Affina avbildningar, itererade funktionssystem och fraktala bilder

En linjär funktion/avbildning $F : \mathbb{R}^2 \to \mathbb{R}^2$ kan skrivas $F(\underline{e}X) = \underline{e}AX$ där A är en konstant 2×2 - matris. Vi använder standardbasen och skriver utan $\underline{e}$

$$F(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

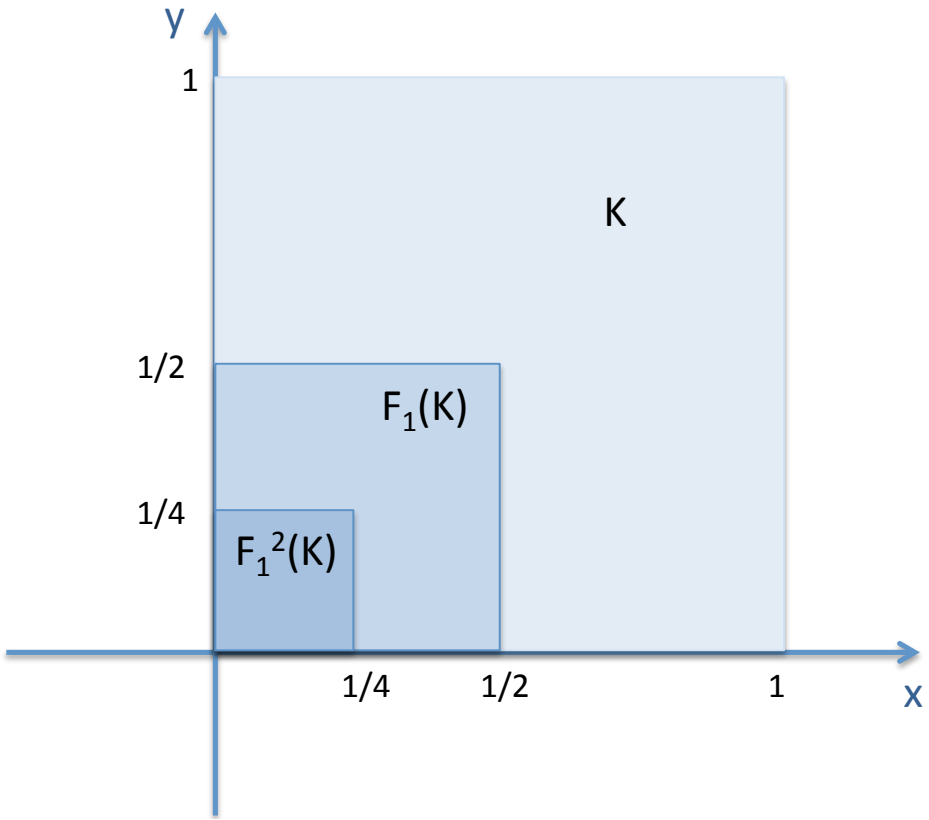
Om man till en linjär avbildning adderar en konstant vektor fås en så kallad affin avbildning:

$$F(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

För envariabelfunktioner är $f(x) = ax$ linjär och $f(x) = ax + b$ affin.

