

A Review of Linear Algebra

Gerald Recktenwald
Portland State University
Mechanical Engineering Department
gerry@me.pdx.edu

These slides are a supplement to the book *Numerical Methods with MATLAB: Implementations and Applications*, by Gerald W. Recktenwald, © 2003, Prentice-Hall, Upper Saddle River, NJ. These slides are © 2003 Gerald W. Recktenwald. The PDF version of these slides may be downloaded or stored or printed only for noncommercial, educational use. The repackaging or sale of these slides in any form, without written consent of the author, is prohibited.

The latest version of this PDF file, along with other supplemental material for the book, can be found at www.prenhall.com/recktenwald.

Version 0.987 November 14, 2003

Primary Topics

- Vectors
- Matrices
- Mathematical Properties of Vectors and Matrices
- Special Matrices

Notation

Variable type	Typographical Convention	Example
scalar	lower case Greek	σ, α, β
vector	lower case Roman	u, v, x, y, b
matrix	upper case Roman	A, B, C

Defining Vectors in MATLAB

- Assign any expression that evaluates to a vector

```
>> v = [1 3 5 7]
>> w = [2; 4; 6; 8]
>> x = linspace(0,10,5);
>> y = 0:30:180
>> z = sin(y*pi/180);
```

- Distinguish between row and column vectors

```
>> r = [1 2 3];      % row vector
>> s = [1 2 3]';    % column vector
>> r - s
??? Error using ==> -
Matrix dimensions must agree.
```

Although `r` and `s` have the same elements, they are not the same vector. Furthermore, operations involving `r` and `s` are bound by the rules of linear algebra.

Vector Operations

- Addition and Subtraction
- Multiplication by a scalar
- Transpose
- Linear Combinations of Vectors
- Inner Product
- Outer Product
- Vector Norms

Vector Addition and Subtraction

Addition and subtraction are element-by-element operations

$$c = a + b \iff c_i = a_i + b_i \quad i = 1, \dots, n$$

$$d = a - b \iff d_i = a_i - b_i \quad i = 1, \dots, n$$

Example:

$$a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad b = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

$$a + b = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} \quad a - b = \begin{bmatrix} -2 \\ 0 \\ 2 \end{bmatrix}$$

Multiplication by a Scalar

Multiplication by a scalar involves multiplying each element in the vector by the scalar:

$$b = \sigma a \iff b_i = \sigma a_i \quad i = 1, \dots, n$$

Example:

$$a = \begin{bmatrix} 4 \\ 6 \\ 8 \end{bmatrix} \quad b = \frac{a}{2} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

Vector Transpose

The *transpose* of a row vector is a column vector:

$$u = [1, 2, 3] \quad \text{then} \quad u^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Likewise if v is the column vector

$$v = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \quad \text{then} \quad v^T = [4, 5, 6]$$

Linear Combinations (1)

Combine scalar multiplication with addition

$$\alpha \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} + \beta \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} \alpha u_1 + \beta v_1 \\ \alpha u_2 + \beta v_2 \\ \vdots \\ \alpha u_m + \beta v_m \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

Example:

$$r = \begin{bmatrix} -2 \\ 1 \\ 3 \end{bmatrix} \quad s = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

$$t = 2r + 3s = \begin{bmatrix} -4 \\ 2 \\ 6 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \\ 9 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 15 \end{bmatrix}$$

Linear Combinations (2)

Any one vector can be created from an infinite combination of other “suitable” vectors.

Examples:

$$w = \begin{bmatrix} 4 \\ 2 \end{bmatrix} = 4 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

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

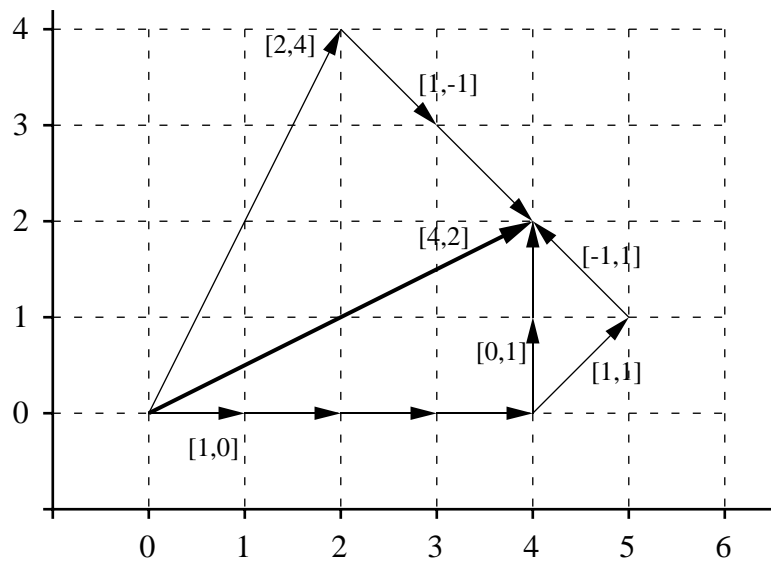
$$w = \begin{bmatrix} 2 \\ 4 \end{bmatrix} - 2 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$w = 2 \begin{bmatrix} 4 \\ 2 \end{bmatrix} - 4 \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Linear Combinations (3)

