MATLAB
for
BEGINNERS

In the High School Classroom

CHINYEN CHUO

Table of Contents

LESSON PLAN 1
INTRODUCTION TO MATLAB

Name: Chinyen Chuo
Content Area: Algebra 2, Precalculus, and Calculus
Grade Level: 9 - 12
Title: Matrix Applications

Next Generation Sunshine State Standards:

- MA.912.D.8.1 - Use matrices to organize and store data. Perform matrix operations (addition, subtraction, scalar multiplication, multiplication).
- MA.912.D.8.2 - Use matrix operations to solve problems.
- MA.912.D.8.3 - Use row-reduction techniques to solve problems.
- MA.912.D.8.4 - Find the inverse of a matrix and use the inverse to solve problems with and without the use of technology.
- MA.912.D.8.5 - Use determinants of 2 x 2 and 3 x 3 matrices as well as higher order matrices with and without the use of technology.
- MA.912.D.9.3 - Use vectors to model and solve application problems.

Common Core State Standard Mathematics:

- MACC.912.N-VM.6.(+) - Use matrices to represent and manipulate data.
- MACC.912.N-VM.7.(+) - Multiply matrices by scalars to produce new matrices.
- MACC.912.N-VM.8.(+) - Add, subtract, and multiply matrices of appropriate dimensions.
- MACC.912.N-VM.9.(+) - Understand that, unlike multiplication of numbers, matrix multiplication for square matrices is not a commutative operation, but still satisfies the associative and distributive properties.
- MACC.912.N-VM.10.(+) - Understand that the zero and identity matrices play a role in matrix addition and multiplication similar to the role of 0 and 1 in the real numbers. The determinant of a square matrix is nonzero if and only if the matrix has a multiplicative inverse.
- MACC.912.N-VM.11.(+) - Multiply a vector (regarded as a matrix with one column) by a matrix of suitable dimensions to produce another vector. Work with matrices as transformations of vectors.
- MACC.912.N-VM.12.(+) - Work with 2 2 matrices as transformations of the plane, and interpret the absolute value of the determinant in terms of area.

Unit: Matrix Applications - Computer Graphics and Cryptography

Learning Goal: Student will be able to understand the fundamental ideas of matrix applications in computer graphics and cryptography.

Objectives:

- Student will be able to understand basic programming design in MATLAB
- Student will be able to perform basic arithmetic comuptations using MATLAB.
- Student will be able to perform matrix operations using MATLAB.
- Student will be able to run small MATLAB programs to obtain results of transformed graphics and explain the underlying fundamental concepts.
- Student will be able to perform matrix operations using MATLAB to en-code and de-code messages, and explain the fundamental ideas of computer graphics using matrix transformations.
- Student will be able to write a mathematics paper explaining concepts, properties, and applications of matrices.

Materials:

- Computer or laptop with internet connection
- Computer software MATLAB, TI calculators, other technology software such as MS Excel, Graphmatica, Graph, etc.
- Tutorial: MATLAB for BEGINNERS (In the High School Classroom) by Chinyen Chuo
- Tutorial 1: Getting Started with MATLAB by Edward Neuman [2]
- Tutorial 2: Getting Started with MATLAB by Edward Neuman [3]

Prior Knowledge:

- Basic graphing skills using graphing calculators or computer programs
- Basic computational skills
- Basic knowledge of matrices and vectors

Procedures: (Time needed - Four 49-minute periods, or ongoing as needed.)

Day 1-2: Use the Tutorial, MATLAB for BEGINNERS - in the High School Classroom, and have students follow directions on the tutorial, type in codes and check for output.

Day 3-4: Let students choose one of the two explorations: Computer Graphics or Crytography for applications. Either have students write a small MATLAB program that will help them to obtain answers and results for the explorations or give them the programs included in the Appendices. Have students trace the code and explain the meanings and purposes of each line, and then modify the codes to suit they needs for new images or their own coded messages.

If time permits, have students write an expository essay using correct mathematical language that shows their understanding and discoveries of the mathematical concepts applied in Computer Graphics and Cryptography.

Evaluation: Tutorial practices and explorations (Computer Graphics and Cryptography).

