MATH 100 – Introduction to the Profession

Random Affine Transformations and Fractals in MATLAB

Greg Fasshauer

Department of Applied Mathematics
Illinois Institute of Technology

Fall 2012

Linear Transformations

Recall that multiplication by the matrix A, i.e.,

$$\boldsymbol{x} \mapsto \mathsf{A}\boldsymbol{x},$$

represents a linear transformation of the vector $\boldsymbol{x}$.

Linear Transformations

Recall that multiplication by the matrix A, i.e.,

$$\boldsymbol{x} \mapsto A\boldsymbol{x},$$

represents a linear transformation of the vector $\boldsymbol{x}$.

2D linear transformations corresponded to

- scalings
- rotations
- reflections
- shear maps (distorted version of our house)

Linear Transformations

Recall that multiplication by the matrix A, i.e.,

$$\boldsymbol{x} \mapsto \mathsf{A}\boldsymbol{x},$$

represents a linear transformation of the vector $\boldsymbol{x}$.

2D linear transformations corresponded to

- scalings
- rotations
- reflections
- shear maps (distorted version of our house)

In particular, the origin $\boldsymbol{x} = [0\ 0]^T$ is mapped to $\mathsf{A}\boldsymbol{x} = [0\ 0]^T$, so is kept fixed.

Affine Transformations

Now we also allow a possible translation by a vector $\boldsymbol{b}$ in addition to the matrix multiplication, i.e.,

$$\boldsymbol{x} \mapsto \mathsf{A}\boldsymbol{x} + \boldsymbol{b}.$$

This is the general form of an affine transformation of the vector $\boldsymbol{x}$.

Affine Transformations

Now we also allow a possible translation by a vector $\boldsymbol{b}$ in addition to the matrix multiplication, i.e.,

$$\boldsymbol{x} \mapsto A\boldsymbol{x} + \boldsymbol{b}.$$

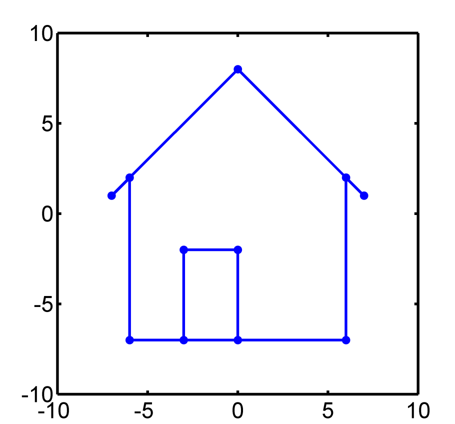
This is the general form of an affine transformation of the vector $\boldsymbol{x}$.

2D affine transformations

- include all linear transformations (when $\boldsymbol{b} = [0\ 0]^T$)
- allow the origin to be moved (translated)

We begin with the house as before:

```
X = house
dot2dot(X)
```



The matrix G $\left(\frac{\pi}{4}\right)$

$$G\left(\frac{\pi}{4}\right) = \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

rotates the house counterclockwise by a 45° angle:

```
G = [sqrt(2)/2 -sqrt(2)/2; sqrt(2)/2 sqrt(2)/2]
dot2dot(G*X)
```

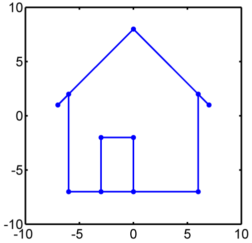
The matrix $G\left(\frac{\pi}{4}\right)$

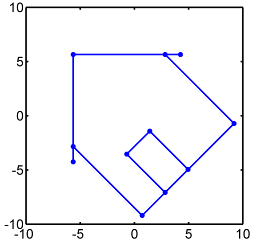
$$G\left(\frac{\pi}{4}\right) = \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

rotates the house counterclockwise by a 45° angle:

```
G = [sqrt(2)/2 -sqrt(2)/2; sqrt(2)/2 sqrt(2)/2]
dot2dot(G*X)
```

This is a linear transformation (as in `wiggle.m`).