Graphical interpretation:

- Vector tails can be moved to convenient locations
- Magnitude and direction of vectors is preserved



Vector Inner Product (1)

In physics, analytical geometry, and engineering, the **dot product** has a geometric interpretation

$$\sigma = x \cdot y \iff \sigma = \sum_{i=1}^n x_i y_i$$

$$x \cdot y = \|x\|_2 \|y\|_2 \cos \theta$$

Vector Inner Product (2)

The rules of linear algebra impose compatibility requirements on the inner product.

The inner product of x and y requires that x be a row vector y be a column vector

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4$$

Vector Inner Product (3)

For two n -element *column* vectors, u and v , the inner product is

$$\sigma = u^T v \iff \sigma = \sum_{i=1}^n u_i v_i$$

The inner product is commutative so that
(for two column vectors)

$$u^T v = v^T u$$

Computing the Inner Product in MATLAB

The `*` operator performs the inner product if two vectors are compatible.

```
>> u = (0:3)';          % u and v are
>> v = (3:-1:0)';      % column vectors
>> s = u*v
??? Error using ==> *
Inner matrix dimensions must agree.
```

```
>> s = u'*v
s =
    4
```

```
>> t = v'*u
t =
    4
```

Vector Outer Product

The inner product results in a scalar.

The *outer product* creates a rank-one matrix:

$$A = uv^T \iff a_{i,j} = u_i v_j$$

Example: *Outer product of two 4-element column vectors*

$$\begin{aligned} uv^T &= \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix} \\ &= \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 & u_1 v_4 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 & u_2 v_4 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 & u_3 v_4 \\ u_4 v_1 & u_4 v_2 & u_4 v_3 & u_4 v_4 \end{bmatrix} \end{aligned}$$

Computing the Outer Product in MATLAB

The `*` operator performs the outer product if two vectors are compatible.

```
u = (0:4)';  
v = (4:-1:0)';  
A = u*v'  
A =  
    0    0    0    0    0  
    4    3    2    1    0  
    8    6    4    2    0  
   12    9    6    3    0  
   16   12    8    4    0
```

Vector Norms (1)

Compare magnitude of scalars with the *absolute value*

$$|\alpha| > |\beta|$$

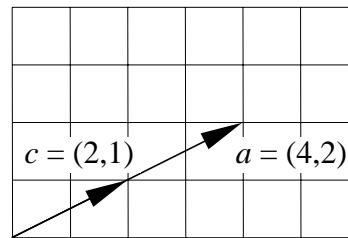
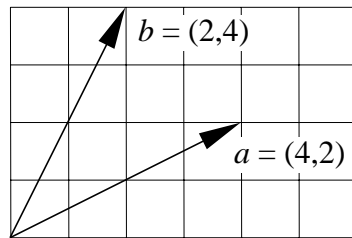
Compare magnitude of vectors with *norms*

$$\|x\| > \|y\|$$

There are several ways to compute $\|x\|$. In other words the size of two vectors can be compared with different norms.

Vector Norms (2)

Consider two element vectors, which lie in a plane



Use geometric lengths to represent the magnitudes of the vectors

$$l_a = \sqrt{4^2 + 2^2} = \sqrt{20}$$

$$l_b = \sqrt{2^2 + 4^2} = \sqrt{20}$$

$$l_c = \sqrt{2^2 + 1^2} = \sqrt{5}$$

We conclude that

$$l_a = l_b \quad \text{and} \quad l_a > l_c$$

or

$$\|a\| = \|b\| \quad \text{and} \quad \|a\| > \|c\|$$

The L_2 Norm

The notion of a geometric length for 2D or 3D vectors can be extended to vectors with arbitrary numbers of elements.

The result is called the *Euclidian* or L_2 norm:

$$\|x\|_2 = (x_1^2 + x_2^2 + \dots + x_n^2)^{1/2} = \left(\sum_{i=1}^n x_i^2 \right)^{1/2}$$

The L_2 norm can also be expressed in terms of the inner product

$$\|x\|_2 = \sqrt{x \cdot x} = \sqrt{x^T x}$$

p -Norms

For any integer p

$$\|x\|_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{1/p}$$

The L_1 norm is sum of absolute values

$$\|x\|_1 = |x_1| + |x_2| + \dots + |x_n| = \sum_{i=1}^n |x_i|$$

The L_∞ norm or *max norm* is

$$\|x\|_\infty = \max(|x_1|, |x_2|, \dots, |x_n|) = \max_i (|x_i|)$$

Although p can be any positive number, $p = 1, 2, \infty$ are most commonly used.