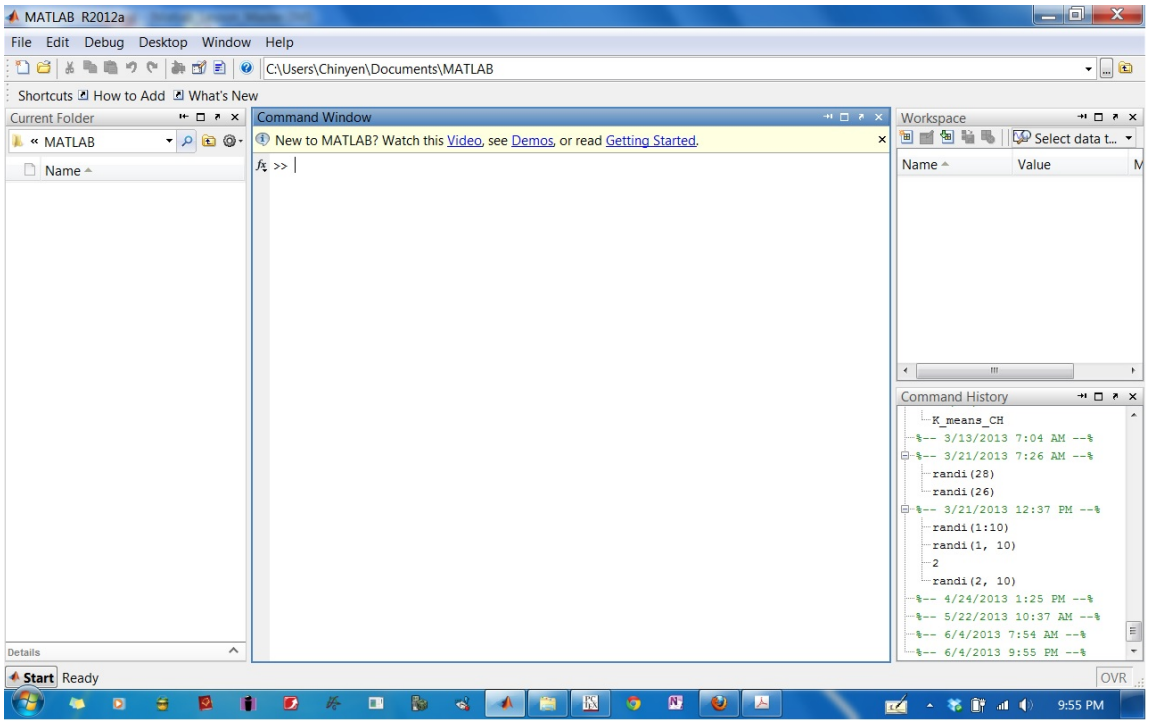
MATLAB
Tutorial & Practice

1 INTRODUCTION

The purpose of this tutorial is to provide a concise material and practice for high school students to gain a basic understanding of MATLAB.

When you open MATLAB, you should see the MATLAB Desktop, and within it, you may see the following windows

- *Command Window* - the main part and you should see the symbol ">>", which is called a *prompt*. Commands are entered here.
- *Current Folder* - where a new file is saved. This also shows the files that are in the current folder.
- *Workspace* - lists all variables in the currently active workspace in alphabetical order.
- *Command History* - shows all commands entered.
- *Details* - shows the details of a file.



1.1 Helpful Commands

- These commands are used in the Command Window.

Command	Desription	Example	Result

`demo`	opens the Help browser and shows some features of MATLAB		
`help`	explains any command	`help exp`	`exp(X) is the exponential of the elements of X, e to the X.`
`help help`	explains how help works		
`lookfor`	searches through the help for a specific word or phrase		
`;`	suppresses output	`>>x = 6*5;`	`>>`
`clear`	removes all variable from workspace		
`clear` *variablename*	clears a specific variable	`clear x`	If 2 was assigned to `x`, now it no longer exists
`clc`	clears command window		
`ctrl+c`	stops a running MATLAB process		
`x=2`	assigns 2 to the variable `x`		
`who`	shows variables that have been defined		
`whos`	shows variables that have been defined in this Command Window		
`format`	sets output format	`format short`	Scaled fixed point format with 5 digits
`rand`	generates random numbers from 0 to 1	`rand`	`ans = 0.8147`

1.2 Arithmetic Operators

Use MATLAB to find answers and fill in the blanks.

MATLAB Operator	Math Operation	Example	Result
`+`	Addition	`3+5`	**ANS:** `ans=8`
`-`	Subtraction	`3-5`	**ANS:** `ans=-2`
`*`	Multiplication	`3*5`	**ANS:** `ans=15`
`/`	Division (Divided by)	`3/5`	**ANS:** `ans=0.6000`
`\`	Division (Divided into)	`3\5`	**ANS:** `ans=1.6667`
`^`	Power	`3^5`	**ANS:** `ans=243`

- MATLAB follows the order of operations:

`( )`	parentheses
`^`	exponentiation
`-`	negation
`*, /, \`	multiplication and division
`+,-`	addition and subtraction

1.3 Constants

Use MATLAB to find answers and fill in the blanks.

MATLAB Command	Description	Example	Output
`pi`	π	`pi+1`	ANS: `4.1416`
`i`	$\sqrt{-1}$	`i+1`	ANS: `1.0000 + 1.0000i`
`j`	$\sqrt{-1}$	`1+j`	ANS: `1.0000 + 1.0000i`
`inf`	∞		ANS:
`-inf`	$-\infty$		ANS:
`NaN`	Not a number	`0/0`	ANS: `ans = NaN`

1.4 Build-in-Functions

Use MATLAB to find answers and fill in the blanks.

