Applied Engineering Problem Solving

Lesson #2: Introduction to Linear Algebra & Array and Matrix Operations in Matlab

Prof. John R. White

Chemical and Nuclear Engineering

UMass-Lowell, Lowell MA

Lesson #2 Goals

Linear Algebra Concepts and 2-D Function Evaluation and Plotting

Creating and working with arrays and array indexing

Element-by-element operation versus formal matrix multiplication (and other operations…)

Linear algebra notation and analytical manipulations

Some special matrices

Evaluating and plotting functions of two variables

Some visualization options…

Gilat:
Chapters 2 - 3 & 10

Chapra:
Chapters 2, 8, 11.1 & 13.1

Lesson # 2 Lecture Notes and Illustrative Examples

Array Notation and Indexing

Consider the 2×4 matrix (2 rows and 4 columns):

$$A = \begin{bmatrix} 0 & 1 & 2 & 4 \\ 5 & -1 & 3 & 1 \end{bmatrix}$$

We refer to the element in row i and column j as a_{ij}

a = A(2,3) -- refers to scalar value in row 2, column 3 (a = 3)

b = A(1,:) -- assigns all elements in row 1 to variable b

c = A(:,3) -- assigns 3rd column of A to variable c

What do the following commands give:

d = A(:,[1 4]) and e = A(1,1:3:4)

The colon operator by itself does it "all"

Let's do it in Matlab…

Storage of Information in a 2-D Array

For example, let G be a matrix for storing homework grades:

G(i,j) = grade for student i on HW j

Now set the grades between 60 and 100, where the rand command generates uniformly distributed random numbers between 0 and 1:

```
G = (100 - 60)*rand(25,7) + 60;
```

Grades for student #5: `GS5 = G(5,:)`

Number of rows and columns: `[NS,NG] = size(G)`

Average HW grade for student #5: `S5ave = sum(GS5)/NG`

What does the following give? `sum(G(:,5))/NS` ???

Mathematical Form

$$S5_{ave} = \frac{1}{ng}\sum_{j=1}^{ng} g_{5j}$$

Let's do it in Matlab…

Element by Element Operations

This type of arithmetic is conceptually simple -- one simply performs the desired operation element-by-element for every term of the array.

The array sizes must be identical

Consider two arrays: $A = [a_{ij}]$ and $B = [b_{ij}]$

Addition: C = A + B $c_{ij} = a_{ij} + b_{ij}$

Subtraction: C = A - B $c_{ij} = a_{ij} - b_{ij}$

Multiplication: C = A.*B $c_{ij} = a_{ij} * b_{ij}$

Division: C = A./B $c_{ij} = a_{ij} / b_{ij}$

Exponentiation: C = A.^n $c_{ij} = a_{ij}^n$ (where n is a scalar)

notice the dots!!!

Using the following two arrays:

```
A = [0 1 2; 3 4 5]
B = [3 4 5; 0 1 2]
```

Let's do it in Matlab…

Matrix Operations

Matrix addition and subtraction are identical to the element-by-element operations.

Addition & Subtraction: C = A ± B $c_{ij} = a_{ij} \pm b_{ij}$

However, matrix multiplication is a completely different story!!!

To introduce this concept, consider the following system of linear equations:

$$\begin{aligned} 3x_1 - 2x_2 + 2x_3 &= 1 \\ x_1 + 2x_2 - 3x_3 &= 0 \\ 4x_1 + x_2 + 2x_3 &= 0 \end{aligned}$$

Think of each equation as a row, with the coefficients of the three unknowns, x_1, x_2, and x_3, properly ordered into the corresponding columns or terms of each equation.

Matrix Operations (cont.)

For a general 3x3 system of equations, we have

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

This generic system is used a lot!!!

where a_{ij} is the coefficient of x_j in the i^{th} equation, and b_i is the value on the RHS of equation i.

Using matrix-vector notation, we have **Ax = b** , where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Definition:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{matrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{matrix}$$

Matrix Operations (cont.)

Definition:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{matrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{matrix}$$

The correspondence here says it all…

For example, for the 1st equation in the set, we see that

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

can be written as

$$b_1 = \sum_{j=1}^{3} a_{1j}x_j$$

and, for any element of b, say for example, b_i, which corresponds to row i of the system of equations, we have

$$b_i = \sum_{j=1}^{3} a_{ij}x_j$$

$$\boxed{Ax = b \quad \Rightarrow \quad b_i = \sum_j a_{ij}x_j}$$

Matrix Operations (cont.)

$$\boxed{Ax = b \quad \Rightarrow \quad b_i = \sum_j a_{ij} x_j}$$

This is referred to as the row view of matrix-vector multiplication.

The inner product of row i of matrix A with vector x gives the scalar b_i.