Application of Norms

Are two vectors (nearly) equal?

Floating point comparison of two scalars with absolute value:

$$\frac{|\alpha - \beta|}{|\alpha|} < \delta$$

where δ is a small tolerance.

Comparison of two vectors with norms:

$$\frac{\|y - z\|}{\|z\|} < \delta$$

Notice that preceding statement is **not equivalent to**

$$\frac{\|y\| - \|z\|}{\|z\|} < \delta$$

Orthogonal Vectors

From geometric interpretation of the inner product

$$u \cdot v = \|u\|_2 \|v\|_2 \cos \theta$$

$$\cos \theta = \frac{u \cdot v}{\|u\|_2 \|v\|_2} = \frac{u^T v}{\|u\|_2 \|v\|_2}$$

Two vectors are orthogonal when $\theta = \pi/2$ or $u \cdot v = 0$.

In other words

$$u^T v = 0$$

if and only if u and v are orthogonal.

Orthonormal Vectors

Orthonormal vectors are **unit vectors** that are *orthogonal*.

A **unit** vector has an L_2 norm of one.

The unit vector in the direction of u is

$$\hat{u} = \frac{u}{\|u\|_2}$$

Since

$$\|u\|_2 = \sqrt{u \cdot u}$$

it follows that $u \cdot u = 1$ if u is a unit vector.

Matrices

- Columns and Rows of a Matrix are Vectors
- Addition and Subtraction
- Multiplication by a scalar
- Transpose
- Linear Combinations of Vectors
- Matrix–Vector Product
- Matrix–Matrix Product
- Matrix Norms

Notation

The matrix A with m rows and n columns looks like:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & & \cdots & a_{mn} \end{bmatrix}$$

a_{ij} = element in **row** i , and **column** j

In MATLAB we can define a matrix with

```
>> A = [ ... ; ... ; ... ]
```

where semicolons separate lists of row elements.

The $a_{2,3}$ element of the MATLAB matrix A is $A(2,3)$.

Matrices Consist of Row and Column Vectors

As a collection of column vectors

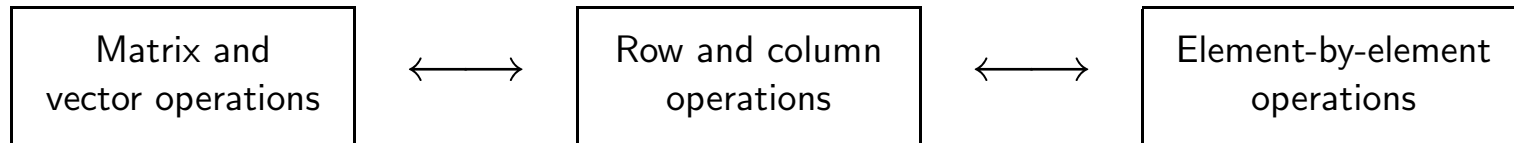
$$A = \left[\begin{array}{c|c|c|c} & & & \\ \hline \mathbf{a}_{(1)} & \mathbf{a}_{(2)} & \cdots & \mathbf{a}_{(n)} \\ \hline & & & \end{array} \right]$$

As a collection of row vectors

$$A = \left[\begin{array}{c} \mathbf{a}'_{(1)} \\ \hline \mathbf{a}'_{(2)} \\ \hline \vdots \\ \hline \mathbf{a}'_{(m)} \end{array} \right]$$

A prime is used to designate a row vector on this and the following pages.

Preview of the Row and Column View



Matrix Operations

- Addition and subtraction
- Multiplication by a Scalar
- Matrix Transpose
- Matrix–Vector Multiplication
- Vector–Matrix Multiplication
- Matrix–Matrix Multiplication

Matrix Operations

Addition and subtraction

$$C = A + B$$

or

$$c_{i,j} = a_{i,j} + b_{i,j} \quad i = 1, \dots, m; \quad j = 1, \dots, n$$

Multiplication by a Scalar

$$B = \sigma A$$

or

$$b_{i,j} = \sigma a_{i,j} \quad i = 1, \dots, m; \quad j = 1, \dots, n$$

Note: Commas in subscripts are necessary when the subscripts are assigned numerical values. For example, $a_{2,3}$ is the row 2, column 3 element of matrix A , whereas a_{23} is the 23rd element of vector a . When variables appear in indices, such as a_{ij} or $a_{i,j}$, the comma is optional

