Numerical Methods in Linear Algebra, Part Two

Larry Caretto
Mechanical Engineering 501A

Seminar in Engineering Analysis

September 25, 2017

California State University Northridge

Outline

- Review last class
- Pivoting strategies to reduce round-off error in solutions
- Gauss-Jordan for matrix inverses
- The LU method
- Numerical analysis software

California State University Northridge 2

Review last class

- Finite representation of numbers in binary computer gives round-off error
- Machine epsilon is measure of precision of floating point representation
  - 1 + ε = 1 below "machine epsilon"
  - epsilon is usually 1.19×10^{-7} for single and 2.22×10^{-16} for double precision
- Round-off error growth in calculations called error propagation

California State University Northridge 3

Review Error Propagation

- Addition and subtraction errors

$$\varepsilon_{rel} \equiv \frac{(x \pm y) - (\tilde{x} \pm \tilde{y})}{x \pm y} = \frac{\varepsilon_x + \varepsilon_y}{x \pm y} \approx \frac{\varepsilon_x + \varepsilon_y}{\tilde{x} \pm \tilde{y}}$$

- Multiplication and division errors $\varepsilon_{rel} = \frac{xy - \tilde{x}\tilde{y}}{xy}$
- Approximate result for both multiply/divide: $\varepsilon_{rel} \approx \frac{\varepsilon_x}{\tilde{x}} + \frac{\varepsilon_y}{\tilde{y}}$

California State University Northridge 4

Review Matrix Vector Norm

$$\|\mathbf{A}\| = \max_{\|\mathbf{x}\|} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}$$

- Both **Ax** and **x** are matrices
- Can use any matrix norm to compute ||**A**||
- Choosing infinity norm as vector norm gives row sum $\|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$
- Choosing one norm as vector norm gives column sum $\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$

California State University Northridge 5

Review Problem Condition

- In ill-conditioned problems small relative changes in data cause large relative changes in results
- Matrix condition number κ(**A**) = ||**A**|| ||$\mathbf{A}^{-1}$|| of 100 or above indicate ill-conditioning

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \le \kappa(\mathbf{A}) \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} = \kappa(\mathbf{A}) \frac{\|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\|}{\|\mathbf{b}\|}$$

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \kappa(\mathbf{A}) \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} \qquad \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \kappa(\mathbf{A}) \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

California State University Northridge 6

Review Gauss Elimination

- Use each row from row 1 to row n-1 as the "pivot" row
  - Work on each row below the pivot row
    - Multiply pivot row by $a_{row,pivot}/a_{pivot,pivot}$
    - Subtract result from row r to make $a_{row,pivot} = 0$
    - Operation requires subtraction for each column of **A** right of pivot column and for **b**
  - Repeat for each row below pivot
- Repeat for rows 1 to n-1 as pivot rows
- Use back substitution for **x** values

Northridge 7

Review Round-off Error Example

- Same set of equations solved with original order and reverse order

$$\begin{bmatrix} 0.00003 & 3 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1.0002 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 0.00003 & 3 \end{bmatrix}\begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} 1 \\ 1.0002 \end{bmatrix}$$

$$\begin{bmatrix} 0.00003 & 3 \\ 0 & -9999 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1.0002 \\ 1 - \dfrac{1.0002}{0.0003} \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 0 & 2.9997 \end{bmatrix}\begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} 1 \\ 0.9999 \end{bmatrix}$$

- Original order solution less accurate by about four significant figures

Northridge 8

Review Round-off Conclusions

- Showed that order of equations was important factor in round-off error
- Problems caused by small pivot elements (diagonal element on pivot row)
- Found large loss of significant figures with original order but no error when order was reversed
- Want to use this idea in algorithms for reducing round-off error

Northridge 9

Zero and Small Pivots

- Gauss elimination may give zero for the pivot element in one row, but swapping equations can give a nonzero pivot
- Always check for a small number not zero
- Use scaled equations so maximum absolute element in each row is one
- Find row with maximum (scaled) element in the pivot column and then swap