To obtain an affine transformation we add a nonzero vector $\boldsymbol{b}$.

However, since in our house example the “vector” $\boldsymbol{x}$ is actually a matrix X (a collection of many points listed in the columns of X), we need to use a translation matrix B consisting of many copies of the (same) translation vector $\boldsymbol{b}$.

To obtain an affine transformation we add a nonzero vector $\boldsymbol{b}$.

However, since in our house example the “vector” $\boldsymbol{x}$ is actually a matrix X (a collection of many points listed in the columns of X), we need to use a translation matrix B consisting of many copies of the (same) translation vector $\boldsymbol{b}$.

This can be done by using MATLAB’s `repmat()` command:

```
G = [sqrt(2)/2 -sqrt(2)/2; sqrt(2)/2 sqrt(2)/2]
b = [1; 2]     % 1 to the right, 2 up
n = size(X,2)  % number of columns/points in X
% make as many copies of b as are needed to match X
B = repmat(b,1,n)
dot2dot(G*X + B)
```

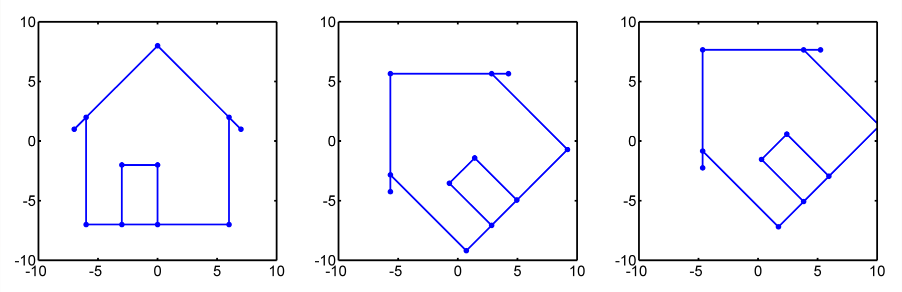


Figure : The original house (left), rotated by $G(\frac{\pi}{4})$ about the origin (middle), and rotated by $G(\frac{\pi}{4})$ about the origin and then translated by $\boldsymbol{b} = [1\ 2]^T$ (right).

Fractal fern

The MATLAB script `fern.m` from [ExM] uses

- three different affine transformations
- and one linear transformation

that are performed randomly with different probabilities to generate a fractal shape that looks like a real-life fern.

Fractal fern

The MATLAB script `fern.m` from [ExM] uses

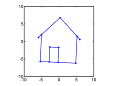
- three different affine transformations
- and one linear transformation

that are performed randomly with different probabilities to generate a fractal shape that looks like a real-life fern.

In particular, `fern.m` uses (plots show effects of A only)

- 85% of the time: a small clockwise rotation and small rescaling with an upward shift:

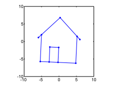
$$\boldsymbol{x} \mapsto A_1\boldsymbol{x} + \boldsymbol{b}_1 = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$



In particular, `fern.m` uses (plots show effects of A only)

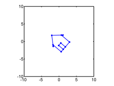
- 85% of the time: a small clockwise rotation and small rescaling with an upward shift:

$$\boldsymbol{x} \mapsto A_1\boldsymbol{x} + \boldsymbol{b}_1 = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$



- 7% of the time: a larger counter-clockwise rotation and larger rescaling with the same upward shift:

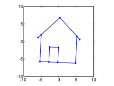
$$\boldsymbol{x} \mapsto A_2\boldsymbol{x} + \boldsymbol{b}_2 = \begin{bmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$



In particular, `fern.m` uses (plots show effects of A only)

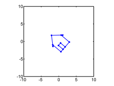
- 85% of the time: a small clockwise rotation and small rescaling with an upward shift:

$$\boldsymbol{x} \mapsto A_1\boldsymbol{x} + \boldsymbol{b}_1 = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$



