Systems of Linear Equations

L. T. Biegler
Chemical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
Biegler@cmu.edu
http://dynopt.cheme.cmu.edu

Gaussian elimination

LU Factorization

Special structures

  Banded

  Sparse - direct methods

QR Decomposition

Singular Value Decomposition

References and software

I. Examples

Linear Equation Systems

- Mass Balances
- Independent Systems of Reactions

Linear Least Squares

- Data Reconciliation
- Parameter Estimation
- System Identification

Subproblems for Nonlinear Problems

- Nonlinear Equations
- Optimization
- Statistical Analysis
- Parametric Sensitivity

Linear Systems Analysis

- Stability
- Controller Design

Basic Building Block for Numerical Methods

"When it comes down to it, all multivariable algorithms are based on solving linear equations."

II. Getting Started

Matrix Notation - Some Definitions

- Scalars - Greek letters, α, β, γ
- Vectors - Roman Letters, lower case
- Matrices - Roman Letters, upper case

Matrix Multiplication - $C = A\,B$ if $A \in \Re^{n \times m}$, $B \in \Re^{m \times l}$ and $C \in \Re^{n \times l}$, $c_{ij} = \sum_k a_{ik}\, b_{kj}$

Transpose - if $A \in \Re^{n \times m}$, then interchanging rows and columns leads to $A^T \in \Re^{m \times n}$

Symmetric Matrix - $A \in \Re^{n \times n}$ (square matrix) and $A = A^T$

Identity Matrix - I, square matrix with ones on diagonal and zeroes elsewhere.

Determinant - "Inverse Volume" measure of a square matrix

$\det(A) = \sum_i (-1)^{i+j} A_{ij} \mathcal{A}_{ij}$ for any j, or

$\det(A) = \sum_j (-1)^{i+j} A_{ij} \mathcal{A}_{ij}$ for any i, where $\mathcal{A}_{ij}$ is the determinant of an order n-1 matrix with row i and column j removed.

Singular Matrix - $\det(A) = 0$

Eigenvalues - $\det(A - \lambda I) = 0$

Eigenvector - $Av = \lambda v$

- These are characteristic values and directions of a matrix.
- For nonsymmetric matrices eigenvalues can be complex, so we often use singular values,

$$\sigma = \lambda(A^T A)^{1/2} \geq 0$$

Some Identities for Determinant

$\det(A\,B) = \det(A)\,\det(B)$ $\qquad$ $\det(A) = \det(A^T)$

$\det(\alpha A) = \alpha^n \det(A)$ $\qquad$ $\det(A) = \prod_i \lambda_i(A)$

Vector Norms

$$\| x \|_p = \left\{ \sum_i |x_i|^p \right\}^{1/p}$$

most common are $p = 1$, $p = 2$ (Euclidean) and $p = \infty$ (max norm $= \max_i |x_i|$)

Matrix Norms

$\|A\| = \max \|A\,x\|/\|x\|$ over x (for p-norms)

$\|A\|_1$ - max. column sum of A, $\max_j \left(\sum_i |A_{ij}| \right)$

$\|A\|_\infty$ - maximum row sum of A, $\max_i \left(\sum_j |A_{ij}| \right)$

$\|A\|_2 = [\sigma_{max}(A)]$ (spectral radius)

$\|A\|_F = [\sum_i \sum_j (A_{ij})^2]^{1/2}$ (Frobenius norm)

$\kappa(A) = \|A\|\,\|A^{-1}\|$ (condition number)

$= \sigma_{max}/\sigma_{min}$ (using 2-norm)

III. Solving Linear Systems

$A\,x = b \qquad x \in \Re^n,\ b \in \Re^n,\ A \in \Re^{n\ x\ n}$

Consider three cases:

1. Unique Solution

$$3\,x_1 + 2\,x_2 = 5$$

$$2\,x_1 - 4\,x_2 = 7$$

$$\Longrightarrow x_1 = 2.125,\ x_2 = -0.6875$$

2. Infinite number of solutions

$$x_1 - 2\,x_2 = 7$$

3. Inconsistent equations

$$2\,x_1 - x_2 = 3$$

$$4\,x_1 - 2\,x_2 = 2$$

What are the characteristics that govern each case?

Some Definitions:

Augmented Matrix - $A_b = [\ A \mid b\ ]$

Rank of a matrix r(A) - order of largest submatrix that has a nonzero determinant.

If $A \in \Re^{m \times n}$ then $r(A) \leq \min(m, n)$

Theorem

If and only if $r(A) = r(A_b)$, then there exists a solution to $A x = b$. If and only if $r(A) = n$, then the solution is unique.

