Computational Methods
CMSC/AMSC/MAPL 460

Linear Systems, Matrices, 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
- Identity, Inverse, Singular Matrices
- Permutation, Lower and Upper Triangular Matrices

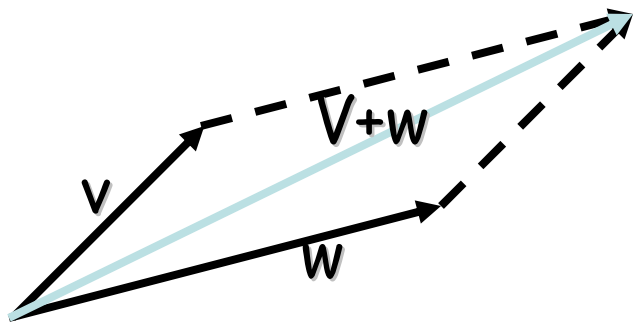
Vectors

- Ordered set of numbers: (1,2,3,4)
- Example: (x,y,z) coordinates of a pt 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 $\mathbf{0}$ $\quad \mathbf{v} + \mathbf{0} = \mathbf{v}$
    - Inverse $-$ $\quad \mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
  - 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:

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

- If the only set of scalars such that

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

  is $\alpha_1 = \alpha_2 = \ldots = \alpha_3 = \mathbf{0}$

  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 **v** will have different coordinates for different bases

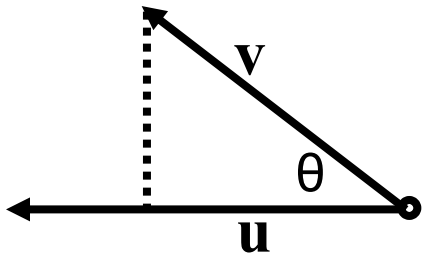
Dot Product

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

$$\mathbf{v}_1 \bullet \mathbf{v}_2 = \mathbf{x}_1\mathbf{x}_2 + \mathbf{y}_1\mathbf{y}_2 + \mathbf{z}_1\mathbf{z}_2 \qquad \text{(in 3D)}$$

- Useful for many purposes
  - Computing the length of a vector: *length*(**v**) = sqrt(**v** • **v**)
  - *Normalizing* a vector, making it unit-length
  - Computing the angle between two vectors:

    $\mathbf{u} \bullet \mathbf{v} = |\mathbf{u}|\,|\mathbf{v}| \cos(\theta)$
  - Checking two vectors for orthogonality
  - *Projecting* one vector onto another



Vector norms

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

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

$$\|v\|_\infty = \max(|x|_1, |x|_2, \ldots, |x|_n)$$

One norm ("Manhattan distance")

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

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

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

Some special matrices

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

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

- 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

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

```
D = diag([1 2 4 6]);
```

Some special matrices

Example: A **tridiagonal** matrix

$$T = \begin{bmatrix} 1 & 3 & 0 & 0 \\ 5 & 2 & 7 & 0 \\ 0 & 9 & 4 & 8 \\ 0 & 0 & 6 & 6 \end{bmatrix}$$

can be defined by

```
T = diag([1 2 4 6]) + diag([5 9 6],-1) + diag([3 7 8], 1);
```

- Matrices may be built up from "blocks" of smaller matrices

Matrix-vector product

- Matrix-vector multiplication applies a linear transformation to a vector:

$$\mathbf{M} \bullet \mathbf{v} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} & \mathbf{M}_{13} \\ \mathbf{M}_{21} & \mathbf{M}_{22} & \mathbf{M}_{23} \\ \mathbf{M}_{31} & \mathbf{M}_{32} & \mathbf{M}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_z \end{bmatrix}$$

- Recall how to do matrix multiplication
- How many operations does a matrix vector product take?

Matrix-vector product

Matrix Transformations

- A *sequence* or *composition* of linear transformations corresponds to the product of the corresponding matrices
  - Note: the matrices to the *right* affect vector first
  - Note: order of matrices matters!
- The *identity matrix* **I** has no effect in multiplication
- Some (not all) matrices have an inverse:

$$\boxed{\mathbf{M}^{-1}\left(\mathbf{M}\left(\mathbf{v}\right)\right)=\mathbf{v}}$$

Ways to implement a matrix vector product

- Access matrix
  - Element-by-element along rows
  - Element-by-element along columns
  - As column vectors
  - As row vectors
- Discuss advantages

```
[m,n]=size(A);
y = zeros(m,1);
for i=1:m,
   for j=1:n,
        y(i) = y(i) + A(i,j)*x(j);
   end
end
```

```
[m,n]=size(A);
y = zeros(m,1);
for i=1:m,
     y(i) = A(i,:) * x;
end
```

```
[m,n]=size(A);
y = zeros(m,1);
for j=1:n,
    y = y + A(:,j)*x(j);
end
```

Matrix norms

- Can be defined using corresponding vector norms
  - Two norm
  - One norm
  - Infinity norm
- Two norm is hard to define … need to find maximum singular value
  - related to idea that matrix acting on unit sphere converts it in to an ellipsoid
- Frobenius norm is defined just using matrix elements

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2$$

$$\|A\|_1 = \max_{\|x\|_1=1} \|Ax\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}|$$

$$\|A\|_\infty = \max_{\|x\|_1=1} \|Ax\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{m} |a_{ij}|$$

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{i,j}|^2 \right)^{1/2}$$