Recall that the inner product of two vectors, y and z, gives a scalar, α, or

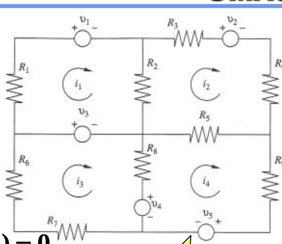
$$\alpha = y \cdot z = y^T z = \begin{bmatrix} y_1 & y_2 & \cdots \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \end{bmatrix} = \sum_i y_i z_i$$

Thus, matrix-vector multiplication is simply a sequence of inner product operations -- one for each row of the system of equations.

Sample Application

Resistive Networks

Kirchoff's voltage law states: the algebraic sum of the voltage drops around a closed loop must be zero



Loop 1: $R_1 i_1 + v_1 + R_2(i_1 - i_2) - v_3 = 0$

Loop 2: $R_2(i_2 - i_1) + R_3 i_2 + v_2 + R_4 i_2 + R_5(i_2 - i_4) = 0$

Loop 3: $R_6 i_3 + v_3 + R_8(i_3 - i_4) + v_4 + R_7 i_3 = 0$

Loop 4: $-v_4 + R_8(i_4 - i_3) + R_5(i_4 - i_2) + R_9 i_4 + v_5 = 0$

Ohm's Law
v = Ri

This can be written as Ax = b, where

$$\begin{bmatrix} R_1 + R_2 & -R_2 & 0 & 0 \\ -R_2 & (R_2 + R_3 + R_4 + R_5) & 0 & -R_5 \\ 0 & 0 & R_6 + R_7 + R_8 & -R_8 \\ 0 & -R_5 & -R_8 & R_5 + R_8 + R_9 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} v_3 - v_1 \\ -v_2 \\ -v_3 - v_4 \\ v_4 - v_5 \end{bmatrix}$$

More Matrix Multiplication

Extending our view of matrix-vector multiplication to matrix-matrix multiplication is quite straightforward.

Consider the system, Ax = b, for two different RHS vectors, or

$$Ax_1 = b_1 \qquad \text{and} \qquad Ax_2 = b_2$$

Now, with our view that a matrix is simply a convenient form for storage of information, let's store the two solution vectors x_1 and x_2 in a matrix, or

$$X = \begin{bmatrix} x_1 & x_2 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

2nd element of the 1st x vector

Writing the two RHS vectors in a similar way, we can form a matrix equation:

$$AX = B \quad \Rightarrow \quad \begin{bmatrix} a_{11} & a_{12} & a_3 \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

More Matrix Multiplication (cont.)

In general, if the size of A is n × p and X is of order p × m, then the resultant matrix, B, is of order n × m -- where we note that, for valid matrix multiplication, the "inner matrix dimensions must agree".

The number of columns of A must be equal to the number of rows of X for the operation, AX, to be valid, or (n × p)(p × m) = n × m

This also implies that, in general, $AX \neq XA$ -- that is, the order of operation is absolutely essential!!!

In the above example, AX = B, we have (3×3)(3×2) = (3×2), so the result will be a 3×2 matrix!

However, note that the operation XA is not even defined since (3×2)(3×3) has inner matrix dimensions that do not match…

More Matrix Multiplication (cont.)

Following the row view of matrix multiplication, we can write the discrete form of AX = B as

$$b_{ij} = \sum_k a_{ik} x_{kj}$$

where the individual element, b_{ij}, of the resultant matrix, B, is given as the inner product of row i of matrix A and column j of matrix X -- where clearly the number of elements (columns) in each row of A must be equal to the number of values (rows) in each column of X (i.e. the "inner matrix dimensions must agree").

In summary, you should view the value of b_{ij} simply as "row i of A into column j of X". With this view, the precise definition of matrix-matrix multiplication is given by

$$\boxed{AX = B \qquad \Rightarrow \qquad b_{ij} = \sum_k a_{ik} x_{kj}}$$

Matrix Multiplication (in detail)

Let's do some examples by hand…

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & 0 \\ 2 & 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & -1 & 2 \\ 1 & -2 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 3 & -1 & 4 \\ 3 & 2 & -2 & 6 \end{bmatrix}$$

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

$$\begin{bmatrix} 1 & 2 & -1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 6 \end{bmatrix}$$

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

$$\begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}\begin{bmatrix} 1 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 3 & 6 & -3 \\ 1 & 2 & -1 \\ 2 & 4 & -2 \end{bmatrix}$$

Transpose Operator and Some Special Matrices

Matrix Transpose: $B = A^T \rightarrow b_{ij} = a_{ji}$ (interchange rows & columns)

Matlab: B = A′

Identity Matrix: $I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (square matrix with unity along main diagonal)

Matlab: I = eye(3)

zeros command: zeros(2,3) = $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

ones command: ones(4,2) = $\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$

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

A = [zeros(3,1),eye(3);2*ones(1,4)]

diag command: diag([1 2 -1 3]) = $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$

A Note about "Matrix Division"

Our illustrations of linear algebra operations thus far have not included matrix division. This is because there is no such thing as formal matrix division!

