


Linear Algebra in MATLAB

Solving Systems of Linear Equations



Vector Operations

Transpose

- Column vector  Row Vector
- Symbolically ... A^T
- In MATLAB ... A'



Vector Operations (cont'd)

Dot Product (Inner Product)

$$\vec{a} \cdot \vec{b} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \cdot \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = a_x b_x + a_y b_y + a_z b_z$$

■ In MATLAB

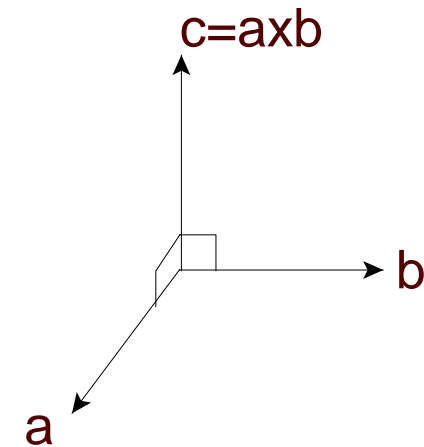
- ▶ `dot(a,b);`
- ▶ `a'*b;`
 - Why does this work?



Vector Operations (cont'd)

Cross Product (Outer Product)

$$\vec{a} \times \vec{b} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_i & a_j & a_k \\ b_i & b_j & b_k \end{vmatrix} = \begin{bmatrix} a_j b_k - a_k b_j \\ a_k b_i - a_i b_k \\ a_i b_j - a_j b_i \end{bmatrix}$$



- In MATLAB `c=cross(a,b)`;
 - ▶ Must be vectors of length 3

Matrix Operations

Matrix - Vector Multiplication

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m,1} & a_{m,1} & \dots & a_{m,n} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$c = Ab$$

$$c_i = \sum_{j=1}^n A_{i,j} b_j$$

In MATLAB, $c=A*b$;



Matrix Operations (cont'd)

Matrix-Matrix Multiplication

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \quad B = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ b_{m,1} & b_{m,2} & \cdots & b_{m,n} \end{bmatrix}$$

$$C = AB$$

$$C_{i,j} = \sum_{k=1}^n A_{i,k} B_{k,j}$$

In MATLAB, $C=A*B$;



Systems of Linear Equations

Notation Defined

☞ We can write any linear system of equations as:

$$\begin{array}{l} a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,n}x_n = b_2 \\ \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \dots + a_{m,n}x_n = b_n \end{array} \quad \longrightarrow \quad \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m,1} & a_{m,1} & \dots & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Here the $a_{i,j}$'s and $b_{i,j}$'s may take on any value

☞ Volunteer to demonstrate this?

☞ Alternatively, we may write $\underline{\underline{A}}\underline{x} = \underline{b}$ or $\mathbf{Ax} = \mathbf{b}$



Solving Linear Systems

- We want to solve the system: $Ax=b$
 - ▶ Multiply both sides by A^{-1}
 - Now, we get $x=A^{-1}b$ Why?
 - Now we must get an expression for A^{-1} !
 - Difficult to do in general
 - ▶ Solve this system without directly computing A^{-1}
 - Many techniques to solve such a system
 - Gaussian Elimination
 - Thomas Algorithm (Tridiagonal Systems)
 - Gauss-Seidel
 - LU - Decomposition
- What are the constraints on A ???



Solving Linear Systems

How to do it in MATLAB

- Solving the system $Ax=b$ in MATLAB
 - ▶ Matlab can invert a matrix to get A^{-1} directly
 - A^{-1} or `inv(A)` computes the inverse of A
 - This is often not necessary, and less efficient than other techniques to solve this system
 - Useful if the inverse is explicitly needed
 - ▶ Alternatively, we may use: $x=A \setminus b$
 - Note that the backslash (`\`) operator is used here!
 - MATLAB computes the solution vector, x .



A Solution Strategy

- Write down all of the equations
 - ▶ Number of equations must equal number of unknowns (i.e. square coefficient matrix)
 - ▶ This is the difficult part!
- Manipulate equations into the form $\mathbf{Ax}=b$
 - ▶ Elements in \mathbf{A} and b may be any value.
- Enter \mathbf{A} and b into MatLab and solve
 - ▶ $x=A\b{b}$; on MatLab command line/m-file
- Interpret & Analyze your results!
 - ▶ Are they reasonable?
 - If not, check your equations for mistakes!



Linear Algebra Tools in MATLAB

Some Useful Functions

`det(A)` Calculates the determinant of a matrix

`cond(A)` Calculates the condition number of A

`eig(A)` Calculates eigenvectors & eigenvalues of A

`rank(A)` Calculates the rank of A

`lu(A)` Calculates LU Factorization of A

“help matfun” for more functions

