	50	0.34

Program 1. (Example of the Deterministic Algorithm)

```
screen 1 : cls 'initialize graphics
dim s(100,100) : dim t(100,100) 'allocate two arrays of pixels
a(1)=0.5:b(1)=0:c(1)=0:d(1)=0.5:e(1)=1:f(1)=1 'input the IFS code
a(2)=0.5:b(2)=0:c(2)=0:d(2)=0.5:e(2)=50:f(2)=1
a(3)=0.5:b(3)=0:c(3)=0:d(3)=0.5:e(3)=25:f(3)=50
for i=1 to 100 'input the initial set A(0), in this case
     a square, into the array t(i,j)

t(i,1)=1: pset(i,1) 'A(0) can be used as a condensation set
t(1,i)=1:pset(1,i)  'A(0) is plotted on the screen
t(100,i)=1:pset(100,i)
t(i,100)=1:pset(i,100)
next: do
for i=1 to 100 'apply W to set A(n) to make A(n+1) in the
      array s(i,j)
for j=1 to 100 : if t(i,j)=1 then
s(a(1)*i+b(1)*j+e(1),c(1)*i+d(1)*j+f(1))=1 'and apply W to A(n)
s(a(2)*i+b(2)*j+e(2),c(2)*i+d(2)*j+f(2))=1
s(a(3)*i+b(3)*j+e(3),c(3)*i+d(3)*j+f(3))=1
end if: next j: next i
cls 'clears the screen--omit to obtain sequence with a A(0) as
     condensation set (see section 9 in Chapter II)
for i=1 to 100 : for j=1 to 100
t(i,j)=s(i,j) 'put A(n+1) into the array t(i,j)
s(i,j)=0 'reset the array s(i,j) to zero
if t(i,j)=1 then
pset(i,j) 'plot A(n+1)
end if : next : next
loop until instat 'if a key has been pressed then stop,
     otherwise compute A(n+1)=W(A(n+1))
```



SystemM2.py

- Denne koden starter med å definere funksjonene som lager et fraktalt mønster.

```
239     plttit = 'System generating \n a Fern attractor'
240     t = 0.900116
241     A = np.array([np.matrix([[1.0, 0, 0, 0],
242                              [0.0, 1e-3, 0, 0],
243                              [0.0, 0, 1e-3, 0],
244                              [0.0, 0, 0, 0.18]]),
245                   np.matrix([[1.0,    0,    0,    0],
246                              [0.0,    0.85, 0,   -0.1],
247                              [0.0,    0,    0.85, 0],
248                              [0.16*t, 0.1,  0,    0.85]]),
249                   np.matrix([[1.0,    0,   0,   0],
250                              [0.0,    0.3, 0,   0],
251                              [0.0,    0,   0.2,-0.2],
252                              [0.08*t, 0,   0.2, 0.2]]),
253                   np.matrix([[1.0,    0,   0,   0],
254                              [0.0,    0.3, 0,   0],
255                              [0.0,    0,  -0.2, 0.2],
256                              [0.08*t, 0,   0.2, 0.2]])])
257     savefile = 'Fig_M2_5-.png'
```

SystemM2.py

- Denne koden starter med å definere funksjonene som lager et fraktalt mønster.
- Her er de gitt som en vector av matriser, A.

```
239     plttit = 'System generating \n a Fern attractor'
240     t = 0.900116
241     A = np.array([np.matrix([[1.0, 0, 0, 0],
242                              [0.0, 1e-3, 0, 0],
243                              [0.0, 0, 1e-3, 0],
244                              [0.0, 0, 0, 0.18]]),
245                   np.matrix([[1.0,    0,    0,    0],
246                              [0.0,    0.85, 0,   -0.1],
247                              [0.0,    0,    0.85, 0],
248                              [0.16*t, 0.1,  0,    0.85]]),
249                   np.matrix([[1.0,    0,   0,   0],
250                              [0.0,    0.3, 0,   0],
251                              [0.0,    0,   0.2,-0.2],
252                              [0.08*t, 0,   0.2, 0.2]]),
253                   np.matrix([[1.0,    0,   0,   0],
254                              [0.0,    0.3, 0,   0],
255                              [0.0,    0,  -0.2, 0.2],
256                              [0.08*t, 0,   0.2, 0.2]])])
257     savefile = 'Fig_M2_5-.png'
```