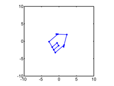
- 7% of the time: a larger counter-clockwise rotation and larger rescaling with the same upward shift:

$$\boldsymbol{x} \mapsto A_2\boldsymbol{x} + \boldsymbol{b}_2 = \begin{bmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$



- 7% of the time: a larger clockwise rotation, rescaling and shear with a smaller upward shift:

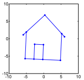
$$\boldsymbol{x} \mapsto A_3\boldsymbol{x} + \boldsymbol{b}_3 = \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 0.44 \end{bmatrix}$$



In particular, `fern.m` uses (plots show effects of A only)

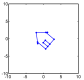
- 85% of the time: a small clockwise rotation and small rescaling with an upward shift:

$$\boldsymbol{x} \mapsto A_1\boldsymbol{x} + \boldsymbol{b}_1 = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$



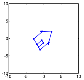
- 7% of the time: a larger counter-clockwise rotation and larger rescaling with the same upward shift:

$$\boldsymbol{x} \mapsto A_2\boldsymbol{x} + \boldsymbol{b}_2 = \begin{bmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$



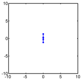
- 7% of the time: a larger clockwise rotation, rescaling and shear with a smaller upward shift:

$$\boldsymbol{x} \mapsto A_3\boldsymbol{x} + \boldsymbol{b}_3 = \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 0.44 \end{bmatrix}$$



- 1% of the time: a projection and rescaling onto the stem:

$$\boldsymbol{x} \mapsto A_4\boldsymbol{x} = \begin{bmatrix} 0 & 0 \\ 0 & 0.16 \end{bmatrix} \boldsymbol{x}$$



Other (mathematically) interesting parts of the MATLAB script `fern.m` are:

- Use of negation to control the loop that keeps adding points (it runs until the “stop” button is pressed, i.e., its value is 1):

```
while ~get(stop,'value')
```

Other (mathematically) interesting parts of the MATLAB script `fern.m` are:

- Use of negation to control the loop that keeps adding points (it runs until the “stop” button is pressed, i.e., its value is 1):

```
while ~get(stop,'value')
```

- Use of a random number generator to generate a random number (the probability of switching between transformations) uniformly distributed in $(0, 1)$:

```
r = rand;
```

Summary scripts

Look at `fern_recap.m` (on the ExM website).

In particular, `finitefern(n,'s')` produces a fern picture in which *n* points are highlighted and added one at a time.

A group at the University of Calgary [Algorithmic Botany] around Przemyslaw Prusinkiewicz has been using so-called L-systems (similar to the system of transformations that generated the fractal fern) to create entire synthetic landscapes:

They have many publications, such as [PalubickiEtAl], from which the above image is taken.

References I

T. A. Driscoll.
Learning MATLAB.
SIAM, Philadelphia, 2009.
http://epubs.siam.org/ebooks/siam/other_titles_in_applied_mathematics/ot115

D. J. Higham and N. J. Higham.
MATLAB Guide (2nd ed.).
SIAM, Philadelphia, 2005.
http://epubs.siam.org/ebooks/siam/other_titles_in_applied_mathematics/ot92

C. Moler.
Numerical Computing with MATLAB.
SIAM, Philadelphia, 2004.
http://www.mathworks.com/moler/index_ncm.html

References II

C. Moler.
Experiments with MATLAB.
Free download at
http://www.mathworks.com/moler/exm/chapters.html

W. Palubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Mech, and P. Prusinkiewicz.
Self-organizing tree models for image synthesis.
ACM Transactions on Graphics 28(3), 58:1–10, 2009. http://algorithmicbotany.org/papers/selforg.sig2009.small.pdf

The MathWorks.
MATLAB 7: Getting Started Guide.
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

Algorithmic Botany.
University of Calgary.
http://algorithmicbotany.org/