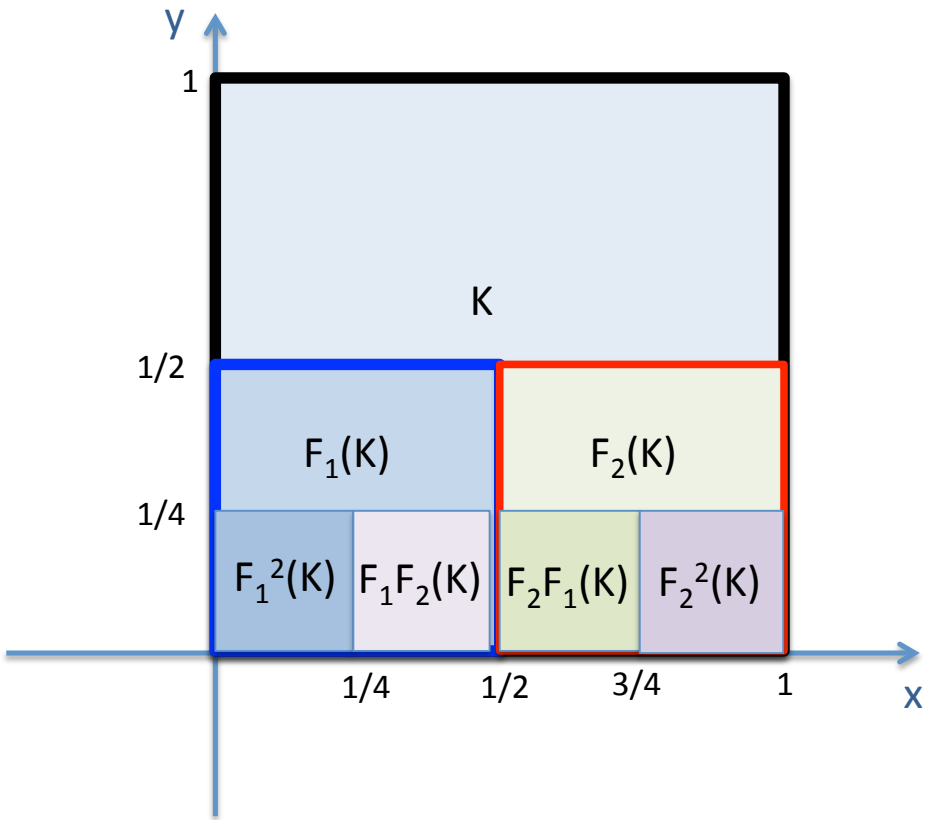
Låt $F_1(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{2} \begin{pmatrix} x \\ y \end{pmatrix}$

Låt K = enhetskvadraten och $F_1(K)$ det som K avbildas på med F_1 (en mindre kvadrat eftersom F_1 halverar alla vektorer). Om vi avbildar $F_1(K)$ med F_1 igen får vi $F_1(F_1(K)) = F_1^2(K)$ etc.



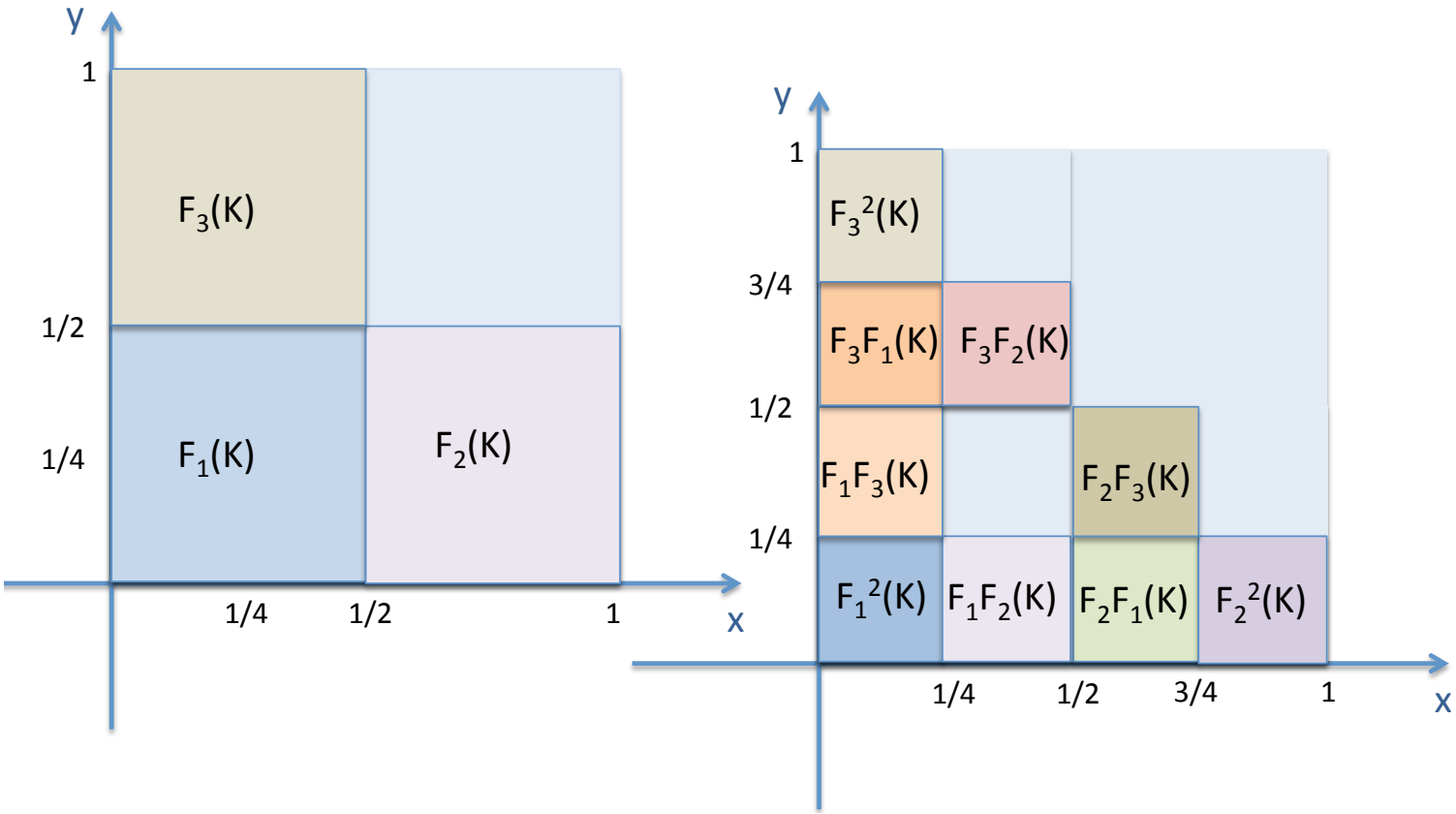
Låt $F_2\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1/2 \\ 0 \end{pmatrix}$