SystemM2.py

- Denne koden starter med å definere funksjonene som lager et fraktalt mønster.
- Her er de gitt som en vector av matriser, A.
- Starter også med et start set, gitt ved X, Y, Z.

```
239     plttit = 'System generating \n a Fern attractor'
240     t = 0.900116
241     A = np.array([np.matrix([[1.0, 0, 0, 0],
242                              [0.0, 1e-3, 0, 0],
243                              [0.0, 0, 1e-3, 0],
244                              [0.0, 0, 0, 0.18]]),
245                   np.matrix([[1.0,    0,    0,    0],
246                              [0.0,    0.85, 0,   -0.1],
247                              [0.0,    0,    0.85, 0],
248                              [0.16*t, 0.1,  0,    0.85]]),
249                   np.matrix([[1.0,    0,   0,   0],
250                              [0.0,    0.3, 0,   0],
251                              [0.0,    0,   0.2,-0.2],
252                              [0.08*t, 0,   0.2, 0.2]]),
253                   np.matrix([[1.0,    0,   0,   0],
254                              [0.0,    0.3, 0,   0],
255                              [0.0,    0,  -0.2, 0.2],
256                              [0.08*t, 0,   0.2, 0.2]])])
257     savefile = 'Fig_M2_5-.png'

285     # For fern
286     r_s = 0.15
287     u = np.linspace(0, 2 * np.pi, 5)
288     v = np.linspace(0, np.pi, 4)
289
290     X = r_s*np.outer(np.cos(u), np.sin(v))
291     Y = r_s*np.outer(np.sin(u), np.sin(v))
292     Z = 2*r_s*np.outer(np.ones(np.size(u)), np.cos(v))+3/10
293     '''
294     SystemOnM2(A, X, Y, Z, itr, plttit, savefile)
```

SystemM2.py

- Denne koden starter med å definere funksjonene som lager et fraktalt mønster.
- Her er de gitt som en vector av matriser, A.
- Starter også med et start set, gitt ved X, Y, Z.
- SystemOnM2(...) er laget som en python funksjon, jeg kunne laget en class også.

```
239     plttit = 'System generating \n a Fern attractor'
240     t = 0.900116
241     A = np.array([np.matrix([[1.0, 0, 0, 0],
242                              [0.0, 1e-3, 0, 0],
243                              [0.0, 0, 1e-3, 0],
244                              [0.0, 0, 0, 0.18]]),
245                   np.matrix([[1.0,    0,    0,    0],
246                              [0.0,    0.85, 0,   -0.1],
247                              [0.0,    0,    0.85, 0],
248                              [0.16*t, 0.1,  0,    0.85]]),
249                   np.matrix([[1.0,    0,   0,   0],
250                              [0.0,    0.3, 0,   0],
251                              [0.0,    0,   0.2,-0.2],
252                              [0.08*t, 0,   0.2, 0.2]]),
253                   np.matrix([[1.0,    0,   0,   0],
254                              [0.0,    0.3, 0,   0],
255                              [0.0,    0,  -0.2, 0.2],
256                              [0.08*t, 0,   0.2, 0.2]])])
257     savefile = 'Fig_M2_5-.png'

285     # For fern
286     r_s = 0.15
287     u = np.linspace(0, 2 * np.pi, 5)
288     v = np.linspace(0, np.pi, 4)
289
290     X = r_s*np.outer(np.cos(u), np.sin(v))
291     Y = r_s*np.outer(np.sin(u), np.sin(v))
292     Z = 2*r_s*np.outer(np.ones(np.size(u)), np.cos(v))+3/10
293     '''
294     SystemOnM2(A, X, Y, Z, itr, plttit, savefile)
```

- Antall iterasjoner (itr, it) bestemmer hvordan oppsette på plotet blir. (for å gjøre det kjapt, ble en zoom lagt inn manuelt, trengte det kun en gang.)

```
85      # Pre-Iteration:
86      fig = plt.figure(figsize=(figsizx[it],figsizy[it]))
87      fig.suptitle(pltit, fontsize=16)
88      if it != 4:
89          ax = fig.add_subplot(plot_layout[it], projection='3d')
90          plt.title('Initial set'), ax.view_init(20, 10)
91          ax.set_xlim3d([-1, 1]), ax.set_ylim3d([-1, 1]), ax.set_zlim3d([-1, 1])
92          #ax.set_xlim3d([-0.7,0.1]), ax.set_ylim3d([-0.25,0.25]), ax.set_zlim3d([0,0.8])
93          ax.plot_surface(Sx, Sy, Sz, color=(1.,0.,0.,.1))
94          ax.plot_surface(x, y, z, color=(0.1,0.4,1.,0.95))
```