MATLAB Command	Description	Example	Output
`help elfun`	shows a list of the elementary math functions	`pi+1`	ANS: `4.1416`
`sin`	`sine` of argument in radians	`sin(pi)`	ANS: `1.2246e-16`
`sind`	`sine` of an argument in degrees	`sind(180)`	ANS: `0`
`asin`	inverse `sine`	`asin(1)`	ANS: `1.5708`
`asind`	inverse `sine` results in degrees	`sind(1)`	ANS: `90`
`exp`	exponential	`exp(1)`	ANS: `2.7183`
`log`	natural logarithm	`log(exp(pi))`	ANS: `3.1416`
`log10`	common logarithm (base 10)	`log10(10)`	ANS: `and = 1`
`log2`	logarithm of base 2	`log2(1/2)`	ANS: `ans = -1`
`sqrt`	square root	`sqrt(4)`	ANS: `ans = 2`
`nthroot(x,n)`	real n-th root of real numbers	`nthroot(8, 3)` $= \sqrt[3]{8}$	ANS: `ans = 2`
`abs`	absolute value	`abs(-4)`	ANS: `ans = 4`
`floor`	round toward negative infinity	`floor(-0.001)`	ANS: `ans = -1`
`ceil`	round toward positive infinity	`ceil(0.01)`	ANS: `ans = 1`
`round`	round toward the nearest integer	`round(3.55)`	ANS: `ans = 4`
`rem`	remainder after division	`rem(12,4)`	ANS: `ans = 3`

1.5 Practice

1. For each of the following, think about what the results would be, and then type them in to MATLAB to verify your answers:

A) `2^4-1` ANS: `15`

B) `2^(4-1)` **ANS:** `8`

C) `4/5` **ANS:** `0.8`

D) `4\5` **ANS:** `1.25`

E) `2*9+6/3` **ANS:** `20`

F) `7--8` **ANS:** `15`

2. What is the result of `rand(10)`? `ran*10`? And `rand*20`?

ANS: `rand(10)` generates a 10 by 10 matrix of real numbers from 0 to 10
`ran*10` generates one real number from 0 to 10
`rand*20` generates one real number from 0 to 20

3. What is the result of the following code?

```
>> low=2;
>> high=6;
>> round(rand*(high-low)+low)
```

ANS: A random integer in the range from 2 to 6

4. Write a code that generates an integer in the range from 0 to 60.

ANS: `round(rand*60)`

5. Write a code to assign 4 to x, suppress the output; divide 3 by $\sqrt{2+x}$; state the output.

ANS: `x=4; 3/sqrt(2+x)`
Output: `x=1.2247`

6. For the following code, describe what each line does and state the output:

```
1. >> x=5;
2. >> y=4;
3. >> z=1;
4. >> y-x;
5. >> w=2/x+5*y*z
```

ANS: `20.4000`

7. Given the code:

```
>> x=5;
>> X=6;
>> x
```

What is the output? Is MATLAB case sensitive?

ANS: `x=5`. Yes, MATLAB is case sensitive.

8. Solve using MATLAB: Assign the values of 2 and 4 to the variables x and y, respectively, then compute the value of $z = 5x + 2y$.

ANS: `z=18`

9. Perform the following and determine the result for the code:

A) Clear all variables **ANS:** `clear`

B) Clear the Command Window **ANS:** `clc`

C)
```
>> %X=5;
>> x=2;
>> X
```
ANS: `Undefined function or variable 'X'`

10. Use the `help` feature to find the command for computing the value of 6!.

ANS: `factorial(6)=720`

11. Create a script file and write a generic code for computing distance between any two points (x_1, y_1) and (x_2, y_2). Then use the code to find the distances between the points:

A) $(3, 4)$ and $(-5, 2)$ **ANS:** `8.2462`

B) $(5, 0)$ and $(-3, -6)$ **ANS:** `10`

ANS:
```
%%% Distance between two points
clc; clear;
x1=input('Please enter a value for x1:');
y1=input('Please enter a value for y1:');
x2=input('Please enter a value for x2:');
y2=input('Please enter a value for y2:');

disp('The distance is:')

  d=sqrt((x1-x2)^2+(y1-y2)^2)
```

12. What is `atan(1)`? What does the output represent?

ANS: `atan(1)`=arctan(1) $= \frac{\pi}{4}$. The result is an angle in radians

13. What is `tan(pi/2)`? What is `tand(pi/2)`?

ANS: `1.6331e+016`. This is equivalent to ∞
`tand(pi/2) = 0.0274`. This is the same as $\tan\left(\frac{\pi}{2}\right)^\circ$

14. What is `log2(0)`? Is the output correct? Why or why not?

ANS: `ans = -inf. This is incorrect since` $\log_2 0$ `is undefined.`

15. What is the output of `exp(1)`? What is the equivalent mathematical expression?

ANS: `ans = 2.7183. This is the same as` e^1

2 MATRICES AND VECTORS

In mathematics, a vector is defined as a quantity that has both magnitude and direction, such as velocity. Matrix is defined as a rectangular array of numbers. The size (order) of a matrix is generally denoted by $r \times c$, that is *row* by *column*. Furthermore, a matrix of size $1 \times n$ can be called a *row vector* and a matrix of $m \times 1$ a column vector. A matrix of one element is called a *scalar* of size 1×1.

Following are examples of matrices:

1. [2] - a scalar
2. [1 4] - a row vector
3. $\begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$ - a column vector
4. $\begin{bmatrix} 1 & 3 & 4 \\ 5 & 6 & 7 \end{bmatrix}$ - a 2×3 matrix

MATLAB is specially designed as a tool for doing numerical computations with matrices and vectors that are used to store sets of values of the same type. MATLAB stands for *Matrix Laboratory*.