Instead, one defines an inverse matrix and does matrix multiplication with the inverse matrix.

For example, for an equation containing scalar variables, say ax = b, we can write x as

$$x = \frac{b}{a} = \left(\frac{1}{a}\right) b = a^{-1} b$$

where each of these forms are equivalent.

However, for a matrix equation, Ax = b, the only valid way to formally write the solution vector x is given by

$$x = A^{-1} b \quad \text{where} \quad A^{-1} A = I$$

Definition of A^{-1}

There is no division involved here!!!

A Note about “Matrix Division” (cont.)

This is a confusing point because Matlab, and many of the texts that describe various Matlab matrix operations, routinely refer to matrix division.

This is a point that we will discuss at some length in Lesson #6.

For now, however, if you need to solve a system of linear equations, you want to use Matlab’s backslash, \, operator.

That is, to solve Ax = b, use x = A\b in Matlab.

This is a BIGGIE!

One could also compute and use A^{-1}, that is, $x = A^{-1}b$, but this is much less efficient than the backslash operator.

To see this point and many other insights related to linear equations, we need to briefly introduce some additional concepts, terminology, and techniques from **Linear Algebra**!

Linear Algebra: Notation and Calculation

Need to discuss:

- The matrix inverse, determinants, adjoints, cofactors, minors, rank of a matrix, etc.
- Elementary row operations
- The uniqueness and existence of solutions for both homogeneous and inhomogeneous equations
- The classical eigenvalue problem and how to compute the eigenvalues and eigenvectors of a matrix

… We will do all this via hand manipulations …

So let’s do it…

So let’s do it…

The Maxwellian Distribution Revisited

(from Lesson #1 "1-D Evaluation and Plotting")

In a **dilute gas**, the kinetic energies of the molecules are distributed according to the Maxwellian distribution function:

> **f(E,T) dE = probability of finding a particle in energy interval dE for a particular gas temperature T**

Since f(E,T) is a probability density function (i.e. probability per unit energy), then

$$\int_0^\infty f(E,T)\,dE = 1$$

See maxwell_1.pdf

At a particular gas temperature,

$$f(E,T) = \frac{2\pi}{(\pi kT)^{3/2}} E^{1/2} e^{-E/kT}$$

> **Our job is to evaluate, plot, and interpret this function**

where E is in eV, T is the absolute temperature in K, and the Boltzmann constant, k, has a value of $k = 8.6170 \times 10^{-5}$ eV/K

The Maxwellian Distribution Revisited

(from Lesson #1 "1-D Evaluation and Plotting")

Solution Algorithm:

1. define k and the absolute gas temperature, T
2. define a discretized energy grid, E_i
3. evaluate f(E,T) using element-by-element vector arithmetic
4. plot and label f(E,T) as appropriate and interpret…

> **Let's do this in Matlab (see maxwell_1.m) …**

Some specific things to look for when using maxwell_1.m:

1. basic distribution of f(E) for a given T (should match expectations)
2. use of logspace and semilogx commands
3. proper use of "dot" arithmetic
4. use of the num2str, input, and axis commands

The Maxwellian Distribution Revisited

Now, for this Lesson, we want to focus on the 2-D nature of this function, f(E,T)

Solution Algorithm:

1. define k and a vector of absolute gas temperatures, T_j
2. define a discretized energy grid, E_i
3. evaluate $f(E,T) \rightarrow f(E_i,T_j) \rightarrow f_{ij}$
4. plot and label f(E,T) in various ways, highlighting both quantitative and qualitative visualization techniques

Steps 1- 3, the computational steps, can be done three different ways using scalar, vector, or matrix arithmetic

Let's work on this first

The Scalar Approach

Evaluate and Plot:

$$f(E,T) = c(T)E^{\frac{1}{2}}e^{-\frac{E}{KT}} \quad \text{for} \quad 0 < E < 0.25 \quad \& \quad \text{several T values}$$

Assume that k, E and T are defined with proper units. Then

```
NE = length(E);   NT = length(T);
f = zeros(NE,NT);
for j = 1:NT
  c = 2*pi/(pi*k*T(j))^1.5;
  for i = 1:NE
    f(i,j) = c*sqrt(E(i))*exp(-E(i)/(k*T(j)));
  end
end
```

NO dots please!!!
There should never be any dots with scalar arithmetic!!!

Note the two nested for … end loops for the scalar approach!!!