F_2 är affin, som F_1 men flyttar också $1/2$ i x-led.

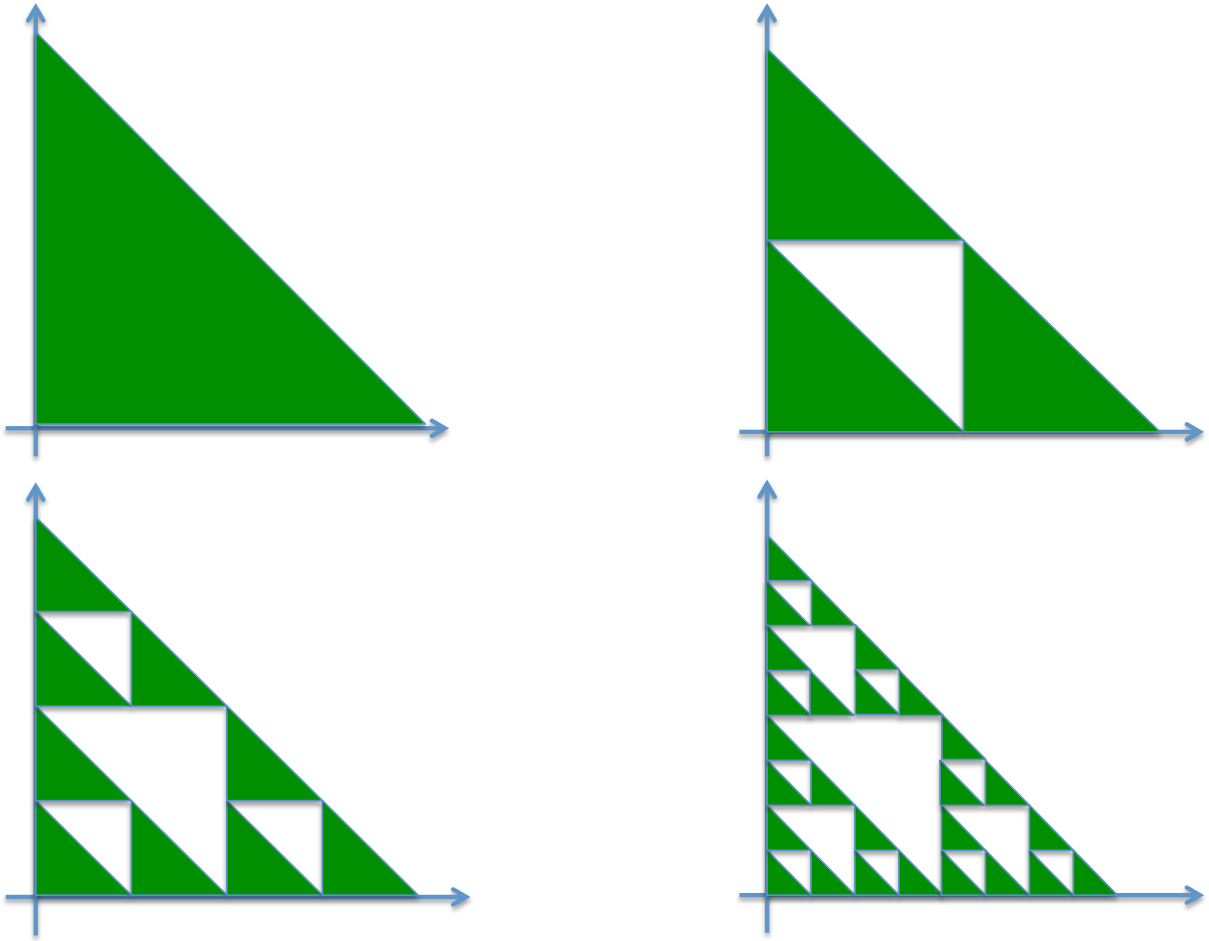


Låt $F_3\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1/2 \end{pmatrix}$

F_3 är affin, som F_2 men flyttar i stället $1/2$ i y-led.



Om vi använder trianglar i stället för kvadrater och stegar på med alla sammansättningar av F_1, F_2 och F_3 får vi till slut en fraktal bild.