2.1 Creating Row and Column Vectors

Note that this is the same as creating a one-dimensional array, or a matrix of size $1 \times n$ or $m \times 1$.

- Three direct ways of creating row vectors:

  1.
```
>> v=[1 2 3  4]

v =
     1     2     3     4
```
  2.
```
>> v=[1,2,3,4]

v =
     1     2     3     4
```
  3.
```
w=1:4

    w =
         1     2     3     4
```

- A column vector:

```
>> u=[1;2;3;4]

u =
     1
     2
     3
     4
```

  This is a 4×1 vector assigned to the variable `u` .

2.2 Creating $m \times n$ Matrices

- A 3×4 matrix:

```
 a=[1 2 3 4; 5 6 7 8; 0 3 2 1]

a =
     1     2     3     4
     5     6     7     8
     0     3     2     1
```

- You may also enter each row on a separate line to keep track of number of entries in a matrix:

```
>> b=[1 2 3 4;
2 4 3 1;
8 9 0 1]

b =
     1     2     3     4
     2     4     3     1
     8     9     0     1
```

  Note that `b` is defined as the matrix above. If you type `b` in the Command Window, you should get a matrix output as the one above. You may also get the variable output from the Workspace.

2.3 Using the Colon Operator and 'linspace' Function

- Creating a vector with all integers from 1 to 8 in steps of 2:

```
>> v2=1:2:8

v2 =
     1     3     5     7
```

- Creating a vector with three values evenly spaced between 1 and 10:

```
>> v3=linspace(1,10,3)

v3 =
    1.0000    5.5000   10.0000
```

2.4 Referring to Elements

- Referring to an entry in a row or column vector:

```
>> v2=1:2:8

v2 =
     1     3     5     7

>> v2(3)

ans =
     5
```

- Referring to an entry in a matrix:

```
>> b=[1 2 3 4;
2 4 3 1;
8 9 0 1]

b =
     1     2     3     4
     2     4     3     1
     8     9     0     1

>> b(3,2)

ans =
     9
```

- Referring to a row or column of elements:

```
>> b=[1:3; 3:5]

b =   %matrix b
     1     2     3
     3     4     5

>> b(1,:) %all elements in the first row

ans =
     1     2     3

>> b(:,2) %all elements in the second column
ans =
     2
     4
```

2.5 Modifying Matrix Elements & Creating New Matrices

- Concatenating two vectors:

```
>> a=zeros(2)

a =
     0     0
     0     0

>> b=ones(2,3)

b =
     1     1     1
     1     1     1

>> newvec=[a,b]
%combining two vectors as one

newvec =

     0     0     1     1     1
     0     0     1     1     1
```

2.6 Dimensions

- Using build-in function 'size':

```
  >> A=[1:3; 4:6]

A =
     1     2     3
     4     5     6

>> A=[1:3; 4:6]' %transpose matrix A

A =
     1     4
     2     5
     3     6

>> size(A)

ans =
     3     2 %size: 3X2

>> [r c]=size(A)

r =
     3  %row = 3

c =
     2  %column = 2
```

- Empty vectors:

```
  >> B=[]

B =
     []

>> size(B)
ans =
     0     0 %size = 0
```

2.7 Matrix Calculations

- Adding / subtracting Matrices: `A+B` or `A-B`

```
>> a=[1 3 5 7; 6 2 -1 4]

a =
     1     3     5     7
     6     2    -1     4

>> b=[3: 6; -9 4 0 2]

b =
     3     4     5     6
    -9     4     0     2

>> c=a+b

c =
     4     7    10    13
    -3     6    -1     6
```

- Multiplying two matrices:

```
>> a=[1 2; 3 4]

a =
     1     2
     3     4

>> b=[3: 6; -9 4 0 2]

b =
     3     4     5     6
    -9     4     0     2

>> c=a*b

c =
   -15    12     5    10
   -27    28    15    26
```

Use your calculator, do the math, and verify the answer.

- Multiplying two matrices element by element:

```
>> a=[1 3 5 7; 6 2 -1 4]

a =
     1     3     5     7
     6     2    -1     4

>> b=[3: 6; -9 4 0 2]
```

```
b =
     3     4     5     6
    -9     4     0     2

>> c=a.*b %(notice the dot after a)

c =
     3    12    25    42
   -54     8     0     8
```

Note: In order for this operation to work, the size of the matrices must be the same.

- Two different divisions of matrices:

```
>> a=[1 2; 3 4]

a =
     1     2
     3     4

>> b=[3 -2; 5 8]

b =
     3    -2
     5     8

>> a/b

ans =
   -0.0588    0.2353
    0.1176    0.5294

>> a./b

ans =
    0.3333   -1.0000
    0.6000    0.5000
```

Note:

  - In MATLAB , `a/b` is equivalent to $a \times b^{-1}$, where b^{-1} is the inverse of `b`
  - `a./b` - This is an *array division* and it divides every element of `a` by elements of corresponding position in `b`

2.8 Summary of Array and Matrix Operations

MATLAB	Operation / Comments
`a/b`	Matrix right division. This is equivalent to `a*inv(b)`
`a\b`	Matrix left division. Matrix division is defined by `inv(a)*b`, where `inv(a)` is the inverse of matrix `a`.
`a.^b`	Array exponentiation. Element-by-element exponentiation of `a` and `b`: $a_{i,j}^{b_{i,j}}$, or `a(i,j)^b(i,j)`. Both arrays must of the same shape, or one of them must be a scalar.
`a./b`	Array right division. Element-by-element division of `a` and `b`: `a(i,j)/b(i,j)`. Both arrays must of the same shape, or one of them must be a scalar.
`a.\b`	Array left division. Element-by-element division of `a` and `b`, but `b` in the numerator `b(i,j)/a(i,j)`. Both arrays must of the same shape, or one of them must be a scalar.