- Antall iterasjoner (itr, it) bestemmer hvordan oppsette på plotet blir. (for å gjøre det kjapt, ble en zoom lagt inn manuelt, trengte det kun en gang.)
- Ser hvor mange funksjoner som er gitt.

```
96      # System setup
97      sys, An = len(a), np.zeros_like(a)
98      for i in range(sys):
99          An[i] = np.matrix([[1.0, 0, 0, 0],
100                             [0.0, 1, 0, 0],
101                             [0.0, 0, 1, 0],
102                             [0.0, 0, 0, 1]])
103     for n in range(it):
104         # Setup plot to iteration
105         ax = fig.add_subplot(plot_layout[it]+n+1, projection='3d')
106         plt.title('Iteration {}'.format(n+1)), ax.view_init(20, 10)
```

- Antall iterasjoner (itr, it) bestemmer hvordan oppsette på plotet blir. (for å gjøre det kjapt, ble en zoom lagt inn manuelt, trengte det kun en gang.)
- Ser hvor mange funksjoner som er gitt.
- ”An” lagrer alle måtene å kombinere funksjonene (itr ganger) på det originale bilde.

```
96      # System setup
97      sys, An = len(a), np.zeros_like(a)
98      for i in range(sys):
99          An[i] = np.matrix([[1.0, 0, 0, 0],
100                             [0.0, 1, 0, 0],
101                             [0.0, 0, 1, 0],
102                             [0.0, 0, 0, 1]])
103     for n in range(it):
104         # Setup plot to iteration
105         ax = fig.add_subplot(plot_layout[it]+n+1, projection='3d')
106         plt.title('Iteration {}'.format(n+1)), ax.view_init(20, 10)
```

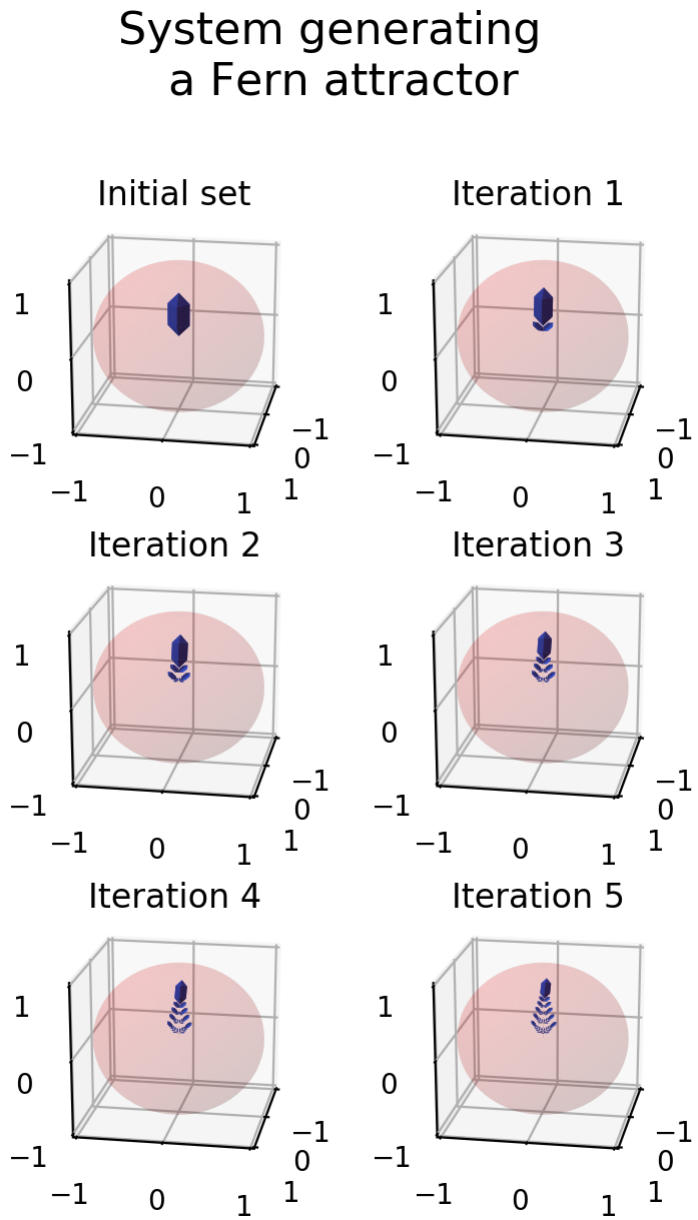
```
111             # Allocate new maps
112             An = np.repeat(An, sys, axis=0)
113
114             # Update maps
115             for k in range(sys**(n+1)):
116                 k_re = np.remainder(k,sys)
117                 Ak = An[k]
118                 An[k] = np.matmul(a[k_re], Ak)
```