Proof: This is a constructive proof in three parts:

1) Gaussian elimination

2) Equivalence of rank of transformations

3) Simplifications of rank

1) Gaussian Elimination Algorithm

a) Take A x = b

$a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n = b_1$

$a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n = b_2$

... ...

$a_{n1} x_1 + a_{n2} x_2 + \ldots + a_{nn} x_n = b_n$

and let i = 1

b) Choose a_{ii} as the pivot element. If $a_{ii} = 0$, choose an element $a_{kj} \neq 0$ with k, j > i and exchange row i with k and column i with j. If there are no $a_{kj} \neq 0$, go to d).

c) Multiply row of A_b by $-a_{ji}/a_{ii}$ and add to row j,

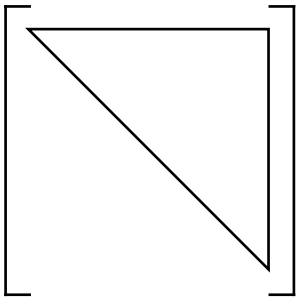
for j = i+1, n

d) i = i+1. If i > n stop, else, go to b).

This procedure terminates in the following ways:

i) U x = p for n equations, if $a_{ii} \neq 0$ always, (i e., no jumps from b to d), where U is upper triangular.

$U \in \Re^{n \times n}$



ii) If no $a_{ii} \neq 0$ can be found for at least one row i, then columns can be rearranged to give:

$$\begin{bmatrix} U & C \\ 0 & 0 \end{bmatrix} x = \begin{bmatrix} p \\ q \end{bmatrix}$$

where U is upper triangular, $U \in \Re^{r \times r}$. Let:

$$S_b = \begin{bmatrix} U & C & p \\ 0 & 0 & q \end{bmatrix}$$

2) Rank Equivalence

Recall:

- $\det(U) = \prod_i u_{ii}$, rank of U = r.
- U was constructed by multiplying a row of A by a scalar and adding to another row.
- Interchanging rows only changes the sign of the determinant

==> $r(A) = r(U)$, also $r(S_b) = r(A_b)$

3) Implications of Rank

Consider the following cases:

i) $U x = p$, $r(A) = r(U) = n$

Now $A x = b$ has a unique solution. Why?

$$x_n = p_n / u_{nn}$$

$$x_{n-1} = (p_n - u_{n-1,\,n}\, x_n)/u_{n-1,n-1}$$

$$x_i = (p_i - \sum_{j=i+1}^{n} u_{ij}\, x_j)/u_{i,i}$$

Note: $u_{ii} \neq 0$ by definition of the algorithm for this case.

ii) $$S_b = \begin{bmatrix} U & C & p \\ 0 & 0 & q \end{bmatrix}$$

If and only if $q \neq 0$, $r(S_b) > r(U)$ and equations are inconsistent. I. e., there is no solution because

$0\ x = q$.

If $q = 0$ then $r < n$ and we partition $x = \begin{bmatrix} y \\ z \end{bmatrix}$ and we can write $Uy + C\ z = p$. Now we can fix z and solve r equations (as in case i)) from $U\ y = p - C\ z$. This gives an infinite number of solutions.

Example - Solution Classification

a) Solve:

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

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

$$\begin{bmatrix} 3 & 4 & 1 & 3 \\ 0 & -10/3 & -1/3 & -4 \\ 0 & 7/3 & 1/3 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 4 & 1 & 3 \\ 0 & -10/3 & -1/3 & -4 \\ 0 & 0 & 3.10 & -0.8 \end{bmatrix}$$

$$x_1 = -0.5484, \quad x_2 = 1.2258, \quad x_3 = -0.2581$$

$$r(A) = r(U) = 3, \det(U) = -31$$

b) Solve

$$A_b = \begin{bmatrix} 2 & 7 & 1 \\ 4 & 14 & 3 \end{bmatrix} \Longrightarrow \begin{bmatrix} 2 & 7 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$r(A_b) = r(S_b) = 2, r(U) = r(A) = 1$$

System is inconsistent, no solution.

IV. Pivoting Algorithms for Gaussian Elimination

Calculations are not done in exact arithmetic. As a result we need to determine good pivoting strategies.

