

Lecture 18: Numerical Linear Algebra #3

Direct methods for solving Linear Least Squares problems

Outline

- 0) Finish LU decomposition
 - Algorithms
 - Operation Counts
 - LU vs $\text{inv}(A)$
- 1) Linear Least Squares Problems
 - A) Example: Polynomial Fitting
 - B) Fundamental idea: projection onto $C(A)$
 - C) The Normal Equations $A^T A x = A^T b$
 - D) Orthogonalization techniques $A = QR$

Direct methods for solution of linear systems $Ax = b$

Gaussian elimination and the LU factorization.

Overview: Given a linear systems $Ax = b$

$Ax = b$

GE w/ Partial Pivoting

$$\begin{pmatrix} E_1 & P_1 \\ \vdots & \vdots \\ E_n & P_n \end{pmatrix} A = U$$

$$PA = LU$$

$$l_{ij} = \frac{a_{ij}}{a_{ii}}$$

$$E_1 = \begin{bmatrix} 1 & & & \\ -l_{21} & 1 & & \\ -l_{31} & & 1 & \\ -l_{n1} & & & 1 \end{bmatrix}$$

$$A \rightarrow \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

Direct methods for solution of linear systems $Ax=b$

General Algorithm, LU with partial pivoting (ltx from Moler)

```
function [L,U,p] = ltx(A)
```

```
%LUTX Triangular factorization, textbook version
% [L,U,p] = ltx(A) produces a unit lower triangular matrix L,
% an upper triangular matrix U, and a permutation vector p,
% so that L*U = A(p,:)
```

```
[n,n] = size(A);
p = (1:n);
```

```
for k = 1:n-1
```

```
% Find index of largest element below diagonal in k-th column
[r,m] = max(abs(A(k:n,k)));
m = m+k-1;
```

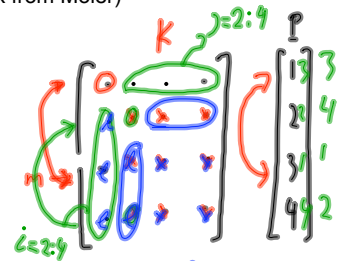
```
% Skip elimination if column is zero
if (A(m,k) == 0)
```

```
% Swap pivot row
if (m ~= k)
    A([k m],:) = A([m k],:);
    p([k m]) = p([m k]);
end
```

```
% Compute multipliers
i = k+1:n;
A(i,k) = A(i,k)/A(k,k);
```

```
% Update the remainder of the matrix
j = k+1:n;
A(i,j) = A(i,j) - A(i,k)*A(k,j);
end
```

```
% Separate result
L = tril(A,-1) + eye(n,n);
U = triu(A);
```



$$A([k \ m], :) = A([m \ k], :)$$

$$A(i, k) * A(k, i)$$

$$L U x = b$$

$$P A = L U$$

$$A x = b$$

$$P A x = P b = b(p)$$

$$L U x = b(p)$$

$$L C = b(p)$$

$$U x = C$$

Direct methods for solution of linear systems $Ax=b$

General Algorithm: forward and backward substitution

```
function x = forward(L,x)
```

```
% FORWARD. Forward elimination.
```

```
% For lower triangular L, x = forward(L,b) solves L*x = b.
```

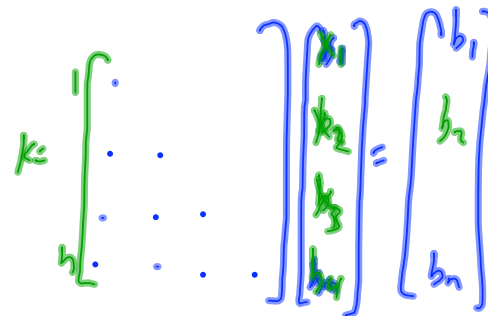
```
[n,n] = size(L);
x(1) = x(1)/L(1,1);
```

```
for k = 2:n
```

```
    j = 1:k-1;
```

```
    x(k) = (x(k) - L(k,j)*x(j))/L(k,k);
```

```
end
```



```
function x = backsubs(U,x)
```

```
% BACKSUBS. Back substitution.
```

```
% For upper triangular U, x = backsubs(U,b) solves U*x = b.
```

```
[n,n] = size(U);
x(n) = x(n)/U(n,n);
```

```
for k = n-1:-1:1
```

```
    j = k+1:n;
```

```
    x(k) = (x(k) - U(k,j)*x(j))/U(k,k);
```

```
end
```