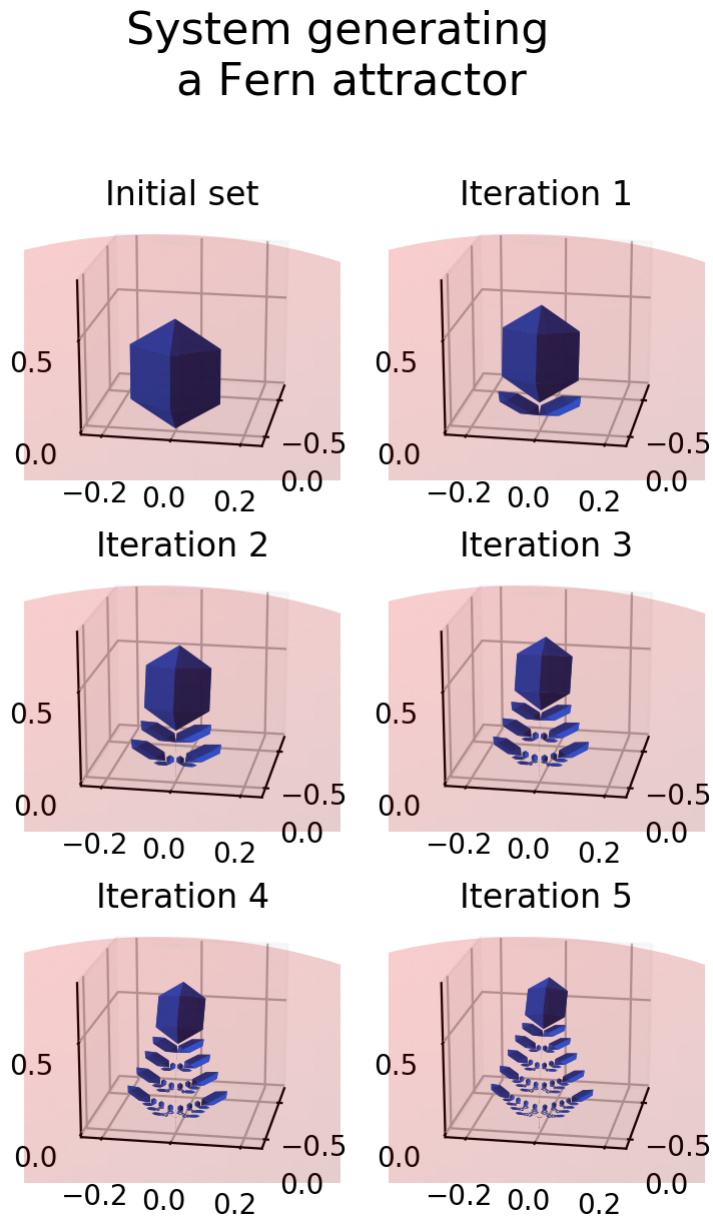
- Antall iterasjoner (itr, it) bestemmer hvordan oppsette på plotet blir. (for å gjøre det kjapt, ble en zoom lagt inn manuelt, trengte det kun en gang.)
- Ser hvor mange funksjoner som er gitt.
- ”An” lagrer alle måtene å kombinere funksjonene (itr ganger) på det originale bilde.
- Går så gjennom iterasjonene og legger inn kombinasjonene.

```
96      # System setup
97      sys, An = len(a), np.zeros_like(a)
98      for i in range(sys):
99          An[i] = np.matrix([[1.0, 0, 0, 0],
100                             [0.0, 1, 0, 0],
101                             [0.0, 0, 1, 0],
102                             [0.0, 0, 0, 1]])
103     for n in range(it):
104         # Setup plot to iteration
105         ax = fig.add_subplot(plot_layout[it]+n+1, projection='3d')
106         plt.title('Iteration {}'.format(n+1)), ax.view_init(20, 10)
```

```
111             # Allocate new maps
112             An = np.repeat(An, sys, axis=0)
113
114             # Update maps
115             for k in range(sys**(n+1)):
116                 k_re = np.remainder(k,sys)
117                 Ak = An[k]
118                 An[k] = np.matmul(a[k_re], Ak)
```