Consider this problem with 11 digits of accuracy:

$$10^{-12}\, x + y = 2$$

$$x + 2\, y = 3$$

$$\begin{bmatrix} 10^{-12} & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix}$$

First Strategy

$$\begin{bmatrix} 10^{-12} & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 10^{-12} & 1 & 2 \\ 0 & -10^{12} & -2\text{x}10^{12} \end{bmatrix}$$

$$\Longrightarrow x = 0,\ y = 2 \qquad \text{Wrong!}$$

Second Strategy - exchange rows

$$\begin{bmatrix} 1 & 2 & 3 \\ 10^{-12} & 1 & 2 \end{bmatrix}$$

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

$$x = -1, y = 2$$

Best solution with 11 digits of accuracy. Close to solution in exact arithmetic.

Pivoting Strategies (Gaussian elimination)

Partial Pivoting - for pivot in col. i,

- choose |largest element| in row i and below as new pivot
- exchange rows.

Complete Pivoting - for pivot in col. i,

- choose |largest element| in row i to n and col. i to n as new pivot
- exchange rows and columns.

Complete pivoting is harder to implement than partial pivoting and only marginally more stable ==> not widely used.

L/U Decomposition

Motivation: Suppose we have a system A x = b with multiple right hand sides or a b is known at a later time. Can we avoid doing Gaussian elimination all over again?

Decompose A = L D U where

U = upper triangular matrix with ones on diagonal

L = lower triangular matrix with ones on diagonal

D = diagonal matrix

We can group these matrices into two forms of LU decomposition:

- Crout decomposition: (LD) U ==> L U = A

  (ones on the diagonal of U)

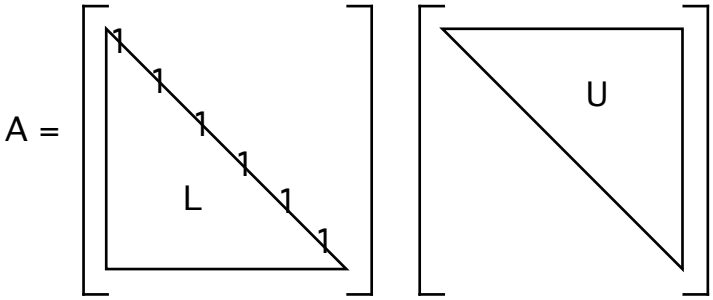
- Doolittle decomposition: L(D U) ==> L U = A

  (ones on the diagonal of L)

Outline of procedure:

Given A x = b. If det(A) ≠ 0, then L and U exist.

We can construct L and U by back-substitution:



Row 1: $a_{11} = u_{11}$ $a_{1j} = u_{1j}$

Column 1: $a_{i1} = u_{11} l_{i1}$ $l_{i1} = a_{i1}/u_{11}$

Row 2: $a_{2j} = l_{21} u_{1j} + u_{2j}$ $u_{2j} = a_{2j} - l_{21} u_{1j}$

Column 2: $a_{i2} = l_{i1}\, u_{12} + l_{i2}\, u_{22}$ $l_{i2} = (a_{i2} - l_{i1}\, u_{12})/u_{22}$

And so on...

Once we have L and U we can write L (U x) = b and:

Solve L y = b by forward substitution

Solve U x = y by backward substitution

Sketch of Algorithm:

1) Set r = 1

2) Set $l_{rr} = 1$

3) Generate row r for U

$$u_{rj} = a_{rj} - \sum_{k=1}^{r-1} l_{rk}\, u_{kj} \qquad \text{for } j = r, n$$

4) Generate column r for L

$$l_{ir} = (a_{ir} - \sum_{k=1}^{r-1} l_{ik}\, u_{kr})/u_{rr} \qquad \text{for } i = r+1, n$$

5) r= r + 1. Go to 2 until r = n.

6) Solve L y = b by forward substitution

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

7) Solve U x = y by backward substitution

$$x_i = (y_i - \sum_{k=i+1}^{n} u_{ik}\, x_k)/u_{ii} \qquad \text{for } i = n, 1$$

Example of LU Decomposition

Solve $\begin{bmatrix} 3 & 4 & 6 \\ 2 & 1 & 2 \\ 7 & 5 & 3 \end{bmatrix} x = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix}$

Decomposition gives:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2/3 & 1 & 0 \\ 7/3 & 2.6 & 1 \end{bmatrix} \qquad U = \begin{bmatrix} 3 & 4 & 6 \\ 0 & -5/3 & -2 \\ 0 & 0 & -5.8 \end{bmatrix}$$