Direct methods for solution of linear systems $Ax=b$

Matlab's Backslash and bslashtx

$$x=A \setminus b$$

```

function x = bslashtx(A,b)
% BSLASHTX Solve linear system (backslash)
% x = bslashtx(A,b) solves A*x = b

[n,n] = size(A);
if isequal(triu(A,1),zeros(n,n))
    % Lower triangular
    x = forward(A,b);
    return
elseif isequal(tril(A,-1),zeros(n,n))
    % Upper triangular
    x = backsubs(A,b);
    return
elseif isequal(A,A')
    [R,fail] = chol(A);
    if ~fail
        % Positive definite
        y = forward(R',b);
        x = backsubs(R,y);
        return
    end
end

% Triangular factorization
[L,U,p] = lutx(A);

% Permutation and forward elimination
y = forward(L,b(p));

% Back substitution
x = backsubs(U,y);

```

Operation Counts for Gaussian Elimination w/ PP

Dense Matrices

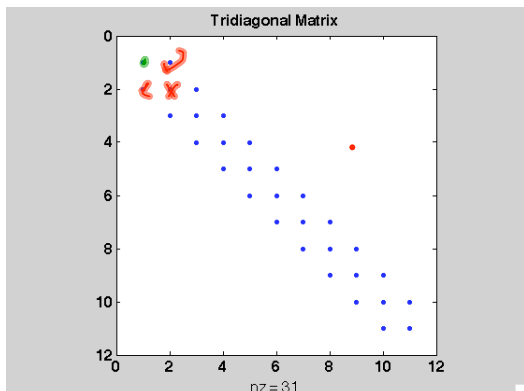
L_4
 n^2
 $(n-1)^2$
 n^2
 $\frac{n^2 n}{3} \quad O\left(\frac{n^3}{3}\right)$

L_4
 $\frac{n^2}{2} \quad x=A \setminus b$

$L_4 \quad O\left(\frac{n^3}{3}\right)$
 ForwardBack $O(n^2)$

Operation Counts for Gaussian Elimination w/ PP

Sparse Banded Matrices *spy(A)*



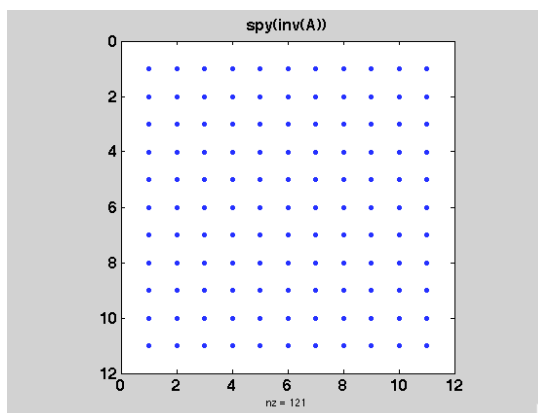
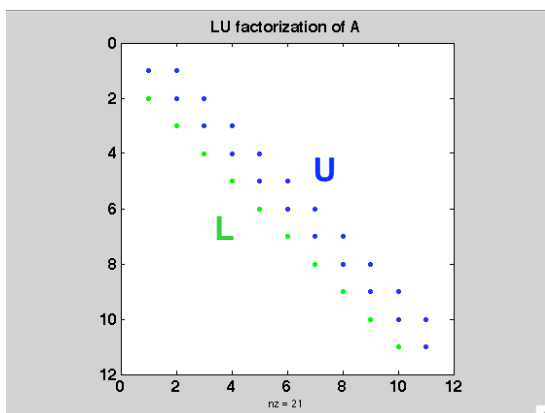
banded $[i,j]=find(A)$
Diag $b=1$ $b=\max(abs(j-i))+1$
Tridiag $b=2$
cost LU for tridiag $b^2 n$
Banded systems $b^2 n \sim 4n$

LU vs. A^{-1}

Operation Costs:

For special matrices can be significantly better than $n^3/3$
 Always better than $\underline{x}=A^{-1}\underline{b}$ $O(n^3)$

$Ax=b$
 $x=A\b{b}$
 *$x=inv(A)*b$*



MATLAB dos and don'ts:

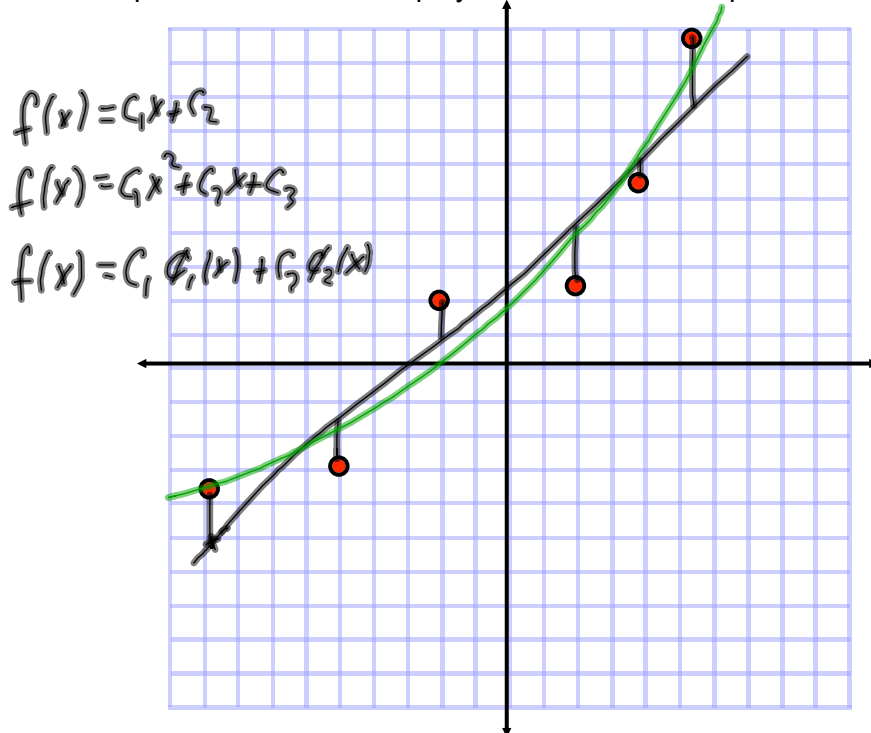
DON'T use: $\underline{x}=inv(A)*\underline{b}$
DO use: $\underline{x}=A\b{b}$

$inv(A) \sim O(n^3) + O(n^2)$
Sparse matrix

(or for multiple RHS $[L,U]=lu(A)$ $x_i=U(L\b{b}_i)$)

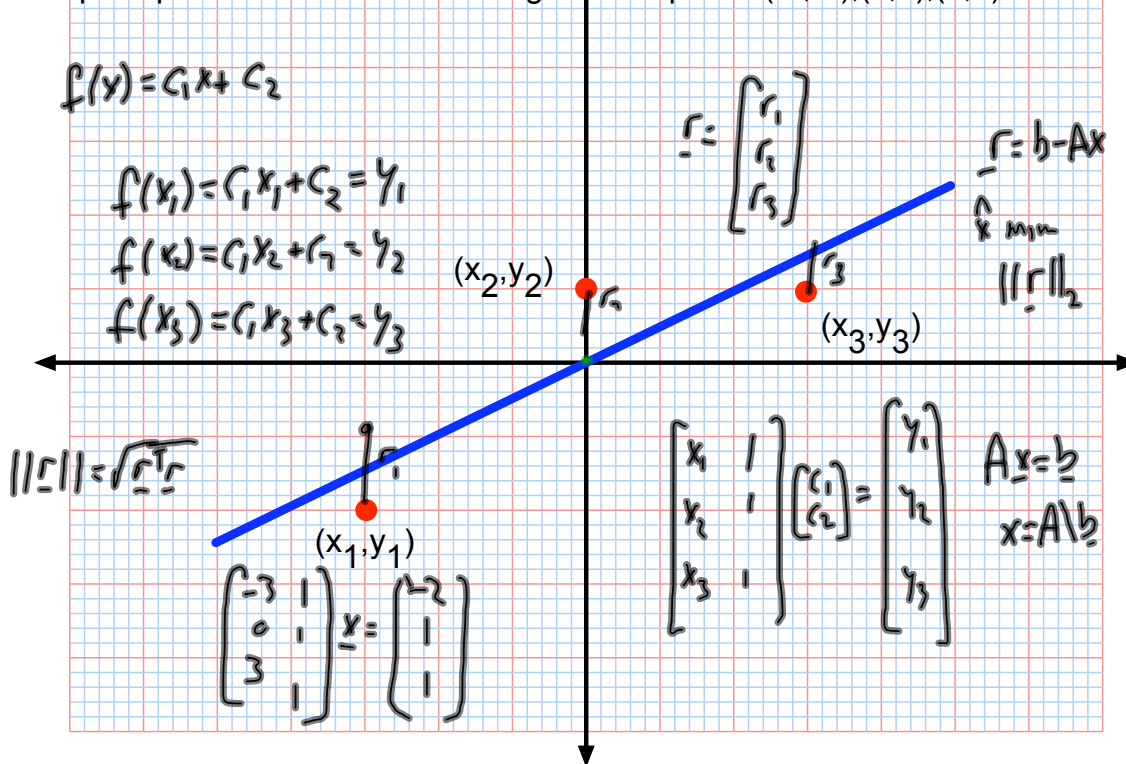
Introduction to Linear-Least Squares problems