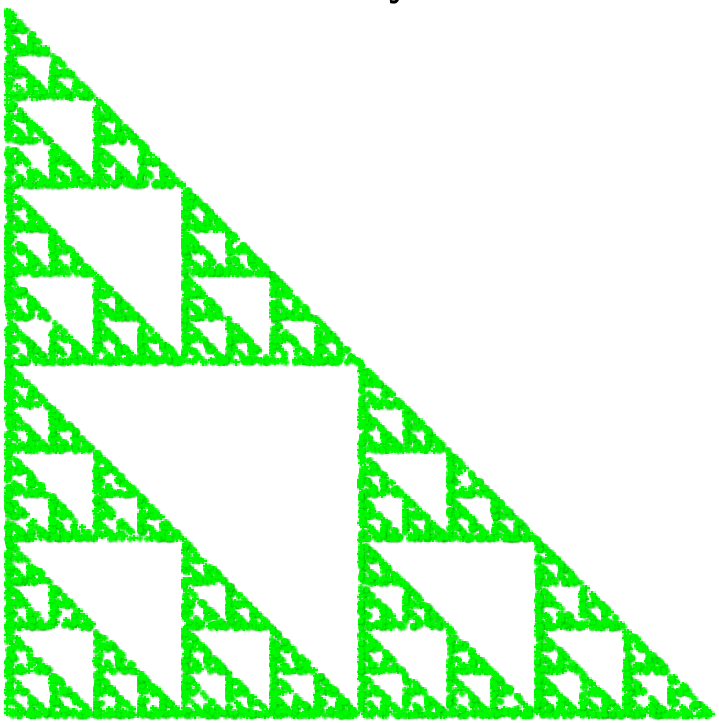


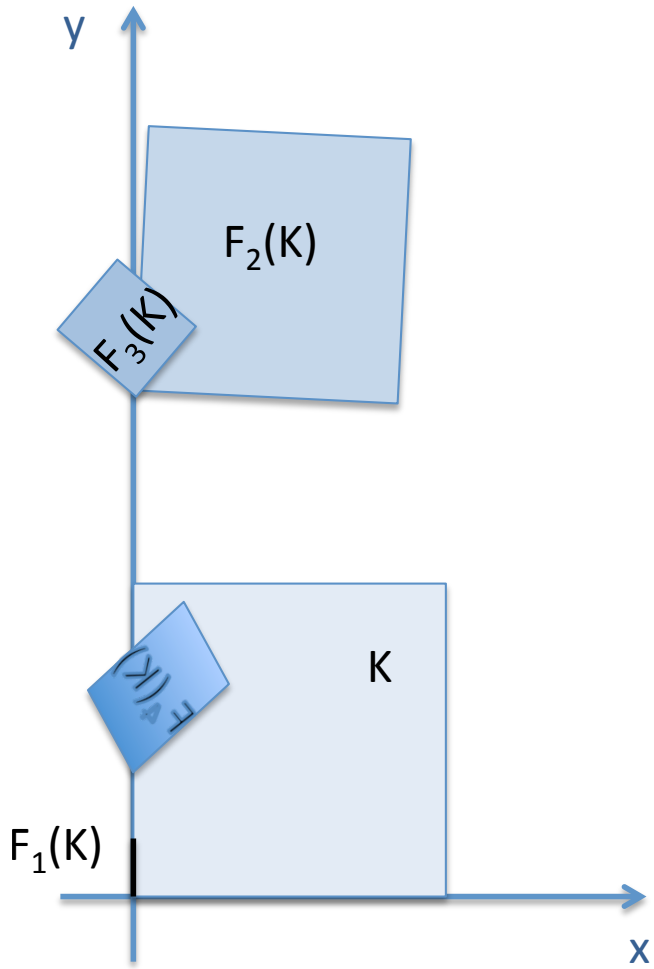
Efyter många steg fås bilden på nästa sida som också kan genereras med en slumpstegsmetod.

Om vi börjar i (0,0) och tar slumpvis en av F_1, F_2 eller F_3 kan vi t.ex. få

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \overset{F_2}{\mapsto} \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix} \overset{F_3}{\mapsto} \begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \end{pmatrix} \overset{F_3}{\mapsto} \begin{pmatrix} \frac{1}{8} \\ \frac{3}{4} \end{pmatrix} \overset{F_1}{\mapsto} \begin{pmatrix} \frac{1}{16} \\ \frac{3}{8} \end{pmatrix} \overset{F_3}{\mapsto} \begin{pmatrix} \frac{1}{32} \\ \frac{11}{16} \end{pmatrix} \overset{F_2}{\mapsto}$$