Solve L y = b by forward substitution

$$y_1 = 4$$
$$y_2 = 2 - 2/3(4) = -2/3$$
$$y_3 = 1-7/3(4) - 2.6(-2/3) = - 6.6$$

Solve U x = y by backward substitution

$$x_3 = -6.6/(-5.8) = 1.138$$
$$x_2 = (-2/3 + 2(1.138))/(-5/3) = -0.9656$$
$$x_1 = (4 - 6(1.138) - 4(-0.9656))/3 = 0.3448$$

Notes (Golub and van Loan):

- Operation Count:

LU factorization	$2\ n^3/3$ flops
	($n^3/3$ mult/divides)
$L\ y = b$	n^2 flops
$U\ x = y$	n^2 flops

- If A is symmetric, then LU becomes a Cholesky ($L\ L^T$) factorization with half the work.

- Accuracy

  Relative accuracy x is related to the conditioning of A ($\kappa(A) = \|A\|\ \|A^{-1}\|$), roundoff error in the matrix as well as machine precision.

  Even if $\|A\ x - b\|$ is small, error in x can be large due to large condition number.

  Partial pivoting is usually implemented in L/U decomposition. It has a strong influence in controlling growth of the roundoff error. It does not affect $\kappa(A)$.

V. Solving Sparse Matrices

If n gets large, A usually becomes sparse in many applications, i.e., it has only a few nonzeroes in each row.

For example, if n = 1000 and there are only 5 nonzero elements per row, then the density of the matrix is given by:

$$\text{Density} = \frac{5 \text{ x } 1000}{1000 \text{ x } 1000} = 0.5\ \%$$

We can exploit this density by carefully ordering the nonzero elements in order to reduce storage:

$$O(n) \text{ vs. } O(n^2)$$

and to reduce math operations:

$$O(n) \text{ vs. } O(n^3)$$

Sparse Example

What difference does equation ordering (i.e., pivot selection) have on storage and time?

$$
\begin{aligned}
w + x + y + z &= 4 \\
w - x &= 0 \\
w + y &= 2 \\
w + 2z &= 3
\end{aligned}
$$

Using Gaussian elimination and diagonal pivoting leads to:

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

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

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 4 \\
0 & -2 & -1 & -1 & -4 \\
0 & 0 & 0.5 & -0.5 & 0 \\
0 & 0 & -0.5 & 1.5 & 1
\end{bmatrix}
$$

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 4 \\
0 & -2 & -1 & -1 & -4 \\
0 & 0 & 0.5 & -0.5 & 0 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0.5 & 1 & 0 \\ 1 & 0.5 & -1 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -2 & -1 & -1 \\ 0 & 0 & 0.5 & -0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

100 % Fill-in; 26 adds, 36 multiplications

Now let's reorder these equations and solve the problem as follows:

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

and we only need to pivot the last column, leaving the following L U factors:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.5 & 1 & -1 & 2 \end{bmatrix}$$

$$U = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

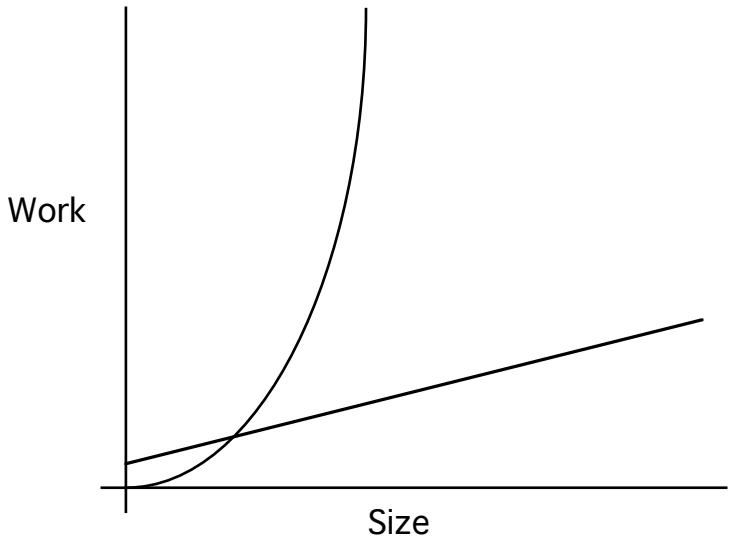
No fill-in for nonzeroes, requires 10 adds, 17 multiplies

50% savings in time

Generalizing to much larger problems leads to much larger savings in storage and time.

Dense work ~ $2n^3/3$

Sparse work ~ $NZ \leq K\ n$



What are the Pivoting Strategies?