2.9 Practice

1. Consider the following sequence of statements and expressions. First conjecture the output of each statement or expression, and then type them in to MATLAB to verify your answers.

A) `a=[1:4; 2 10 3 5]`

B) `a(1,4)`

C) a(:, 4)

D) `a(4)`

E) `size(a)`

F) `numel(a)`

G) `reshape(a, 1, numel(a))`

H) `vec=a(1,:)`

I) `vec(2)`

J) `vec(3) = []`

K) `vec(5) = 8`

L) `vec = [vec 9]`

2. Use the **linspace** function to create a matrix of five evenly spaced values from 5 to 20. (Hint: Use **help** to find the syntax code).

3. A) Create a matrix `A` of size 1×4 that stores angle measures 10, 15, 70, and 90 in degrees.

B) Create an empty matrix called `radians`.

C) Convert all the entries in `A` to radians measures and store them in the matrix `radian`.

D) Create a 4 matrix call `table` that stores all angle values in degrees from matrix `A` and in radians from matrix `radian` in 2 columns. (Hint: Use transpose)

ANS:

Practice 2.9

1. A) `a=[1:4; 2 10 3 5]` - This generates a 2×4 matrix.

B) `a(1,4)`- This refers to the entry $a_{1,4}$ Hence, output is 4

C) `a(:, 4)` - Output: all entries in the 4th column

D) `a(4)` - Output: 10

E) `size(a)` - Output: `2 4`

F) `numel(a)` - Output: `8`, total number of elements

G) `reshape(a, 1, numel(8))` - Output:`1 2 2 10 3 3 4 5`

H) `vec=a(1,:)` - Assigns first row of entries to variable `vec`; output: `1 2 3 4`

I) `vec(2)` - Output: `2`

J) `vec(3) = []` - This takes away 3rd column in vector `vec`; hence output: `ans = 1 2 4`

K) `vec(5) = 8` - This assigns 8 to the fifth entry in vector `vec`; output: `ans = 1 2 4 0 8`. Since entry 4 did not exist, a 0 is assigned to that entry.

L) `vec = [vec 9]` - This concatenates 9 to the vector `vec`; output: `vec = 1 2 4 0 8 9`

2. `linspace(5,20,5)`
Output:
`ans = 5.0000 8.7500 12.5000 16.2500 20.0000`

3. A)
```
>> A=[10 15 70 90]
```

B)
```
>> radian=[]
```

C)
```
>> radian=A.*pi/180
radian =
0.1745 0.2618 1.2217 1.5708
```

D)
```
>> table=[A', radian']
tale =
10.0000 0.1745
15.0000 0.2618
70.0000 1.2217
90.0000 1.5708
```

Appendices

Appendix A - Exploration #1: Computer Graphics [7]

Computer Graphics I

Matrix algebra is the basic tool used in computer graphics to manipulate images on a computer screen. We will see how matrix multiplication can be used to "move" a point in the plane to a prescribed location. Combining such moves enables us to stretch, compress, rotate, and otherwise transform a figure, as we see in the images below.

Image Compressed Rotated Sheared

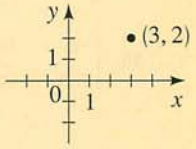
Moving Points in the Plane

Let's represent the point (x, y) in the plane by a 2×1 matrix:

$$(x, y) \leftrightarrow \begin{bmatrix} x \\ y \end{bmatrix}$$

For example, the point $(3, 2)$ in the figure is represented by the matrix

$$P = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

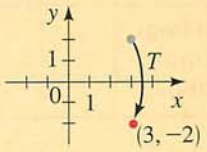


Multiplying by a 2×2 matrix *moves* the point in the plane. For example, if

$$T = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

then multiplying P by T we get

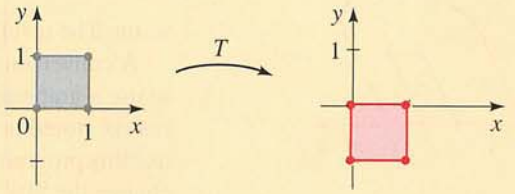
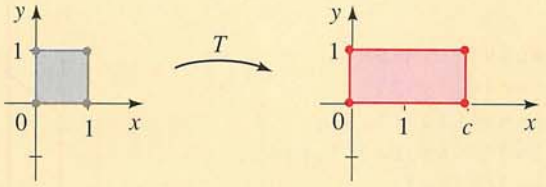
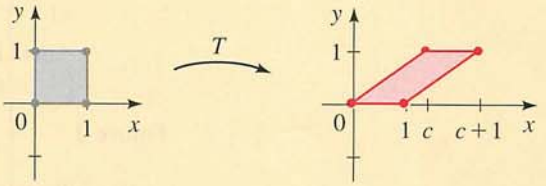
$$TP = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$



We see that the point $(3, 2)$ has been moved to the point $(3, -2)$. In general, multiplication by this matrix T reflects points in the x-axis. If every point in an image is multiplied by this matrix, then the entire image will be flipped upside down about the x-axis. Matrix multiplication "transforms" a point to a new point in the plane. For this reason, a matrix used in this way is called a **transformation**.