$$\begin{pmatrix} \frac{33}{64} \\ \frac{11}{32} \end{pmatrix} \overset{F_2}{\mapsto} \begin{pmatrix} \frac{97}{128} \\ \frac{11}{64} \end{pmatrix} \overset{F_1}{\mapsto} \begin{pmatrix} \frac{97}{256} \\ \frac{11}{128} \end{pmatrix} \overset{F_3}{\mapsto} \begin{pmatrix} \frac{97}{512} \\ \frac{139}{256} \end{pmatrix} \overset{F_1}{\mapsto} \dots$$

Om vi tar 10.000 steg och plottar alla punkter får vi en fraktal bild som är samma som fås med systematiska metoden på förra sidan:





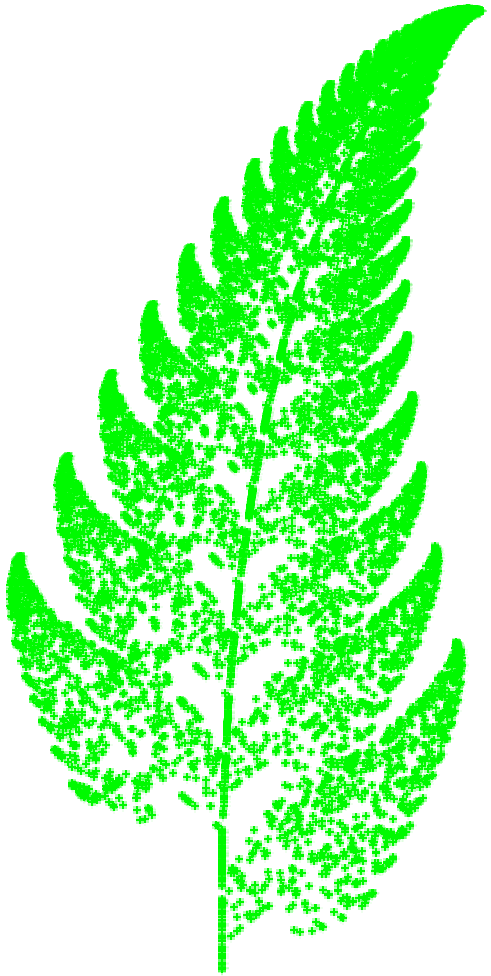
Låt nu $F_1\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} 0 & 0 \\ 0 & 0.16 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