Exhaustive search of pivot strategies to minimize fill-in?
Combinatorial optimization problem - expensive to compute

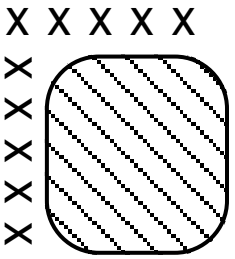
Heuristic Strategy: Min col/Min row strategies

a) delete rows and columns already pivoted
b) select column with fewest nonzeroes.
c) in that column, select row with fewest nonzeroes as pivot
d) go to a) until done

Heuristic Strategy: Min row/Min col strategies
Same with rows and columns reversed (transpose)

Heuristic Strategy: Markowitz Criterion (most popular)

a) delete rows and columns already pivoted
b) Define r_i - remaining nonzeroes in row i
c_j - remaining nonzeroes in column j
c) choose pivot i,j with minimum product $(r_i-1)(c_j-1)$
d) go to a) until done
Motivation: minimize fill-in



How are these strategies implemented?

Analyze - Choose pivot sequence based on Markowitz criterion. Also make sure pivot elements are not too small. This step is expensive.

Factor - Create L U factors using the given pivot sequence

Solve - Forward Elimination: $L\ y = b$

Backward Elimination: $U\ x = y$

Representative Sparse Codes:

Markowitz Based
    MA28, MA48 (Harwell), YSMP
Frontal Methods
    MA45 (Harwell)
Nodal Methods
    SuperLU
Special Structure (Symmetric, $LT^{T}BL$, etc.)
    MA27, MA47 (Harwell)

Notes:
1. Analyze step is useful if the sparsity pattern is the same for a number of different tasks:

- Nonlinear equation solvers by Newton Raphson
- DAE solvers using BDF methods, etc.
- Decomposition methods for optimization

2. There is a trade-off in pivot selection between minimizing fill-in and preserving numerical stability (larger pivots). This is user-specified through a threshold tolerance, t. Choose pivot a_{kk} if $|a_{kk}| \geq t \max_i |a_{ik}|$, $i \geq k$. Typically, $t \sim 0.1$.

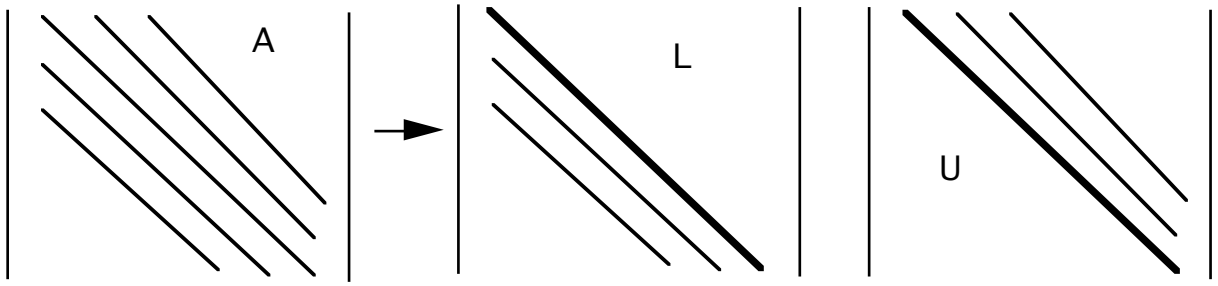
3. The analyze phase sets up triplets and linked lists to speed up solution and save storage.

    NZ index --> Column --> Row --> Value

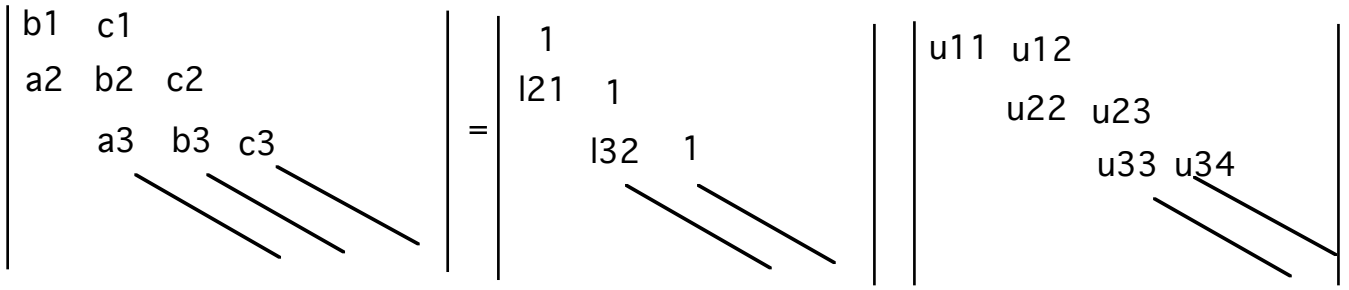
These are augmented to include filled in elements and overwrites.