Matrix Transpose

$$B = A^T$$

or

$$b_{i,j} = a_{j,i} \quad i = 1, \dots, m; \quad j = 1, \dots, n$$

In MATLAB

```
>> A = [0 0 0; 0 0 0; 1 2 3; 0 0 0]
```

```
A =
```

```
    0    0    0
    0    0    0
    1    2    3
    0    0    0
```

```
>> B = A'
```

```
B =
```

```
    0    0    1    0
    0    0    2    0
    0    0    3    0
```

Matrix–Vector Product

- The Column View
 - ▷ gives mathematical insight
- The Row View
 - ▷ easy to do by hand
- The Vector View
 - ▷ A square matrix rotates and stretches a vector

Column View of Matrix–Vector Product (1)

Consider a **linear combination of a set of column vectors** $\{a_{(1)}, a_{(2)}, \dots, a_{(n)}\}$. Each $a_{(j)}$ has m elements

Let x_i be a set (a vector) of scalar multipliers

$$x_1 a_{(1)} + x_2 a_{(2)} + \dots + x_n a_{(n)} = b$$

or

$$\sum_{j=1}^n a_{(j)} x_j = b$$

Expand the (hidden) row index

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \dots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Column View of Matrix–Vector Product (2)

Form a matrix with the $a_{(j)}$ as columns

$$\left[\begin{array}{c|c|c|c} & & & \\ \hline & & & \\ \hline a_{(1)} & a_{(2)} & \cdots & a_{(n)} \\ \hline & & & \\ \hline \end{array} \right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \\ \\ \\ b \end{bmatrix}$$

Or, writing out the elements

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Save space with matrix notation

$$Ax = b$$

Column View of Matrix–Vector Product (3)

The matrix–vector product $b = Ax$ produces a vector b from a linear combination of the columns in A .

$$b = Ax \iff b_i = \sum_{j=1}^n a_{ij}x_j$$

where x and b are column vectors

Column View of Matrix–Vector Product (4)

Algorithm 7.1

```
initialize:  $b = \text{zeros}(n, 1)$   
for  $j = 1, \dots, n$   
  for  $i = 1, \dots, m$   
     $b(i) = A(i, j)x(j) + b(i)$   
  end  
end
```

Compatibility Requirement

Inner dimensions must agree

$$\begin{array}{ccc} A & x & = & b \\ [m \times n] & [n \times 1] & = & [m \times 1] \end{array}$$

Row View of Matrix–Vector Product (1)

Consider the following matrix–vector product written out as a linear combination of matrix columns

$$\begin{bmatrix} 5 & 0 & 0 & -1 \\ -3 & 4 & -7 & 1 \\ 1 & 2 & 3 & 6 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ -3 \\ -1 \end{bmatrix}$$
$$= 4 \begin{bmatrix} 5 \\ -3 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 4 \\ 2 \end{bmatrix} - 3 \begin{bmatrix} 0 \\ -7 \\ 3 \end{bmatrix} - 1 \begin{bmatrix} -1 \\ 1 \\ 6 \end{bmatrix}$$

This is the column view.

Row View of Matrix–Vector Product (2)

Now, group the multiplication and addition operations by row:

$$4 \begin{bmatrix} 5 \\ -3 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 4 \\ 2 \end{bmatrix} - 3 \begin{bmatrix} 0 \\ -7 \\ 3 \end{bmatrix} - 1 \begin{bmatrix} -1 \\ 1 \\ 6 \end{bmatrix}$$
$$= \begin{bmatrix} (5)(4) + (0)(2) + (0)(-3) + (-1)(-1) \\ (-3)(4) + (4)(2) + (-7)(-3) + (1)(-1) \\ (1)(4) + (2)(2) + (3)(-3) + (6)(-1) \end{bmatrix} = \begin{bmatrix} 21 \\ 16 \\ -7 \end{bmatrix}$$

Final result is identical to that obtained with the column view.

Row View of Matrix–Vector Product (3)