Example: best fit n'th order polynomial to $m > n$ data points



Introduction to Linear-Least Squares problems

Simplest problem: best fit line through 3 data points $(-3, -2), (0, 1), (3, 1)$



Orthogonal Projection onto a Subspace

$Ax=b$

$$\begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

columns of $A \in \mathbb{R}^3$

$A\hat{x} = p$

$\underline{r} = b - p = b - A\hat{x}$

$A^T \underline{r} = 0$

$A^T(b - p) = 0$

$A^T(b - A\hat{x}) = 0$

$A^T A \hat{x} = A^T b$

p projection of b onto $r(A)$

$\underline{r} \in N(A^T) \quad x \in N(A)$

$A^T \underline{r} = 0 \quad Ax = 0$

$$\begin{bmatrix} -a_1^T \\ -a_2^T \\ -r^T \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a_1^T \underline{r} \\ a_2^T \underline{r} \\ r^T \underline{r} \end{bmatrix}$$

Normal Equations

$$\begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 3 & 1 \end{bmatrix} \underline{c} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \quad A^T b = \begin{bmatrix} -3 & 0 & 3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} -3 & 0 & 3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -3 & 1 \\ 0 & 1 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 18 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 9 \\ 0 \end{bmatrix} \quad \begin{matrix} c_1 = \frac{1}{2} \\ c_2 = 0 \end{matrix}$$

Solution of General Linear Least-Squares by the normal equations

fit $f(x) = c_1 \phi_1(x) + c_2 \phi_2(x) + c_3 \phi_3(x)$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$A = \begin{bmatrix} \phi_1(x) & \phi_2(x) & \phi_3(x) \\ \vdots & \vdots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \phi_3(x_n) \end{bmatrix} \quad \underline{c} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$A \underline{c} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \phi_3(x_1) \\ \vdots & \vdots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \phi_3(x_n) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$f(x_1) = y_1$
 $f(x_2) = y_2$
 \vdots
 $f(x_n) = y_n$

msfmb

$$c = A \setminus y \quad A^T A c = A^T y$$

$$c = (A^T A) \setminus (A^T y) \quad A \underline{x} = b$$

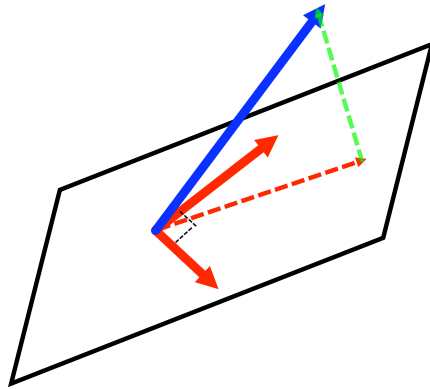
Normal Equations and Condition number squaring

The SVD and the $\text{cond}(A)_2$

Normal Equations and Condition number squaring

Example: $A = \begin{bmatrix} 1 & a & 0 \\ 1 & 0 & a \end{bmatrix}$

***Solution of Least Squares by Orthogonalization
(QR factorization)***



***Solution of Least Squares by Orthogonalization
(QR factorization)***

**QR Factorization Algorithms:
#1 Modified Gram-Schmidt Orthogonalization**

Idea: Take $A \rightarrow Q$ by repeated projections, get $R=Q^T A$

**QR Factorization Algorithms:
#1 Modified Gram-Schmidt Orthogonalization**

Algorithm:

***QR Factorization Algorithms:
#2 Orthogonalization by Householder transformation***

Idea: Take $A \rightarrow R$ directly by repeated applications of Q matrices
(similar algorithm to LU factorization)