Special Sparse Systems - Banded Structures

Banded systems arise frequently particularly in staged operations and in discretization of PDEs. The advantage is that for banded systems, the fill-in stays within the band and there is no need for extra storage and much less work:



To illustrate this case, consider the simple case of tridiagonal systems:



We can again start by simple multiplication:

1. $u_{11} = b_1$, $u_{12} = c_1$

$a_i = l_{i,i-1}\, u_{i-1,i-1}$

$b_i = l_{i,i-1}\, u_{i-1,i} + u_{ii}$

$c_i = u_{i,i+1}$

2. $l_{i+1,i} = a_{i+1} / u_{ii}$

3. $u_{i,i+1} = c_i$

4. $u_{i+1,i+1} = b_{i+1} - l_{i+1,i}\, u_{i,i+1}$

Now for the algorithm, consider A x = d and we define:

$s_i = u_{ii}$ $m_i = l_{i+1,i}$ $t = L^{-1} d$

and derive the Thomas algorithm:

1) Reduce A to an upper bidiagonal system (U).

$s_1 = b_1$ $t_1 = d_1$

for i = 1, n-1

$m_i = a_{i+1}/s_i$

$s_{i+1} = b_{i+1} - m_i\, c_i$

$t_{i+1} = d_{i+1} - m_i\, t_i$

2) Back Substitution

Set $x_n = t_n/s_n$

for i = n-1, 1 $x_i = (t_i - c_i\, x_{i+1})/ s_i$

The algorithm requires only 3 n additions and 6 n multiplications (vs. $n^3/3$).

(A block version of this algorithm if the key to solving distillation models.)

VI. QR Factorization

Motivation:

- need more stable way to solve linear systems if they are ill-conditioned
- instead of dividing by small numbers (pivoting), multiply by specialized matrices
- goal is to write A = Q R where Q is an orthonormal matrix ($Q^TQ = I$) and R is upper triangular.

In addition Q R factorization has the following characteristics:

- QR can deal with singular A matrices (L U will blow up). In this case, R will have a zero on the diagonal.

- QR is much more stable than L U. It requires no scaling or partial pivoting strategies.

- QR can handle rectangular matrices and gives:

$$A = [\ Q_1 \mid Q_2]\ \begin{bmatrix} R \\ \end{bmatrix} = Q_1 R$$

- This is natural way to find the null space of A. Q_2 is orthogonal to all of the columns of A.
- QR is twice as expensive as dense LU. There are sparse QRs as well but improvements are not as dramatic and method are not as popular as for sparse LU.

How does it work?

Define Householder matrix:

$W = I - 2\, u\, u^T$ where $\| u \| = 1$

This matrix is orthonormal:

$$\begin{aligned} W^T W &= (I - 2\, u\, u^T)^T (I - 2\, u\, u^T) \\ &= I - 4\, u\, u^T + 4\, u\, (u^T u)\, u^T = I \end{aligned}$$

and we now choose the vector u so that multiplication by W zeroes out columns of A.

Let a be a column of A and define the vector

$$r^T = [\, r_1, 0, 0 \ldots, 0]$$

with $r_1 = \| a \|$. Then

$$u = (a - r)/\|a-r\|$$

allows us to write:

$$\begin{aligned} W\, a &= (I - 2\, u\, u^T)\, a \\ &= a - 2\, (a - r)(a - r)^T/\|a-r\|^2\, a = r \end{aligned}$$

The Householder matrix will rotate any vector a to r as long as they have the same length. This procedure is known as a Givens rotation.

So now we apply these rotations to the matrix A

Choose a_1, r_1 and u_1 as above, with $r_{11} = \| a_1 \|$, then

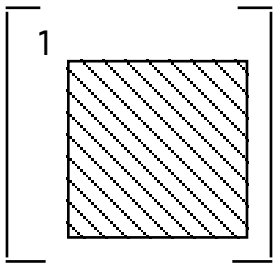
$$W_1 A = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \tilde{a}_{n2} & \cdots & \tilde{a}_{nn} \end{bmatrix}$$