Table 1 gives some standard transformations and their effects on the gray square in the first quadrant.

Table 1

Transformation matrix	Effect
$T = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ Reflection in x-axis	
$T = \begin{bmatrix} c & 0 \\ 0 & 1 \end{bmatrix}$ Expansion (or contraction) in the x-direction	
$T = \begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix}$ Shear in x-direction	

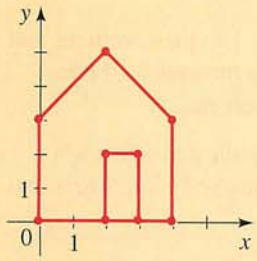


Figure 1

Moving Images in the Plane

Simple line drawings such as the house in Figure 1 consist of a collection of vertex points and connecting line segments. The entire image in Figure 1 can be represented in a computer by the 2×11 **data matrix**

$$D = \begin{bmatrix} 2 & 0 & 0 & 2 & 4 & 4 & 3 & 3 & 2 & 2 & 3 \\ 0 & 0 & 3 & 5 & 3 & 0 & 0 & 2 & 2 & 0 & 0 \end{bmatrix}$$

The columns of D represent the vertex points of the image. To draw the house, we connect successive points (columns) in D by line segments. Now we can transform the whole house by multiplying D by an appropriate transformation matrix. For example, if we apply the shear transformation $T = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$, we get the following matrix.

$$TD = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 2 & 4 & 4 & 3 & 3 & 2 & 2 & 3 \\ 0 & 0 & 3 & 5 & 3 & 0 & 0 & 2 & 2 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 1.5 & 4.5 & 5.5 & 4 & 3 & 4 & 3 & 2 & 3 \\ 0 & 0 & 3 & 5 & 3 & 0 & 0 & 2 & 2 & 0 & 0 \end{bmatrix}$$

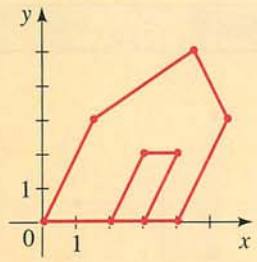
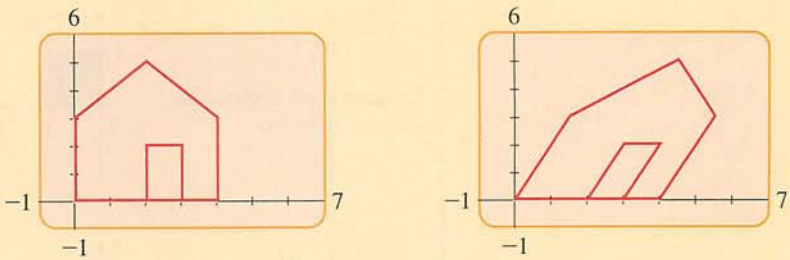


Figure 2

```
PROGRAM:IMAGE
:For(N,1,10)
:Line([A](1,N),
 [A](2,N),[A](1,N+1),
 [A](2,N+1))
:End
```

To draw the image represented by TD, we start with the point $\begin{bmatrix} 2 \\ 0 \end{bmatrix}$, connect it by a line segment to the point $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$, then follow that by a line segment to $\begin{bmatrix} 1.5 \\ 3 \end{bmatrix}$, and so on. The resulting tilted house is shown in Figure 2.

A convenient way to draw an image corresponding to a given data matrix is to use a graphing calculator. The TI-83 program in the margin converts a data matrix stored in `[A]` into the corresponding image, as shown in Figure 3. (To use this program for a data matrix with m columns, store the matrix in `[A]` and change the "10" in the `For` command to $m - 1$.)



(a) House with data matrix D

(b) Tilted house with data matrix TD

Figure 3

We will revisit computer graphics in the *Discovery Project* on page 792, where we will find matrices that rotate an image by any given angle.

1. The gray square in Table 1 has the following vertices.

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Apply each of the three transformations given in Table 1 to these vertices and sketch the result, to verify that each transformation has the indicated effect. Use $c = 2$ in the expansion matrix and $c = 1$ in the shear matrix.

2. Verify that multiplication by the given matrix has the indicated effect when applied to the gray square in the table. Use $c = 3$ in the expansion matrix and $c = 1$ in the shear matrix.

$$T_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad T_2 = \begin{bmatrix} 1 & 0 \\ 0 & c \end{bmatrix} \qquad T_3 = \begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix}$$

Reflection in y-axis — Expansion (or contraction) in y-direction — Shear in y-direction

3. Let $T = \begin{bmatrix} 1 & 1.5 \\ 0 & 1 \end{bmatrix}$.

(a) What effect does T have on the gray square in the Table 1?