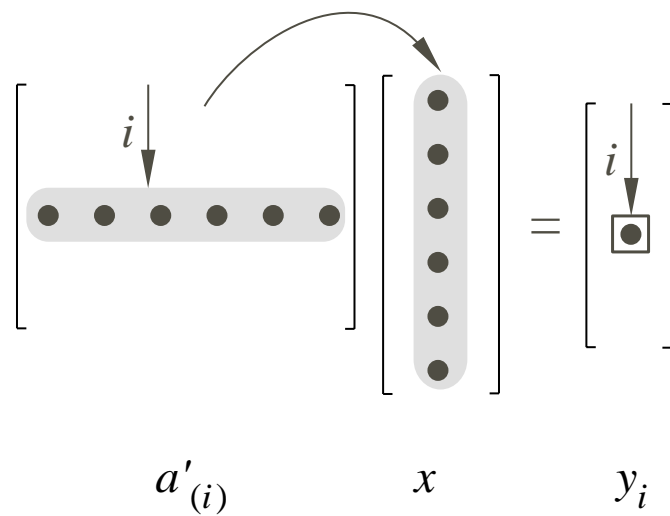
Product of a 3×4 matrix, A , with a 4×1 vector, x , looks like

$$\begin{bmatrix} a'_{(1)} \\ \hline a'_{(2)} \\ \hline a'_{(3)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a'_{(1)} \cdot x \\ a'_{(2)} \cdot x \\ a'_{(3)} \cdot x \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

where $a'_{(1)}$, $a'_{(2)}$, and $a'_{(3)}$, are the *row vectors* constituting the A matrix.

The matrix–vector product $b = Ax$ produces elements in b by forming inner products of the rows of A with x .

Row View of Matrix–Vector Product (4)



Vector View of Matrix–Vector Product

If A is square, the product Ax has the effect of stretching and rotating x .

Pure stretching of the column vector

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

Pure rotation of the column vector

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

Vector–Matrix Product

Matrix–vector product

$$\begin{array}{c} \left[\right. \\ \left. \right] \\ m \times n \end{array} \begin{array}{c} \left[\right. \\ \left. \right] \\ n \times 1 \end{array} = \begin{array}{c} \left[\right. \\ \left. \right] \\ m \times 1 \end{array}$$

Vector–Matrix product

$$\begin{array}{c} \left[\right. \\ \left. \right] \\ 1 \times m \end{array} \begin{array}{c} \left[\right. \\ \left. \right] \\ m \times n \end{array} = \begin{array}{c} \left[\right. \\ \left. \right] \\ 1 \times n \end{array}$$

Compatibility Requirement: Inner dimensions must agree

$$\begin{array}{c} u \\ [1 \times m] \end{array} \begin{array}{c} A \\ [m \times n] \end{array} = \begin{array}{c} v \\ [1 \times n] \end{array}$$

Matrix–Matrix Product

Computations can be organized in **six different ways** We'll focus on just two

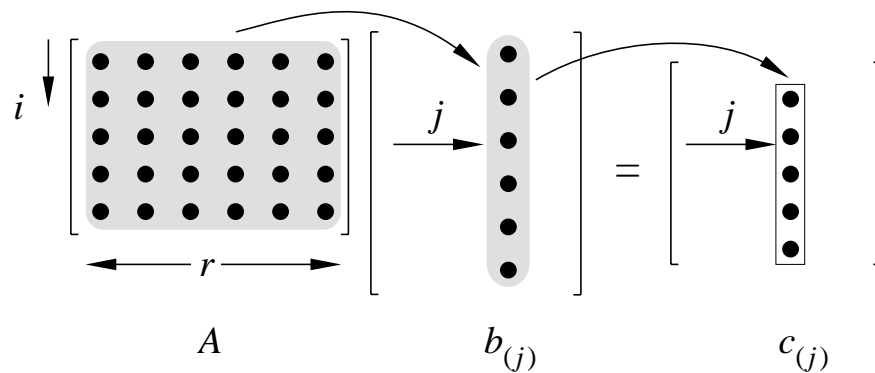
- Column View — extension of column view of matrix–vector product
- Row View — inner product algorithm, extension of column view of matrix–vector product

Column View of Matrix–Matrix Product

The product AB produces a matrix C . The columns of C are linear combinations of the columns of A .

$$AB = C \quad \iff \quad c_{(j)} = Ab_{(j)}$$

$c_{(j)}$ and $b_{(j)}$ are column vectors.



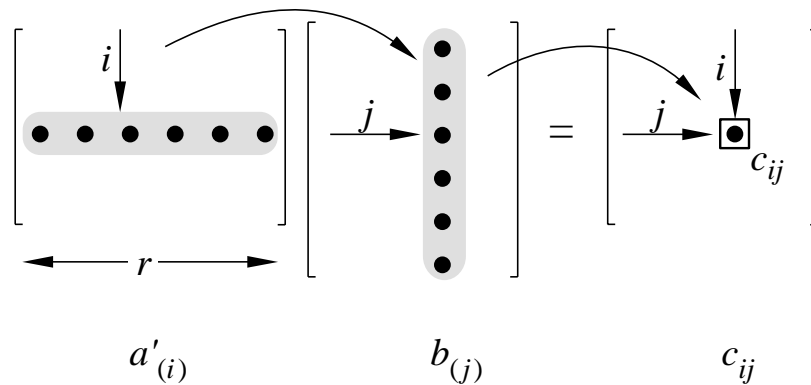
The column view of the matrix–matrix product $AB = C$ is helpful because it shows the relationship between the columns of A and the columns of C .