Northridge 10

Finding Inverse Matrices

- For **B = A**$^{-1}$, **AB** = **I,** gives

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & \cdots & a_{nn} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} & b_{13} & \cdots & \cdots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \cdots & \cdots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \cdots & \cdots & b_{3n} \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \cdots & \cdots & b_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 1 \end{bmatrix}$$

- This requires n solutions of n equations in n unknowns (one solution for each b column)

Northridge 11

Finding Inverse Matrices II

- E. g., for the second column of **B** we solve

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & \cdots & a_{nn} \end{bmatrix}\begin{bmatrix} b_{12} \\ b_{22} \\ b_{32} \\ \vdots \\ \vdots \\ b_{n2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}$$

- We can solve for all n columns of the inverse matrix at one time

Northridge 12

Finding Inverse Matrices III

- Gauss-Jordan subtracts pivot row from all rows above and below to give final result shown below

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 1 \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \\ b_{32} \\ \vdots \\ \vdots \\ b_{n2} \end{bmatrix} = \begin{bmatrix} \beta_{12} \\ \beta_{22} \\ \beta_{32} \\ \vdots \\ \vdots \\ \beta_{n2} \end{bmatrix}$$

- Diagonal form gives solutions by inspection: $b_{12} = \beta_{12}$, $b_{22} = \beta_{22}$, $b_{32} = \beta_{32}$, …, $b_{n2} = \beta_{n2}$,

California State University Northridge 13

Gauss-Jordan Matrix Inversion

- Start with augmented **A** matrix

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} & 1 & 0 & 0 & \cdots & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2n} & 0 & 1 & 0 & \cdots & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & \cdots & a_{3n} & 0 & 0 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots & \vdots & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & \cdots & a_{nn} & 0 & 0 & 0 & \cdots & \cdots & 1 \end{bmatrix}$$

- Gauss elimination with diagonals set to 1 and pivot row subtracted from all rows

California State University Northridge 14

Gauss-Jordan Result

- Augmented matrix at end of process

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 & b_{11} & b_{12} & b_{13} & \cdots & \cdots & b_{1n} \\ 0 & 1 & 0 & \cdots & \cdots & 0 & b_{21} & b_{22} & b_{23} & \cdots & \cdots & b_{2n} \\ 0 & 0 & 1 & \cdots & \cdots & 0 & b_{31} & b_{32} & b_{33} & \cdots & \cdots & b_{3n} \\ \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & 1 & b_{n1} & b_{n2} & b_{n3} & \cdots & \cdots & b_{nn} \end{bmatrix}$$

- Inverse is read from augmented (right-hand) part of matrix

California State University Northridge 15

Gauss-Jordan Pseudocode

```
Loop over all rows as pivots (1 to N)
   Find the row with the maximum
      (scaled) element in the pivot
      column and swap it with the pivot
      current pivot row
   Divide all elements the pivot row
       by a[pivot,pivot]
   Loop over all rows, subtracting the
      pivot row times a[row,pivot] from
      each row (1 to N except pivot)
```

California State University Northridge 16

Gauss-Jordan Example

- Solve the set of equations on the right

$$\begin{aligned} 2x_1 - 4x_2 - 26x_3 &= -34\,(i) \\ -3x_1 + 2x_2 + \ \ 9x_3 &= \ \ 13\,(ii) \\ 7x_1 + 3x_2 + \ \ 8x_3 &= \ \ 14\,(iii) \end{aligned}$$

- Divide equation (i) by $a_{11} = 2$

$$\begin{aligned} 1x_1 - 2x_2 - 13x_3 &= -17\,(i) \\ -3x_1 + 2x_2 + \ \ 9x_3 &= \ \ 13\,(ii) \\ 7x_1 + 3x_2 + \ \ 8x_3 &= \ \ 14\,(iii) \end{aligned}$$