(b) Find T^{-1}.

(c) What effect does T^{-1} have on the gray square?

(d) What happens to the square if we first apply T, then T^{-1}?

4. (a) Let $T = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$. What effect does T have on the gray square in Table 1?

(b) Let $S = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$. What effect does S have on the gray square in Table 1?

(c) Apply S to the vertices of the square, and then apply T to the result. What is the effect of the combined transformation?

(d) Find the product matrix $W = TS$.

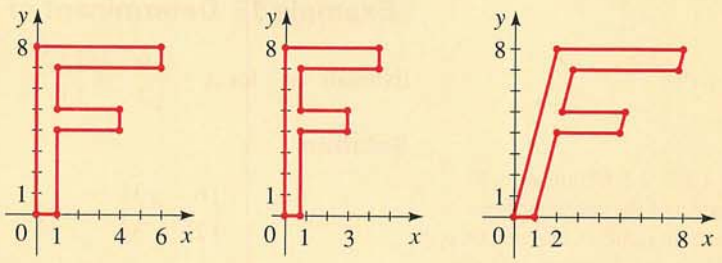
(e) Apply the transformation W to the square. Compare to your final result in part (c). What do you notice?

5. The figure shows three outline versions of the letter **F**. The second one is obtained from the first by shrinking horizontally by a factor of 0.75, and the third is obtained from the first by shearing horizontally by a factor of 0.25.

(a) Find a data matrix D for the first letter **F**.

(b) Find the transformation matrix T that transforms the first **F** into the second. Calculate TD and verify that this is a data matrix for the second **F**.

(c) Find the transformation matrix S that transforms the first **F** into the third. Calculate SD and verify that this is a data matrix for the third **F**.



6. Here is a data matrix for a line drawing.

$$D = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 4 & 4 & 0 \end{bmatrix}$$

(a) Draw the image represented by D.

(b) Let $T = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}$. Calculate the matrix product TD and draw the image represented by this product. What is the effect of the transformation T?

(c) Express T as a product of a shear matrix and a reflection matrix. (See problem 2.)

Appendix B - Exploration #2: Matrices and Cryptography [8]

EXTEND

6-3 Graphing Technology Lab
Matrices and Cryptography

Objective
Use a graphing calculator and matrices to encode and decode messages.

Cryptography is the study of coded messages. Matrices can be used to code messages so that they can only be read after being deciphered by a *key*.

The first step is to decide on a key that can be used to encode the matrix. The key must be an $n \times n$ invertible matrix. The next step is to convert the message to numbers and write it as a matrix. Each letter of the alphabet is represented by a number. The number 0 is used to represent a blank space.

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Finally, the message is encoded by multiplying it by the key.

ACTIVITY 1 Encode a Message

Use $\begin{bmatrix} 1 & -2 \\ -1 & 3 \end{bmatrix}$ to encode the message SATURDAY AT NOON.

Step 1 Convert the message to numbers and write it as a matrix.

S	A	T	U	R	D	A	Y	_	A	T	_	N	O	O	N
19	1	20	21	18	4	1	25	0	1	20	0	14	15	15	14

The key is a 2×2 matrix. To make the matrix multiplication possible, write the message as an 8×2 matrix.

$$\begin{bmatrix} 19 & 1 \\ 20 & 21 \\ 18 & 4 \\ 1 & 25 \\ 0 & 1 \\ 20 & 0 \\ 14 & 15 \\ 15 & 14 \end{bmatrix}$$

StudyTip
Conversion Add zeros to the end of a message if additional entries are needed to fill a matrix.

Step 2 Multiply the converted message by the key using a graphing calculator.

```
[[18   -35]
 [-1   23 ]
 [14   -24]
 [-24  73 ]
 [-1   3  ]
 [20   -40]
 [-1   17 ]↓
```

Step 3 Remove the matrix notation to reveal the encoded message.

18 −35 −1 23 14 −24 −24 73 −1 3 20 −40 −1 17 1 12

Exercises

Use $\begin{bmatrix} 3 & 5 \\ -2 & -3 \end{bmatrix}$ to encode each message.

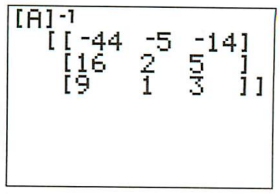
1. CALL ME
2. SEE YOU LATER
3. ORDER PIZZA
4. **CHALLENGE** Use $\begin{bmatrix} 2 & -4 & 3 \\ -1 & 5 & 1 \\ 6 & -2 & 4 \end{bmatrix}$ to encode the message MEET ME AT FIVE.

To decode a message, the inverse of the key must be found. The coded message is then written in matrix form to make the multiplication possible. For instance, if the key is an $n \times n$ matrix, the message is written as a $k \times n$ matrix, where k is the number of rows necessary to include each number in the matrix. If there are not enough characters to fill a row, insert "0"s as spaces. Finally, the coded matrix is multiplied by the inverse of the key.

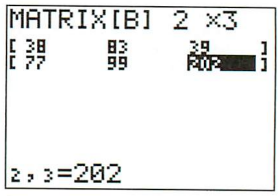
ACTIVITY 2 Decode a Message

Use the inverse of $\begin{bmatrix} -1 & -1 & -3 \\ 3 & 6 & 4 \\ 2 & 1 & 8 \end{bmatrix}$ to decode the message 38 83 39 77 99 202.