Inner Product (Row) View of Matrix–Matrix Product

The product AB produces a matrix C . The c_{ij} element is the *inner product* of row i of A and column j of B .

$$AB = C \quad \iff \quad c_{ij} = a'_{(i)} b_{(j)}$$

$a'_{(i)}$ is a row vector, $b_{(j)}$ is a column vector.



The inner product view of the matrix–matrix product is easier to use for hand calculations.

Matrix–Matrix Product Summary (1)

The **Matrix–vector product** looks like:

$$\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} = \begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}$$

The **vector–Matrix product** looks like:

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \end{bmatrix} \begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} = \begin{bmatrix} \bullet & \bullet & \bullet \end{bmatrix}$$

The **Matrix–Matrix product** looks like:

$$\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

Matrix–Matrix Product Summary (2)

Compatibility Requirement

$$\begin{array}{ccc} A & B & = & C \\ [m \times r] & [r \times n] & = & [m \times n] \end{array}$$

Inner dimensions must agree

Also, in general

$$AB \neq BA$$

Matrix Norms

The *Frobenius norm* treats a matrix like a vector: just add up the sum of squares of the matrix elements.

$$\|A\|_F = \left[\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right]^{1/2}$$

More useful norms account for the affect that the matrix has on a vector.

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2 \quad L_2 \text{ or spectral norm}$$

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}| \quad \text{column sum norm}$$

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}| \quad \text{row sum norm}$$

Mathematical Properties of Vectors and Matrices

- Linear Independence
- Vector Spaces
- Subspaces associated with matrices
- Matrix Rank
- Matrix Determinant

Linear Independence (1)

Two vectors lying along the same line are not independent

$$u = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad v = -2u = \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix}$$

Any two independent vectors, for example,

$$v = \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix} \quad \text{and} \quad w = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

define a plane. Any other vector in this plane of v and w can be represented by

$$x = \alpha v + \beta w$$

x is **linearly dependent** on v and w because it can be formed by a linear combination of v and w .

Linear Independence (2)

A set of vectors is linearly independent if it is impossible to use a linear combination of vectors in the set to create another vector in the set.

Linear independence is easy to see for vectors that are orthogonal, for example,

$$\begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ -3 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

are linearly independent.

Linear Independence (3)

Consider two linearly independent vectors, u and v .

If a third vector, w , *cannot* be expressed as a linear combination of u and v , then the set $\{u, v, w\}$ is linearly independent.

In other words, if $\{u, v, w\}$ is linearly independent then

$$\alpha u + \beta v = \delta w$$

can be true *only if* $\alpha = \beta = \delta = 0$.

More generally, if the only solution to

$$\alpha_1 v_{(1)} + \alpha_2 v_{(2)} + \cdots + \alpha_n v_{(n)} = 0 \quad (1)$$

is $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$, then the set $\{v_{(1)}, v_{(2)}, \dots, v_{(n)}\}$ is **linearly independent**

Conversely, if equation (1) is satisfied by at least one nonzero α_i , then the set of vectors is **linearly dependent**.

Linear Independence (4)

Let the set of vectors $\{v_{(1)}, v_{(2)}, \dots, v_{(n)}\}$ be organized as the columns of a matrix. Then the condition of linear independence is

$$\left[\begin{array}{c|c|c|c} & & & \\ \hline & & & \\ \hline v_{(1)} & v_{(2)} & \cdots & v_{(n)} \\ \hline & & & \\ \hline \end{array} \right] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2)$$

The columns of the $m \times n$ matrix, A , are linearly independent if and only if $x = (0, 0, \dots, 0)^T$ is the only n element column vector that satisfies $Ax = 0$.

Vector Spaces

- Spaces and Subspaces
- Span of a Subspace
- Basis of a Subspace
- Subspaces associated with Matrices

Spaces and Subspaces

Group vectors according to number of elements they have.
Vectors from these different groups cannot be mixed.

\mathbf{R}^1 = Space of all vectors with one element. These vectors define the points along a line.

\mathbf{R}^2 = Space of all vectors with two elements. These vectors define the points in a plane.