Ploter så iterasjonene og får (andre plot er med manuell zoom)





Har også en del i SystemM2.py som tester at vise likninger holder. Likningene kan fines på side 15/23 i masterpresentasjonen.

```
14  # Test that the positive trace-preserving maps are CPT maps
15  def CPTtest(a):
16      for i in range(len(a)):
17          eq = 0
18          if (abs(a[i,3,3])+abs(a[i,3,0])) -1.0 == 0.0:
19              if a[i,1,0] == 0:
20                  if a[i,2,0] == 0:
21                      eq = 4
22                  else:
23                      print('T_{} do not hold 1. and 2.'.format(i+1))
24              else:
25                  print('T_{} do not hold 1. and 2.'.format(i+1))
26          else:
```

Algoritme til Program 2

- Ta et start punkt, x_0.

Algoritme til Program 2

- Ta et start punkt, x_0.
- Velg tilfeldig fra et set funksjoner, f_{i_1}.

Algoritme til Program 2

- Ta et start punkt, x_0.
- Velg tilfeldig fra et set funksjoner, f_{i_1}.
- Får så punktet $x_1 = f_{i_1}(x_0)$.

Algoritme til Program 2

- Ta et start punkt, x_0.
- Velg tilfeldig fra et set funksjoner, f_{i_1}.
- Får så punktet $x_1 = f_{i_1}(x_0)$.
- Neste punkt blir $x_2 = f_{i_2}(x_1)$, hvor f_{i_2} en tilfeldig funksjon fra samme set.

Algoritme til Program 2

- Ta et start punkt, x_0.
- Velg tilfeldig fra et set funksjoner, f_{i_1}.
- Får så punktet $x_1 = f_{i_1}(x_0)$.
- Neste punkt blir $x_2 = f_{i_2}(x_1)$, hvor f_{i_2} en tilfeldig funksjon fra samme set.
- Itererer et viss antall ganger, og plotter resultatet.

Program 2 fra FRACTALS EVERYWHERE, MICHAEL F. BARNSLEY

- The program is written in BASIC.

Table III.3. IFS code for a fern.

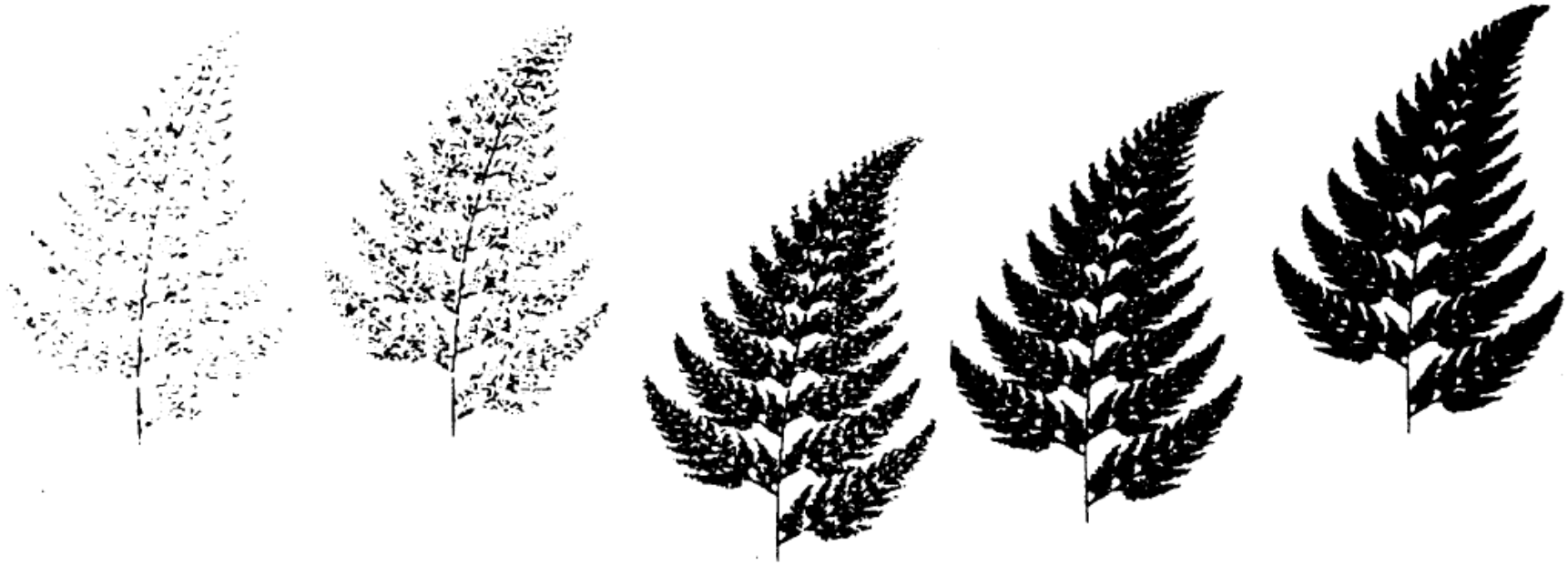
w	a	b	c	d	e	f	p
1	0	0	0	0.16	0	0	0.01
2	0.85	0.04	-0.04	0.85	0	1.6	0.85
3	0.2	-0.26	0.23	0.22	0	1.6	0.07
4	-0.15	0.28	0.26	0.24	0	0.44	0.07