- Subtract –3 times (i) from equation (ii) and 7 times (i) from (iii) to replace (ii) and (iii)

California State University Northridge 17

Gauss-Jordan Example II

$$[-3-(-3)1]x_1 + [2-(-3)(-2)]x_2 + [9-(-3)(-13)]x_3 = [13-(-3)(-17)]$$

$$[7-(7)1]x_1 + [3-(7)(-2)]x_2 + [8-(7)(-13)]x_3 = [14-(7)(-17)]$$

- Result from first set of operations

$$\begin{aligned} 1x_1 - 2x_2 - 13x_3 &= -17\,(i) \\ 0x_1 - 4x_2 - 30x_3 &= -38\,(ii) \\ 0x_1 + 17x_2 + 99x_3 &= 133\,(iii) \end{aligned}$$

- Divide equation (ii) by a_{22} = -4

$$\begin{aligned} 1x_1 - 2x_2 - 13x_3 &= -17\,(i) \\ 0x_1 + 1x_2 + 7.5x_3 &= 9.5\,(ii) \\ 0x_1 + 17x_2 + 99x_3 &= 133\,(iii) \end{aligned}$$

California State University Northridge 18

Gauss-Jordan Example III

- Subtract –2 times (ii) from equation (i) and 17 times (ii) from (iii) to replace (i) and (iii)

$$[1-(-2)0]x_1+[-2-(-2)(1)]x_2+[13-(-2)(7.5)]x_3=[-17-(-2)(9.5)]$$

$$[0-(17)1]x_1+[17-(17)(1)]x_2+[99-(17)(7.5)]x_3=[133-(17)(9.5)]$$

- Result from second set of operations

$$\begin{aligned} 1x_1+0x_2+2x_3&=2 \quad (i)\\ 0x_1+1x_2+7.5x_3&=9.5 \quad (ii)\\ 0x_1+0x_2-28.5x_3&=-28.5 \quad (iii)\end{aligned}$$

California State University Northridge 19

Gauss-Jordan Example IV

- Divide equation (iii) by a_{33} = -28.5

$$\begin{aligned} 1x_1+0x_2+2x_3&=2\,(i)\\ 0x_1+1x_2+7.5x_3&=9.5\,(ii)\\ 0x_1+0x_2+1x_3&=1\,(iii)\end{aligned}$$

- Subtract 2 times (iii) from equation (i) and 7.5 times (iii) from (ii) to replace (i) and (ii)

$$[1-(2)0]x_1+[0-(2)(0)]x_2+[2-(2)(0)]x_3=[2-(2)(1)]$$

$$[0-(7.5)0]x_1+[1-(7.5)(-0)]x_2+[7.5-(7.5)(1)]x_3=[9.5-(7.5)(1)]$$

California State University Northridge 20

Gauss-Jordan Example V

- Results after using row iii as pivot row

$$\begin{aligned} 1x_1+0x_2+0x_3&=0\,(i)\\ 0x_1+1x_2+0x_3&=2\,(ii)\\ 0x_1+0x_2+1x_3&=1\,(iii)\end{aligned}$$

- Solutions are seen to be x_1 = 0, x_2 = 2, and x_3 = 1
- No back substitution required
- Not generally used because more computations required (compared to standard Gauss elimination)

California State University Northridge 21

Gauss-Jordan Inverse Example

- Use previous matrix to get inverse

$$[\mathbf{A},\mathbf{I}]=\begin{bmatrix} 2 & -4 & -26 & 1 & 0 & 0\\ -3 & 2 & 9 & 0 & 1 & 0\\ 7 & 3 & 8 & 0 & 0 & 1\end{bmatrix}$$

- After first row as pivot row

$$\begin{bmatrix} 1 & -2 & -13 & 0.5 & 0 & 0\\ 0 & -4 & -30 & 1.5 & 1 & 0\\ 0 & 17 & 99 & -3.5 & 0 & 1\end{bmatrix}$$