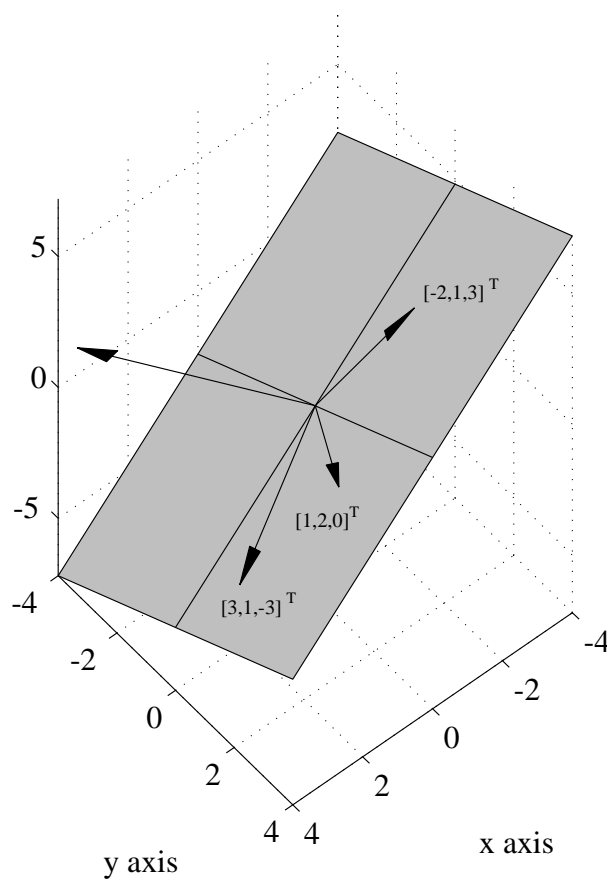
\mathbf{R}^n = Space of all vectors with n elements. These vectors define the points in an n -dimensional space (hyperplane).

Subspaces

The three vectors

$$u = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \quad v = \begin{bmatrix} -2 \\ 1 \\ 3 \end{bmatrix}, \quad w = \begin{bmatrix} 3 \\ 1 \\ -3 \end{bmatrix},$$

lie in the same plane. The vectors have three elements each, so they belong to \mathbf{R}^3 , but they **span** a **subspace** of \mathbf{R}^3 .



Span of a Subspace

If w can be created by the linear combination

$$\beta_1 v_{(1)} + \beta_2 v_{(2)} + \cdots + \beta_n v_{(n)} = w$$

where β_i are scalars, then w is said to be in the subspace that is **spanned** by $\{v_{(1)}, v_{(2)}, \dots, v_{(n)}\}$.

If the v_i have m elements, then the subspace spanned by the $v_{(i)}$ is a subspace of \mathbf{R}^m . If $n \geq m$ it is possible, though not guaranteed, that the $v_{(i)}$ could span \mathbf{R}^m .

Basis and Dimension of a Subspace

- A **basis** for a subspace is a set of **linearly independent** vectors that **span** the subspace.
- Since a basis set must be linearly independent, it also must have the smallest number of vectors necessary to span the space. (Each vector makes a unique contribution to spanning some other direction in the space.)
- The number of vectors in a basis set is equal to the **dimension** of the **subspace** that these vectors span.
- Mutually orthogonal vectors (an orthogonal set) form convenient basis sets, but basis sets need not be orthogonal.

Subspaces Associated with Matrices

The matrix–vector product

$$y = Ax$$

creates y from a linear combination of the columns of A

The column vectors of A form a basis for the **column space** or **range** of A .

Matrix Rank

The **rank** of a matrix, A , is the number of linearly independent columns in A .

$\text{rank}(A)$ is the dimension of the column space of A .

Numerical computation of $\text{rank}(A)$ is tricky due to roundoff.

Consider

$$u = \begin{bmatrix} 1 \\ 0 \\ 0.00001 \end{bmatrix} \quad v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad w = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Do these vectors span \mathbf{R}^3 ? What if $u_3 = \varepsilon_m$?

Matrix Rank (2)

We can use MATLAB's built-in **rank** function for exploratory calculations on (relatively) small matrices

Examples:

```
>> A = [1 0 0; 0 1 0; 0 0 1e-5]    % A(3,3) is small
```

```
A =  
    1.0000         0         0  
         0    1.0000         0  
         0         0    0.0000
```

```
>> rank(A)
```

```
ans =  
     3
```

```
>> A(3,3) = eps/2    % A(3,3) is even smaller
```

```
A =  
    1.0000         0         0  
         0    1.0000         0  
         0         0    0.0000
```

```
>> rank(A)
```

```
ans =  
     2
```

Even though $A(3,3)$ is not identically zero, it is small enough that the matrix is *numerically* rank-deficient

Matrix Determinant (1)

- Only square matrices have determinants.
- The determinant of a (square) matrix is a scalar.
- If $\det(A) = 0$, then A is singular, and A^{-1} does not exist.
- $\det(I) = 1$ for any identity matrix I .
- $\det(AB) = \det(A) \det(B)$.
- $\det(A^T) = \det(A)$.
- Cramer's rule uses (many!) determinants to express the the solution to $Ax = b$.