We can now do the same for the second column using the elements from $\tilde{a}_{22}$ to $\tilde{a}_{n2}$. We can define a vector a_2 and r_2 so that

$$a_2^T = [\, 0, \tilde{a}_{22}, \ldots, \tilde{a}_{n2}] \qquad r_2^T = [\, 0, \|a_2\|, 0, \ldots, 0]$$

$$u_2 = (a_2 - r_2)/\|a_2 - r_2\|$$

As the first elements are zero, the matrix W_2 looks like:



and the first row and column of W_1A are not affected. As a result,

$$W_2W_1A = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \tilde{\tilde{a}}_{33} & \tilde{\tilde{a}}_{3n} \\ 0 & 0 & \tilde{\tilde{a}}_{n3} & \tilde{\tilde{a}}_{nn} \end{bmatrix}$$

We continue using W_3 using

$$u_3 = (a_3 - r_3)/\|a_3 - r_3\|$$

$$a_3^T = [\, 0, 0, \tilde{\tilde{a}}_{33}, \ldots, \tilde{\tilde{a}}_{n3}]$$

$$r_3^T = [\, 0, 0, \|a_3\|, 0, \ldots, 0]$$

This process repeats to zero out the n-1 subdiagonal columns until we have:

$$(W_nW_{n-1} \ldots W_2W_1)\, A = R$$

Defining $Q^T = (W_nW_{n-1} \ldots W_2W_1)$ as an orthonormal matrix, we have:

$$Q^T A = R \text{ or } A = Q\, R$$

For A x = b, we have:

$$Q\,R\,x = b \qquad x = R^{-1}\,Q^{T}b$$

Notes on implementation:

Q is not stored or even created. The vectors $u_1, u_2, \ldots u_{n-1}$ can be stored efficiently in a lower triangular matrix along with R.

Efficient dense strategies are part of LINPACK and LAPACK.

QR Example

Solve Ax = b:

$$\begin{bmatrix} 1 & 2 & 5 \\ 4 & 3 & 1 \\ 6 & 1 & 2 \end{bmatrix} x = \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}$$

$$a_1^T = [\ 1, 4, 6]$$

$$r_1^T = [\ 7.28, 0, 0]$$

$$u_1^T = (a_1 - r_1)^T / \|a_1 - r_1\| = [-0.657, 0.418, 0.627]$$

$$W_1 = I - 2\, u_1\, u_1^T = \begin{bmatrix} 0.1374 & 0.5494 & 0.8242 \\ 0.5494 & 0.6500 & -0.5249 \\ 0.8242 & -0.5249 & 0.2126 \end{bmatrix}$$

$$W_1 A = \begin{bmatrix} 7.2801 & 2.7472 & 2.8846 \\ 0 & 2.5241 & 2.3474 \\ 0 & 0.2861 & 4.0211 \end{bmatrix}$$

Repeat with the second column

$$a_2^T = [\, 0, 2.5241, 0.2861]$$

$$r_2^T = [\, 0, 2.5402, 0]$$

$$u_2^T = (a_2 - r_2)^T / \|a_2 - r_2\| = [0, -0.0564, 0.9984]$$

$$W_2 = I - 2\, u_2\, u_2^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9936 & 0.1126 \\ 0 & 0.1126 & -0.9936 \end{bmatrix}$$

$$R = W_2 W_1\, A = \begin{bmatrix} 7.2801 & 2.7472 & 2.8846 \\ 0 & 2.5402 & 2.7853 \\ 0 & 0 & -3.7311 \end{bmatrix}$$

and this matrix is upper triangular. Thus we have:

$$A = Q\, R$$

$$= \begin{bmatrix} 0.1373 & 0.6388 & -0.7570 \\ 0.5494 & 0.5868 & 0.5948 \\ 0.8241 & -0.4976 & -0.2704 \end{bmatrix} \begin{bmatrix} 7.2801 & 2.7472 & 2.8846 \\ 0 & 2.5402 & 2.7853 \\ 0 & 0 & -3.7311 \end{bmatrix}$$

$$x = A^{-1}\, b = R^{-1}\, Q^T\, b$$

$$x = \begin{bmatrix} -0.1449 \\ 1.0580 \\ 0.4058 \end{bmatrix}$$

VII. Solving Least Squares Problems

Consider the problem:

$$\text{Min } 1/2\ (Ax - b)^T(Ax - b)$$

where $n > m$ and $A \in \Re^{n \times m}$, $b \in \Re^n$, $x \in \Re^m$

The least squares solution is:

$$\partial/\partial x\ ((Ax - b)^T(Ax - b)/2) = A^TAx - A^Tb = 0$$

and if A is full rank (and A^TA is nonsingular) then we have:

$$x = -(A^TA)^{-1}A^Tb$$

An efficient and very stable way to solve this problem is by using QR factorizations

$$\text{Using} \qquad A = [Q_1 | Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1R$$

$$\begin{aligned} (A^TA)^{-1}A^Tb &= (R^TQ_1^TQ_1\ R)^{-1}R^TQ_1^Tb \\ &= (R^TR)^{-1}R^TQ_1^Tb \\ &= R^{-1}R^{-T}R^TQ_1^Tb \\ &= R^{-1}Q_1^Tb \end{aligned}$$