Program 2. (Example of the Random Iteration Algorithm)

```
  'Iterated Function System Data
a[1] = 0.5 : b[1] =0 : c[1] =0 : d[1] =.5 : e[1] =1 : f[1] =1
a[2] = 0.5 : b[2] =0 : c[2] =0 : d[2] =.5 : e[2] =50 : f[2] =1
a[3] = 0.5 : b[3] =0 : c[3] =0 : d[3] =.5 : e[3] =50 : f[3] =50

screen 1 : cls 'initialize computer graphics
window (0,0)-(100,100) 'set plotting window to 0<x<100, 0<y<100
x =0  : y  =  0: numits =1000  'initialize (x,y) and define
     the number of iterations, numits
for n =1  to numits 'Random Iteration begins!
k  =  int(3*rnd-0.00001)  +1  'choose one of the numbers 1, 2,
     and 3 with equal probability

'apply affine transformation number k to (x,y)
newx =a[k]*x+b[k]*y+e[k]  : newy =c[k]*x+d[k]*y+f[k]
x =newx  : y =newy  'set (x,y) to the point thus obtained
if n  >  10 then pset (x,y) 'plot (x,y) after the first 10
     iterations
next :   end
```



RandomM2.py

- I denne koden trengs funksjonene og en vektor av hvor sannsynlig funksjonene er.

```
85      # Fern
86      plttit = 'Random point approximation \n of the Fern attractor'
87      t = 0.900116
88      A = np.array([np.matrix([[1.0, 0, 0, 0],
89                               [0.0, 0, 0, 0],
90                               [0.0, 0, 0, 0],
91                               [0.0, 0, 0, 0.18]]),
92                    np.matrix([[1.0,    0,    0,    0],
93                               [0.0,    0.85, 0,   -0.1],
94                               [0.0,    0,    0.85, 0],
95                               [0.16*t, 0.1,  0,    0.85]]),
96                    np.matrix([[1.0,    0,   0,   0],
97                               [0.0,    0.3, 0,   0],
98                               [0.0,    0,   0.2,-0.2],
99                               [0.08*t, 0,   0.2, 0.2]]),
100                   np.matrix([[1.0,    0,   0,   0],
101                              [0.0,    0.3, 0,   0],
102                              [0.0,    0,  -0.2, 0.2],
103                              [0.08*t, 0,   0.2, 0.2]])])
104     P = np.array([0.01, 0.85, 0.07, 0.07])
105     X0= np.array([0,0,0]) # Start point
106     Zoom = np.array([-0.7,0.1, -0.25,0.25, 0,0.8]) #set axis
107     savefile = 'Fig_M2_5R.png'
```

RandomM2.py

- I denne koden trengs funksjonene og en vektor av hvor sannsynlig funksjonene er.
- Zoom ble lagt til ordentlig.

```
85      # Fern
86      plttit = 'Random point approximation \n of the Fern attractor'
87      t = 0.900116
88      A = np.array([np.matrix([[1.0, 0, 0, 0],
89                               [0.0, 0, 0, 0],
90                               [0.0, 0, 0, 0],
91                               [0.0, 0, 0, 0.18]]),
92                    np.matrix([[1.0,    0,    0,    0],
93                               [0.0,    0.85, 0,   -0.1],
94                               [0.0,    0,    0.85, 0],
95                               [0.16*t, 0.1,  0,    0.85]]),
96                    np.matrix([[1.0,    0,   0,   0],
97                               [0.0,    0.3, 0,   0],
98                               [0.0,    0,   0.2,-0.2],
99                               [0.08*t, 0,   0.2, 0.2]]),
100                   np.matrix([[1.0,    0,   0,   0],
101                              [0.0,    0.3, 0,   0],
102                              [0.0,    0,  -0.2, 0.2],
103                              [0.08*t, 0,   0.2, 0.2]])])
104     P = np.array([0.01, 0.85, 0.07, 0.07])
105     X0= np.array([0,0,0]) # Start point
106     Zoom = np.array([-0.7,0.1, -0.25,0.25, 0,0.8]) #set axis
107     savefile = 'Fig_M2_5R.png'
```