California State University Northridge 22

Gauss-Jordan Inverse Example II

- After second row as pivot row

$$\begin{bmatrix} 1 & 0 & 2 & -0.25 & -0.5 & 0\\ 0 & 1 & 7.5 & -0.375 & -0.25 & 0\\ 0 & 0 & -28.5 & 2.875 & 4.25 & 1\end{bmatrix}$$

- Final step (row 3 as pivot) shows [**I**,$\mathbf{A}^{-1}$]

$$\begin{bmatrix} 1 & 0 & 0 & -0.0482456 & -0.201754 & 0.0701754\\ 0 & 1 & 0 & 0.381579 & 0.868421 & 0.263168\\ 0 & 0 & 1 & -0.100877 & -0.149123 & -0.0350877\end{bmatrix}$$

California State University Northridge 23

LU Methods

- Based on factoring original **A** matrix into a product **LU** = **A**
- **L** is lower triangular
- **U** is upper triangular
- Different ways to do this (Doolittle, Crout and Cholesky)
- Can get **L** and **U** without knowing **b**
- Useful when several **b** solutions (for same **A**) required at different times

California State University Northridge 24

Crout Algorithm

- The **L** and **U** elements are stored in the space used for the **A** elements
  - In Crout algorithm, the lower triangular matrix, **L**, has values of 1 on the principal diagonal which does not have to be stored
  - All upper triangular matrix elements, u_{ij}, including diagonal elements are stored in place of the upper a elements
  - This storage pattern for the **L** and **U** matrices is shown on the next slide

25

A = LU

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & \cdots & a_{nn} \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ l_{n1} & l_{n2} & a_{n3} & \cdots & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & u_{nn} \end{bmatrix} = \mathbf{LU}$$

26

How do we get L and U?

- Conventional matrix multiplication $\quad a_{ij} = \sum_{k=1}^{n} l_{ik}u_{kj}$
- **L** and **U** structure give effective upper limits less than n

$l_{11} = 1$

$$a_{1j} = \sum_{k=1}^{n} l_{1k}u_{kj} = l_{11}u_{1j} + (0)u_{2j} + (0)u_{3j} + \cdots + (0)\,u_{nj} = (1)u_{1j} = u_{1j}$$

$$a_{i1} = \sum_{k=1}^{n} l_{ik}u_{k1} = l_{i1}u_{11} + l_{i2}\,(0) + l_{i3}(0) + \cdots + l_{i1}(0) = l_{i1}u_{11}$$

- These two equations give $u_{1j} = a_{1j}$ and $l_{i1} = a_{i1}/u_{11}$ (first u row and l column)

27

General L and U Equations

$$a_{mj} = \sum_{k=1}^{m-1} l_{mk}u_{kj} + l_{mm}\,u_{mj} + \sum\left(\textit{from } k = m+1 \textit{ to } n \textit{ is zero}\right)$$

- Use the following equations to compute components of **L** and **U** matrices ($l_{mm} = 1$)

$$u_{mj} = a_{mj} - \sum_{k=1}^{m-1} l_{mk}u_{kj} \qquad j = m, \ldots, n$$

$$l_{im} = \frac{a_{im} - \sum_{k=1}^{m-1} l_{ik}u_{km}}{u_{mm}} \qquad i = m+1, \ldots, n$$

28

Solving Ax = b

- Define **y** such that **y** = **Ux**
- **b** = **Ax** = **LUx** = **Ly** = **b**
- So we have to solve the following for **y**

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

29

Solving Ax = b Continued

- We find **y** from **y** = **Ux** (**Ly** = **b**)
- $y_1 = b_1$, $y_2 = b_2 - l_{21}y_1$, etc.

$$y_i = b_i - \sum_{k=1}^{i-1} l_{ik}y_k \qquad i = 1, 2, \ldots n$$

- Solution for y_i requires only y_k with $k < i$
- Once **y** is known, we have to solve **y** = **Ux** for **x**