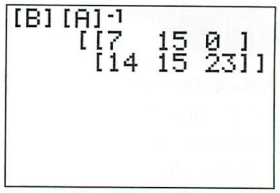
Step 1 Use a graphing calculator to find the inverse of the key.



Step 2 Write the coded message as a matrix. The coded matrix will have 3 columns because the key is a 3 × 3 matrix. It is a 2 × 3 matrix because there are enough numbers to fill two rows. Enter it into your graphing calculator.



Step 3 Use a graphing calculator to multiply the coded matrix by the inverse of the key.



Step 4 Remove the matrix notation and convert the numbers to letters.

7	15	0	14	15	23
G	O	_	N	O	W

Exercises

Use the inverse of $\begin{bmatrix} 12 & -7 \\ -5 & 3 \end{bmatrix}$ to decode each message.

5. 128 −73 232 −135 300 −175 99 −56 83 −48 180 −104 300 −175

6. −27 17 38 −21 84 −49 21 −11 131 −76 201 −116 161 −93

7. 151 −88 150 −86 93 −54 −35 22 −5 3 191 −111 −30 18 182 −105

8. 102 −58 45 −26 −48 29 −69 42 39 −21 228 −133 141 −81 −19 12 228 −133

9. CHALLENGE Use the inverse of $\begin{bmatrix} 2 & 4 & 6 & 0 \\ 1 & 8 & -4 & -6 \\ 7 & 6 & -5 & 3 \\ 1 & 7 & 9 & 2 \end{bmatrix}$ to decode

126 265 −49 −34 198 347 193 96 174 239 49 72 177 286 −61 −27 48 200 70 −76 122 162 −21 35 81 190 −37 −63 130 331 214 17 67 267 94 −25 93 161 120 25.

Appendix C - MATLAB source file: `dot2dot.m`

```
%%%  Exploration #1: Computer Graphics
%%%  Filename: dot2dot.m
%%%  This program plots and connects points to form an image
%%%
%x=rand(10,1)
%y=repmat(var(x),length(x),1)
x=[-6 -6 -7 0 7 6 6 -3 -3 0 0 -6];
y=[-7 2 1 8 1 2 -7 -7 -2 -2 -7 -7];
%plot(x,y,'Marker','o')
plot(x,y,'Marker','s')
%plot(x,y,'Marker','diamond')
A=[x;y]
```

Appendix D - MATLAB source file: `dot2dot2.m`

```
%%%  Exploration #1: Computer Graphics
%%%  Transforms a graphic
%%%  Filename: dot2dot2.m
%%%
%x=rand(10,1)
%y=repmat(var(x),length(x),1)
x=[-6 -6 -7 0 7 6 6 -3 -3 0 0 -6];
y=[-7 2 1 8 1 2 -7 -7 -2 -2 -7 -7];
%plot(x,y,'Marker','o')
plot(x,y,'Marker','s')
%plot(x,y,'Marker','diamond')
A=[x;y]
B=[1 0; 1 -1]
C=B*A
d=C(1,:)
e=C(2,:)
plot(d,e,'Marker','s')
```

Appendix E - MATLAB source file: `encode.m`

```
%%%  Encode & decode messages
%%%
%%%  B='ABCDEFGHIJKLMNOPQRSTUVWXYZ';
message = 'MEET YOU SATURDAY AT NOON';
n = length(message)
if rem(n, 2) ~= 0
    n = n+1
    message = strcat(message, '.')
    length(message)
end
message_v = double(message)
message_m = reshape(message_v, n/2, 2);
key = [1 -2 ; 3 -4]
inv(key)
encoded = message_m * key;
encoded_v = reshape (encoded, 1, n)
code = encoded_v;
n = length(code)
if rem(n,2) ~= 0
    n = n+1
    code = [code, 0]
end
code_m = reshape (code, n/2, 2);
decoded = code_m * inv(key);
decoded_v = int8(reshape (decoded, 1, n));
char(decoded_v)
```

References

[1] *MATLAB Primer, R2013a*. MathWorks Product Documentation.
`http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf`

[2] Neuman, Edward. *Tutorial 1: Getting Started with MATLAB*.
`http://www.math.siu.edu/matlab/tutorial1.pdf`

[3] Neuman, Edward. *Tutorial 1: Getting Started with MATLAB*.
`http://www.math.siu.edu/matlab/tutorial2.pdf`

[4] Arnold, David. *Plotting in MATLAB*. College of the Redwoods.
`http://msemac.redwoods.edu/darnold/math50c/matlab/plotting/index.php`

[5] Moore, Holly. *MATLAB for Engineers*. 2nd ed. New York: Prentice Hall, 2009

[6] Attaway, Storm, and Butterworth-Heinemann. *MATLAB: A Practical Introduction to Programming and Problem solving*.

[7] Stewart, James, and L. Redlin. *Precalculus: Mathematics for Calculus.* 5th ed. Australia: Brooks/Cole, 2010. p.700-703.

[8] *Precalculus*. Columbus, OH: Glencoe/McGraw-Hill, 2011. p.395-396.