RandomM2.py

- I denne koden trengs funksjonene og en vektor av hvor sannsynlig funksjonene er.
- Zoom ble lagt til ordentlig.
- I denne python funksjonen blir...

```
85      # Fern
86      plttit = 'Random point approximation \n of the Fern attractor'
87      t = 0.900116
88      A = np.array([np.matrix([[1.0, 0, 0, 0],
89                               [0.0, 0, 0, 0],
90                               [0.0, 0, 0, 0],
91                               [0.0, 0, 0, 0.18]]),
92                    np.matrix([[1.0,    0,    0,    0],
93                               [0.0,    0.85, 0,   -0.1],
94                               [0.0,    0,    0.85, 0],
95                               [0.16*t, 0.1,  0,    0.85]]),
96                    np.matrix([[1.0,    0,   0,   0],
97                               [0.0,    0.3, 0,   0],
98                               [0.0,    0,   0.2,-0.2],
99                               [0.08*t, 0,   0.2, 0.2]]),
100                   np.matrix([[1.0,    0,   0,   0],
101                              [0.0,    0.3, 0,   0],
102                              [0.0,    0,  -0.2, 0.2],
103                              [0.08*t, 0,   0.2, 0.2]])])
104     P = np.array([0.01, 0.85, 0.07, 0.07])
105     X0= np.array([0,0,0]) # Start point
106     Zoom = np.array([-0.7,0.1, -0.25,0.25, 0,0.8]) #set axis
107     savefile = 'Fig_M2_5R.png'
```

```
14  def RandomOnM2(a, p, x0, zoom, pltit=' ', savpng=0):
15      numits = 1000000 # Number of iterations
16      # Get random array and allocate (x,y,z).
17      K, X = np.random.rand(numits), np.zeros((numits+1,3))
18      X[0] = x0
```

RandomM2.py

- I denne koden trengs funksjonene og en vektor av hvor sannsynlig funksjonene er.
- Zoom ble lagt til ordentlig.
- I denne python funksjonen blir...
- Her blir