If R is nonsingular, we have a unique solution to this problem. However, if R or A^TA is singular, then the problem is overparameterized and this yields an infinite number of solutions. In this case, we are interested in a particular solution, x^*, that has the smallest norm. How do we get this?

Singular Value Decomposition

This is the most expensive way to solve a linear system but it is also the most stable and works even for singular systems.

Here we decompose the matrix A into:

$$A = U\ D\ V^T$$

where $A \in \Re^{n \times m}$

$U \in \Re^{n \times n}$ and orthonormal

$V \in \Re^{m \times m}$ and orthonormal

$D \in \Re^{n \times m}$ where $D_{ii} = \sigma_i$, $i = 1, \min(n,m)$ and zero otherwise.

Here σ_i are singular values of A (eigenvalues of A^TA or AA^T) and are always nonnegative.

The solution of the least distance linear least squares problem is given by:

$$A^TA\ x = A^T\ b$$
$$(U\ D\ V^T)^T\ (U\ D\ V^T)\ x = V\ D\ U^Tb$$
$$V\ D^TD\ V^T\ x = V\ D\ U^Tb$$

Following the proof in Dennis and Schnabel leads to the solution:

$$x = V\ D^+U^Tb = A^+\ b$$

where

A^+ - pseudo inverse of A

D^+ - $D^+_{ii} = 1/\sigma_i$ (if $\sigma_i > 0$) i = 1, min(n,m) and zero otherwise.

SVD is obtained by repeated QR factorizations and scalar shifting (i.e., solve $Q(A^TA - \beta I) Q$) to reveal the singular values, σ_i

Work is a constant multiple of the QR work (see Golub and Van Loan for details)

SVD is an indispensible tool for:

- Solution of ill-conditioned and singular linear systems
- Matrix analysis
- Linear systems theory

VIII. Classification of Methods

Dense Linear Algebra ($n \leq 1000$)

L U	$n^3/3$ multiplies
Q R	$2n^3/3$ multiplies
S V D	$\sim 10\ n^3/3$ multiplies

Sparse Direct Methods ($100 \leq n \leq 1{,}000{,}000$)

Banded Solvers

Sparse L U

Large-scale Indirect Methods ($n \geq 100{,}000$)

Diagonally dominant

- Jacobi, Gauss-Seidel, SOR

Preconditioned Krylov Solvers

- Symmetric (Conjugate Gradient)
- Unsymmetric (GMRES)

IX. Software

Dense Linear Algebra ($n \leq 1000$)

Platforms (MATLAB, Mathematica, MathCAD)

LAPACK - publically available on Netlib.org

Sparse Direct Methods ($100 \leq n \leq 1{,}000{,}000$)

Banded Solvers - LAPACK

Sparse L U (Harwell, NAG Libraries)

Large-scale Indirect Methods ($n \geq 100{,}000$)

Parallel implementations are the main motivation for these methods. Requires specialized hardware as well as software.

Diagonally dominant

Jacobi, Gauss-Seidel, SOR (See Netlib)

Preconditioned Krylov Solvers (See Netlib)

Symmetric (Conjugate Gradient)

Unsymmetric (GMRES)

References

- Dennis, J.E. and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ 1986.

- Duff, I., A. Erisman and J. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.

- Forsythe, G.E,, M.A. Malcolm and C.B. Moler, *Computer Methods For Mathematical Computations,* Prentice-Hall, Englewood Cliffs, NJ, 1977.

- Golub, G. H. and C. F. Van Loan, *Matrix Computations,* Johns Hopkins Press, Baltimore, 1996

- Heath, M.T., *Scientific Computing: An Introductory Survey,* McGraw-Hill, New York, 1997.

- Nocedal, J. and S.J. Wright, *Numerical Optimization*, Springer Verlag, 1999.
-
- Phillips, G. M. and P. J. Taylor, *Theory and Applications of Numerical Analysis*, Academic Press, second edition, London, 1996.