

Computational Methods
CMSC/AMSC/MAPL 460

Vectors, Matrices, Linear Systems, LU
Decomposition,

Ramani Duraiswami,
Dept. of Computer Science

Class Outline

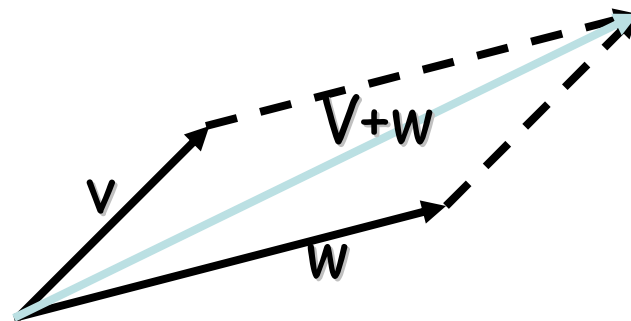
- Much of scientific computation involves solution of linear equations
 - Even non-linear problems are solved by linearization
- Some interpretations of matrices and vectors
- Matrix vector multiplication and complexity
 - Memory organization and access of elements
- Identity, Inverse, Singular Matrices
- Permutation, Lower and Upper Triangular Matrices

Vectors

- Ordered set of numbers: (1,2,3,4)
- Example: (x,y,z) coordinates of a point in space.
- Line joining the origin of coordinates to the point
- Vectors usually indicated with bold lower case letters. Scalars with lower case
- Operations with vectors:
 - Addition operation $\mathbf{u} + \mathbf{v}$, with:
 - Identity $\mathbf{0}$ $\mathbf{v} + \mathbf{0} = \mathbf{v}$
 - Inverse - $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
 - Scalar multiplication:
 - Distributive rule: $\alpha(\mathbf{u} + \mathbf{v}) = \alpha(\mathbf{u}) + \alpha(\mathbf{v})$
 $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$

Vector Addition

$$\mathbf{v} + \mathbf{w} = (x_1, x_2) + (y_1, y_2) = (x_1 + y_1, x_2 + y_2)$$



Vector Spaces

- A *linear combination* of vectors results in a new vector:

$$\mathbf{v} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \dots + \alpha_n \mathbf{v}_n$$

- If the only set of scalars such that

$$\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \dots + \alpha_n \mathbf{v}_n = \mathbf{0}$$

is

$$\alpha_1 = \alpha_2 = \dots = \alpha_n = \mathbf{0}$$

then we say the vectors are *linearly independent*

- The *dimension* of a space is the greatest number of linearly independent vectors possible in a vector set
- For a vector space of dimension n , any set of n linearly independent vectors form a *basis*

Vector Spaces: Basis Vectors

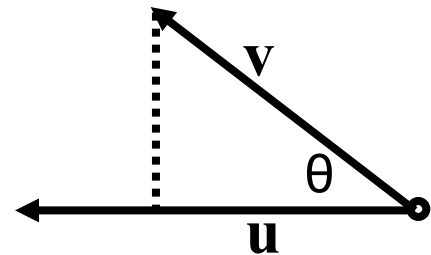
- Given a basis for a vector space:
 - Each vector in the space is a *unique* linear combination of the basis vectors
 - The *coordinates* of a vector are the scalars from this linear combination
 - Best-known example: Cartesian coordinates
 - Example
 - Note that a given vector \mathbf{v} will have different coordinates for different bases

Dot Product

- The *dot product* or, more generally, *inner product* of two vectors is a scalar:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = x_1x_2 + y_1y_2 + z_1z_2 \text{ (in 3D)}$$

- Useful for many purposes
 - Computing the length of a vector: $length(\mathbf{v}) = \sqrt{\mathbf{v} \cdot \mathbf{v}}$
 - *Normalizing* a vector, making it unit-length
 - Computing the angle between two vectors:
 $\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}| |\mathbf{v}| \cos(\theta)$
 - Checking two vectors for orthogonality
 - *Projecting* one vector onto another



Vector norms

$$v = (x_1, x_2, \dots, x_n)$$

Two norm (Euclidean norm)

$$\|v\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$$

If $\|v\|_2 = 1$, v is a unit vector

Infinity norm

$$\|v\|_\infty = \max(|x_1|, |x_2|, \dots, |x_n|)$$

One norm ("Manhattan distance")

$$\|v\|_1 = \sum_{i=1}^n |x_i|$$

For a 2 dimensional vector, write down the set of vectors with two, one and infinity norm equal to unity

Linear Transformations: Matrices

- *A linear transformation:*
 - Maps one vector to another
 - Preserves linear combinations
- Thus behavior of linear transformation is completely determined by what it does to a basis
- Turns out any linear transform can be represented by a *matrix*

Matrices

- By convention, matrix element \mathbf{M}_{ij} is located at row i and column j :

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} & \cdots & \mathbf{M}_{1n} \\ \mathbf{M}_{21} & \mathbf{M}_{22} & \cdots & \mathbf{M}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{M}_{m1} & \mathbf{M}_{m2} & \cdots & \mathbf{M}_{mn} \end{bmatrix}$$

- By convention, vectors are columns:

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix}$$

How are matrices stored in a computer?

- Matlab and Fortran: column by column
 - Indices start at 1
 - What is the most efficient way to access a matrix?
- C arrays are closely linked to pointers
 - Indices start at 0
 - C native matrices are row major
 - Many issues which must be dealt with by properly defining matrices, or using a set of matrix definitions
 - (see <http://www.library.cornell.edu/nr/bookcpdf/c1-2.pdf> for a nice discussion)

Ways to define matrices

- Perhaps all entries are random ...
- More often, they are somehow functions of i and j

Example: A **Hilbert** matrix A of size 5×5 , with element (i, j)

$$a_{ij} = \frac{1}{i + j - 1}.$$

- Matlab code to generate matrix
- If matrix is of size $N \times N$ how many operations are needed to enter values?

```
n=5;
A = zeros(n,n);
for i=1:n,
    for j=1:n,
        A(i,j) = 1/(i+j-1);
    end
end
```

- Sometimes each column or row is given as a formula:
 - Example Vandermonde matrix in polynomial interpolation

Some special matrices

Example: A **Vandermonde** matrix A is defined by a vector of elements x_1, \dots, x_n . Its first column is all ones. Each later column is the preceding one times this vector.

- Matlab code
- How many operations and memory does this take?
- Vectorized operations
- Matrix may be sparse, i.e. most elements are zero.
- How many operations/memory?
- Answer still N^2 unless we avoid referring to the zero elements altogether

```
n = length(x);  
V(:,1) = ones(n,1);  
for j=2:n,  
    V(:,j) = x.*V(:,j-1);  
end
```

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

$$D = \text{diag}([1 \ 2 \ 4 \ 6]);$$