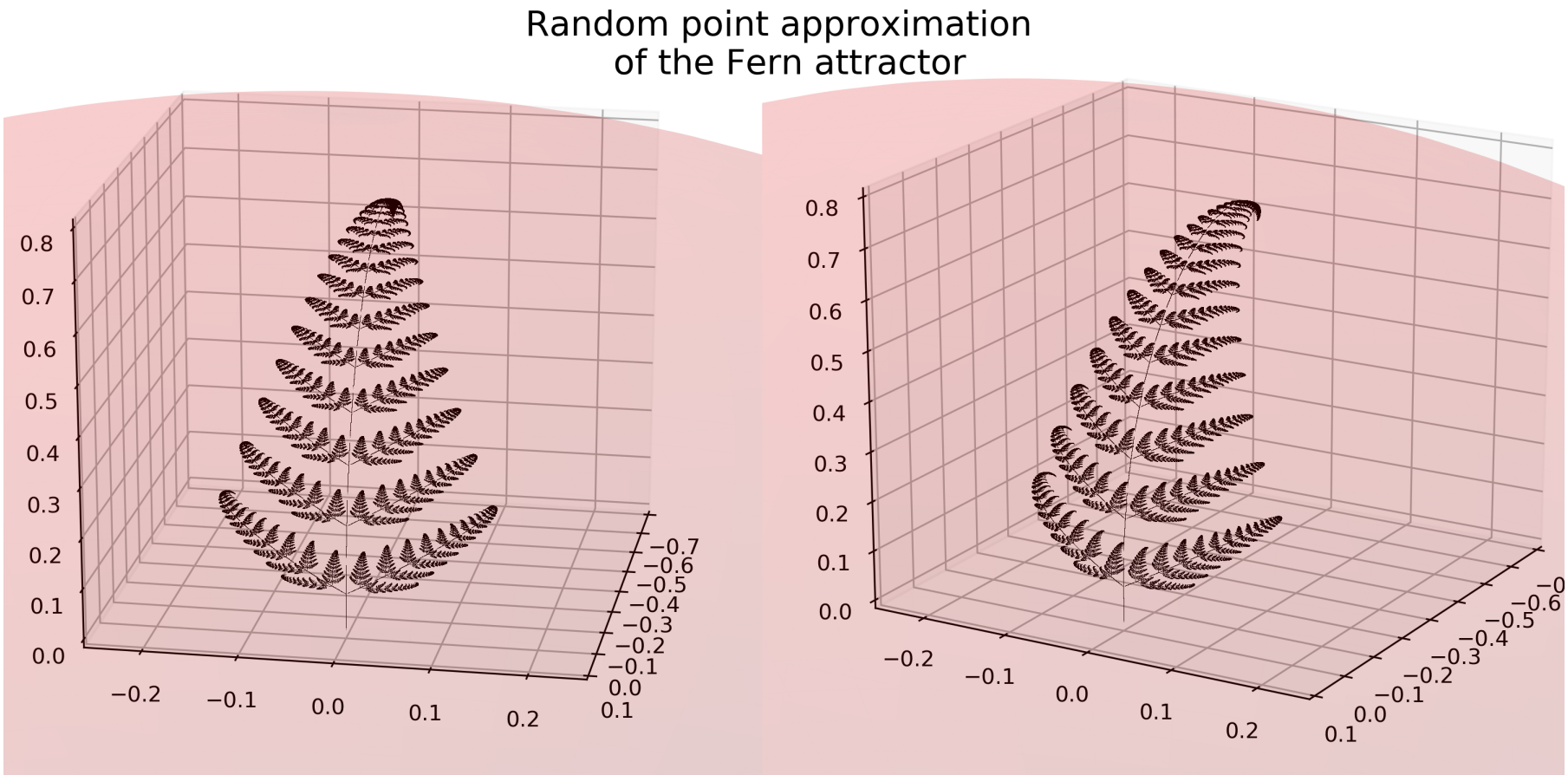
```
85      # Fern
86      plttit = 'Random point approximation \n of the Fern attractor'
87      t = 0.900116
88      A = np.array([np.matrix([[1.0, 0, 0, 0],
89                               [0.0, 0, 0, 0],
90                               [0.0, 0, 0, 0],
91                               [0.0, 0, 0, 0.18]]),
92                    np.matrix([[1.0,    0,    0,    0],
93                               [0.0,    0.85, 0,   -0.1],
94                               [0.0,    0,    0.85, 0],
95                               [0.16*t, 0.1,  0,    0.85]]),
96                    np.matrix([[1.0,    0,   0,   0],
97                               [0.0,    0.3, 0,   0],
98                               [0.0,    0,   0.2,-0.2],
99                               [0.08*t, 0,   0.2, 0.2]]),
100                   np.matrix([[1.0,    0,   0,   0],
101                              [0.0,    0.3, 0,   0],
102                              [0.0,    0,  -0.2, 0.2],
103                              [0.08*t, 0,   0.2, 0.2]])])
104     P = np.array([0.01, 0.85, 0.07, 0.07])
105     X0= np.array([0,0,0]) # Start point
106     Zoom = np.array([-0.7,0.1, -0.25,0.25, 0,0.8]) #set axis
107     savefile = 'Fig_M2_5R.png'
```

```
14  def RandomOnM2(a, p, x0, zoom, pltit=' ', savpng=0):
15      numits = 1000000 # Number of iterations
16      # Get random array and allocate (x,y,z).
17      K, X = np.random.rand(numits), np.zeros((numits+1,3))
18      X[0] = x0
```

- Plukker så ut en random funksjon for hvert steg, og bruker det på det forrige punktet.

```
26      for j in range(numits): # Geting the approximat attractor
27          i = 0
28          for k in range(len(p)): # Pick out A_i
29              if K[j] >= sum(p[0:k]):
30                  i = k
31              else:
32                  break
33          x, y, z = X[j]
34          X[j+1,0] = a[i,1,1]*x +a[i,1,2]*y +a[i,1,3]*z +a[i,1,0]
35          X[j+1,1] = a[i,2,1]*x +a[i,2,2]*y +a[i,2,3]*z +a[i,2,0]
36          X[j+1,2] = a[i,3,1]*x +a[i,3,2]*y +a[i,3,3]*z +a[i,3,0]
```

- Plukker så ut en random funksjon for hvert steg, og bruker det på det forrige punktet.
- Ploter blir så



Stian B. Søisdal

Utvikling av kode til masteroppgave

presentasjon