30

Solving **Ax** = **b** Concluded

- Solving **y** = **Ux** is just back substitution

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ y_n \end{bmatrix}$$

California State University Northridge 31

LU with Pivoting

- Can do usual pivoting in **LU** method by selecting maximum u_{mm} for division
- How do we consider row swaps when we have a new **b** vector?
- Maintain one-dimension integer array to keep track of row swaps
- See notes for details
- Library programs call for this array

California State University Northridge 32

Tridiagonal Systems

- Such systems have the following general form: $A_ix_{i-1} + B_ix_i + C_ix_{i+1} = D_i$
- Occur in special cases such as ordinary differential equation boundary value problems and fitting cubic spline polynomials
- Simplified solution procedure, Thomas Algorithm, which is really Gaussian Elimination for this simple system

California State University Northridge 33

Thomas Algorithm*

- Loop over all rows from k = 1 to k = N-2; for each k value compute B_{k+1} and D_{k+1}

$$B_{k+1} \leftarrow B_{k+1} - A_{k+1}C_k / B_k \qquad D_{k+1} \leftarrow D_{k+1} - A_{k+1}D_k / B_k$$

- Compute $x_N = D_N / B_N$
- Loop over all rows from k = N – 1 to k = 2 in reverse order. For each row, k, compute x_k from the following equation

$$x_k = \frac{D_k - C_k x_{k+1}}{B_k}$$

*Also called TriDiagonal Matrix Algorithm (TDMA)

California State University Northridge 34

Iterative Methods

- Used for large sparse systems of simultaneous linear equations
  - May have thousands of equations, but any one equation will have only a few (five to seven) nonzero coefficients
  - Such systems are associated with numerical solution of partial differential equations
  - Covered in ME 501B numerical solution of partial differential equations discussion

California State University Northridge 35

QR Method

- Based on the idea of transforming the **A** matrix of **Ax** = **b** into **A** = **QR**, where
  - **Q** is an orthogonal matrix ($\mathbf{Q}^{-1} = \mathbf{Q}^T$), and
  - **R** is an upper triangular matrix
- Then have $\mathbf{A}^{-1} = (\mathbf{QR})^{-1} = \mathbf{R}^{-1}\mathbf{Q}^{-1} = \mathbf{R}^{-1}\mathbf{Q}^T$
  - $\mathbf{b} = \mathbf{A}^{-1}\mathbf{x} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{x}$ where premultiplication by $\mathbf{R}^{-1}$ is simple because **R** upper triangular
- Formation of orthogonal **Q** known as modified Gram-Schmidt process

California State University Northridge 36

Singular Value Decomposition

- Based on the idea of transforming the **A** matrix of **Ax** = **b** into **A** = **USV**^{T}, where
  - **U** and **V** are orthogonal matrices (**U**$^{-1}$ = **U**^{T} and **V**$^{-1}$ = **V**^{T}) and **S** is a diagonal matrix
- Singular value decomposition (SVD) and the QR Method are also used in solving least squares problems where an experimental data are used to get the best fit to a theoretical model

California State University Northridge 37

Software

- Excel has array functions minverse, mmult, and mdeterm
- Matlab has many functions for matrices, including eigenvectors and eigenvalues
- Linear Algebra PACKage (LAPACK) is available on many computers
- Visual numerics (IMSL library)
- http:/gams.nist.gov/

California State University Northridge 38

Software Issues

- What do you want to do?
  - Solve a problem
  - Develop a general method
- To solve one problem (even several times) use Excel or Matlab
- To develop a general method get libraries from LAPACK, GAMS, or IMSL (Visual Numerics)

California State University Northridge 39

Conclusions

- Solving problems in linear algebra has a strong background and literature
- Much available software
- Be sure that you have sufficient precision to avoid round-off error and use pivoting
- Check matrix condition number if you suspect near linear dependence
- Matlab is versatile tool for matrix equations, eigenvalues and eigenvectors

California State University Northridge 40