$$F_2\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1.6 \end{pmatrix}$$

$$F_3\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} 0.2 & -0.26 \\ 0.23 & 0.22 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1.6 \end{pmatrix}$$

$$F_4\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0.44 \end{pmatrix}$$

Om vi börjar i (0,0) och tar steg med F_1, F_2, F_3 eller F_4 med sannolikheter 0.01, 0.85, 0.07 resp. 0.07, får vi om vi plottar 10.000 första punkterna:



Ett itererat funktionssystem (IFS) består av ett antal affina funktioner $F_1, \ldots, F_n$ med tillhörande sannolikheter $p_1, \ldots, p_n$. Om man stegar sig fram och plottar alla punkter fås en fraktal bild. IFS kan användas i datorgrafik.

Omvänt, om man har en given bild finns metoder för att approximera den med en fraktal bild och hitta affina funktioner och ett IFS som leder till bilden. Detta ger en metod för bildkompression genom lagring av en bild med mycket mindre information (affina funktionerna + sannolikheter) än vad som behövs för att lagra alla pixlar. I stället för att ange hur bilden ser ut i minsta detalj anger man ett enkel algoritm som skapar den.

Vi kommer senare se andra metoder från linjär algebra som används för bildkompression.

Maple-koder för de IFS som ger bilderna ovan.

```
with(stats[random]) : with(plots) :
 n := 10000 :
Mat1 := linalg[matrix]([[0.5, 0.0], [0.0, 0.5]]) :
Mat2 := linalg[matrix]([[0.5, 0.0], [0.0, 0.5]]) :
Mat3 := linalg[matrix]([[0.5, 0.0], [0.0, 0.5]]) :
Vector1 := linalg[vector]([0, 0]) :
Vector2 := linalg[vector]([0, 0.5]) :
Vector3 := linalg[vector]([0.5, 0]) :
Prob1 := 1/3 :
Prob2 := 1/3 :
Prob3 := 1/3 :
P := linalg[vector]([0, 0]) :
for m from 1 to n do
  prob := uniform( ) :
  if prob < Prob1 then P := evalm(Mat1&*P + Vector1)
  elif prob < Prob1 + Prob2 then P := evalm(Mat2&*P + Vector2)
  else P := evalm(Mat3&*P + Vector3);
  fi:
 A[m, 1] := P[1] : A[m, 2] := P[2] :
od:
C := matrix(n, 2, (i, j) → A[i, j]) :
pointplot(C, scaling = constrained, axes = none, color = green);
```

```
with(stats[random]) : with(plots) :
 n := 10000 :
Mat1 := linalg[matrix]([[0.0, 0.0], [0.0, 0.16]]) :
Mat2 := linalg[matrix]([[0.85, 0.04], [-0.04, 0.85]]) :
Mat3 := linalg[matrix]([[0.2, -0.26], [0.23, 0.22]]) :
Mat4 := linalg[matrix]([[-0.15, 0.28], [0.26, 0.24]]) :
Vector1 := linalg[vector]([0, 0]) :
Vector2 := linalg[vector]([0, 1.6]) :
Vector3 := linalg[vector]([0, 1.6]) :
Vector4 := linalg[vector]([0, 0.44]) :
Prob1 := 0.01 :
Prob2 := 0.85 :
Prob3 := 0.07 :
Prob4 := 0.07 :
P := linalg[vector]([0, 0]) :
for m from 1 to n do
  prob := uniform( ) :
  if prob < Prob1 then P := evalm(Mat1&*P + Vector1)
  elif prob < Prob1 + Prob2 then P := evalm(Mat2&*P + Vector2)
  elif prob < Prob1 + Prob2 + Prob3 then P := evalm(Mat3&*P + Vector3)
   else P := evalm(Mat4&*P + Vector4);
  fi:
  A[m, 1] := P[1] : A[m, 2] := P[2] :
 od:
 C := matrix(n, 2, (i, j) → A[i, j]) :
 pointplot(C, scaling = constrained, axes = none, color = green);
```