The matrix determinant has a number of useful properties:

Matrix Determinant (2)

- $\det(A)$ is not useful for numerical computation
 - ▷ Computation of $\det(A)$ is expensive
 - ▷ Computation of $\det(A)$ can cause overflow
- For diagonal and triangular matrices, $\det(A)$ is the product of diagonal elements
- The built in **det** computes the determinant of a matrix by first factoring it into $A = LU$, and then computing

$$\begin{aligned}\det(A) &= \det(L) \det(U) \\ &= (\ell_{11}\ell_{22} \dots \ell_{nn})(u_{11}u_{22} \dots u_{nn})\end{aligned}$$

Special Matrices

- Diagonal Matrices
- Tridiagonal Matrices
- The Identity Matrix
- The Matrix Inverse
- Symmetric Matrices
- Positive Definite Matrices
- Orthogonal Matrices
- Permutation Matrices

Diagonal Matrices (1)

Diagonal matrices have non-zero elements only on the main diagonal.

$$C = \text{diag}(c_1, c_2, \dots, c_n) = \begin{bmatrix} c_1 & 0 & \cdots & 0 \\ 0 & c_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & c_n \end{bmatrix}$$

The **diag** function is used to either create a diagonal matrix from a vector, or and extract the diagonal entries of a matrix.

```
>> x = [1 -5 2 6];  
>> A = diag(x)  
A =  
    1     0     0     0  
    0    -5     0     0  
    0     0     2     0  
    0     0     0     6
```

Diagonal Matrices (2)

The **diag** function can also be used to create a matrix with elements only on a specified *super*-diagonal or *sub*-diagonal. Doing so requires using the two-parameter form of **diag**:

```
>> diag([1 2 3],1)
ans =
     0     1     0     0
     0     0     2     0
     0     0     0     3
     0     0     0     0
>> diag([4 5 6],-1)
ans =
     0     0     0     0
     4     0     0     0
     0     5     0     0
     0     0     6     0
```

Identity Matrices (1)

An identity matrix is a square matrix with ones on the main diagonal.

Example: *The 3×3 identity matrix*

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

An identity matrix is special because

$$AI = A \quad \text{and} \quad IA = A$$

for *any* compatible matrix A . This is like multiplying by one in scalar arithmetic.

Identity Matrices (2)

Identity matrices can be created with the built-in **eye** function.

```
>> I = eye(4)
```

```
I =
```

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

Sometimes I_n is used to designate an identity matrix with n rows and n columns. For example,

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Identity Matrices (3)

A non-square, *identity-like* matrix can be created with the two-parameter form of the eye function:

```
>> J = eye(3,5)
J =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
```

```
>> K = eye(4,2)
K =
     1     0
     0     1
     0     0
     0     0
```

J and K are *not* identity matrices!

Matrix Inverse (1)

Let A be a square (i.e. $n \times n$) with real elements. The *inverse* of A is designated A^{-1} , and has the property that

$$A^{-1} A = I \quad \text{and} \quad A A^{-1} = I$$

The **formal solution** to $Ax = b$ is $x = A^{-1}b$.

$$Ax = b$$

$$A^{-1} Ax = A^{-1}b$$

$$Ix = A^{-1}b$$

$$x = A^{-1}b$$

Matrix Inverse (2)

Although the formal solution to $Ax = b$ is $x = A^{-1}b$, it is considered *bad practice* to evaluate x this way. The recommended procedure for solving $Ax = b$ is Gaussian elimination (or one of its variants) with backward substitution. This procedure is described in detail in Chapter 8.

Solving $Ax = b$ by computing $x = A^{-1}b$ requires more work (more floating point operations) than Gaussian elimination. Even if the extra work does not cause a problem with execution speed, the extra computations increase the roundoff errors in the result. If A is small (say 50×50 or less) and well conditioned, the penalty for computing $A^{-1}b$ will probably not be significant. Nonetheless, Gaussian elimination is preferred.

Functions to Create Special Matrices

Matrix	MATLAB function
Diagonal	<code>diag</code>
Tridiagonal	<code>tridiags</code> (NMM Toolbox)
Identity	<code>eye</code>
Inverse	<code>inv</code>

Symmetric Matrices

If $A = A^T$, then A is called a *symmetric* matrix.

Example:

$$\begin{bmatrix} 5 & -2 & -1 \\ -2 & 6 & -1 \\ -1 & -1 & 3 \end{bmatrix}$$

Note: $B = A^T A$ is symmetric for any (real) matrix A .

To Do

Add slides on:

- Tridiagonal Matrices
- Positive Definite Matrices
- Orthogonal Matrices
- Permutation Matrices