The **Vector** Approach

Evaluate and Plot:

$$f(E,T) = c(T)E^{\frac{1}{2}}e^{-\frac{E}{KT}} \quad \text{for } 0 < E < 0.25 \quad \& \quad \text{several T values}$$

Assume that k, E and T are defined with proper units. Then

```
NE = length(E);  NT = length(T);
f = zeros(NE,NT);
c = 2*pi./(pi*k*T).^1.5;
for j = 1:NT
  f(:,j) = c(j)*sqrt(E).*exp(-E/(k*T(j)));
end
```

Now we need dot arithmetic!!! Also note that E should be a column vector with this code…

Only one for … end loop is needed for the vector approach!!!

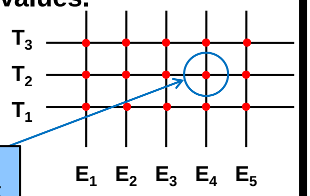
The **Matrix** Approach

Evaluate and Plot:

$$f(E,T) = c(T)E^{\frac{1}{2}}e^{-\frac{E}{KT}} \quad \text{for } 0 < E < 0.25 \quad \& \quad \text{several T values}$$

For the moment, let's assume that E has 5 values and that T has 3 values. This means that f will be a 3×5 array (15 values) evaluated on a grid with 5 E values and 3 T values.

```
E = [1 3 5 7 9]; T = [300 400 500];
[EE,TT] = meshgrid(E,T)
EE =
     1     3     5     7     9
     1     3     5     7     9
     1     3     5     7     9
TT =
   300   400   500   300   300
```



f_{24} is evaluated at T_2 and E_4 or TT_{24} and EE_{24}

Note: values of E given here are for illustrative purposes only -- not within range specified…

The **Matrix** Approach (cont.)

Evaluate and Plot:

$$f(E,T) = c(T)E^{\frac{1}{2}} e^{-\frac{E}{KT}} \quad \text{for } 0 < E < 0.25 \quad \& \quad \text{several } T \text{ values}$$

Assume that k, E and T are defined with proper units. Then

```
[EE,TT] = meshgrid(E,T);
kT = k*TT; c = 2*pi./(pi*kT).^1.5;
F = c.*sqrt(EE).*exp(-EE./kT);
```

Everything is now a 2-D array -- all with the same size…

Now, NO loops are required!!!
Pretty nice!!!

Note all the dots…

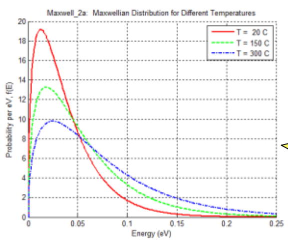
Note all the dots…

CHEN.3170 Applied Engineering Problem Solving
Lesson #2 Linear Algebra + Array and Matrix Operations in Matlab (Sept. 2017)

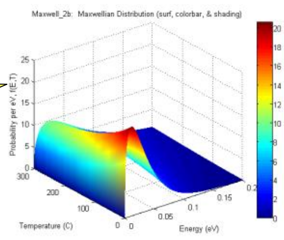
The Maxwellian Distribution Revisited

Okay, now that we have the fundamentals of how to do the evaluations, let's put some of these ideas to work in Matlab and address how to plot functions of two variables…

Let's do this in Matlab
(see maxwell_2a and maxwell_2b.m)



Lots to discuss here, so let's get to it!!!



CHEN.3170 Applied Engineering Problem Solving
Lesson #2 Linear Algebra + Array and Matrix Operations in Matlab (Sept. 2017)

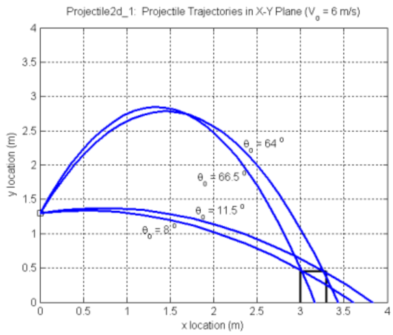
More Illustrative Examples

Sand Pit Utilization

Example that illustrates how to use array indexing and various matrix operations (matrix multiplication and the transpose operation) to assist in the analysis of data stored in a couple of 2-D arrays.

2-D Projectile Motion: A Bottle Cap Tossing Simulation

This example is a good introduction to our next subject , "Programming in Matlab", where we will expand upon some of the programming techniques used here…



Lesson #2 Summary

In this Lesson we have discussed the following topics:

Linear Algebra Concepts &
2-D Function Evaluation and Plotting in Matlab

Creating and working with arrays and array indexing

Element-by-element operation versus formal matrix multiplication (and other operations…)

Linear algebra notation and analytical manipulations

Some special matrices

Evaluating and plotting functions of two variables

Some visualization options…

You should now be much more comfortable with these topics…