

Engineering Analysis ENG 3420 Fall 2009

Dan C. Marinescu

Office: HEC 439 B

Office hours: Tu-Th 11:00-12:00

Lecture 14

- Last time:
 - Solving systems of linear equations (Chapter 9)
 - Graphical methods
 - Cramer's rule
 - Gauss elimination
- Today:
 - Discussion of pivoting
 - Tri-diagonal system solver
 - Examples
- Next Time
 - LU Factorization (Chapter 10)

```

function x=GaussNaive(A,b)
ExA=[A b];
[m,n]=size(A);
q=size(b);
if (m~=n)
    fprintf ('Error: input matrix is not square; n = %3.0f, m=%3.0f \n', n,m);
End
if (n~=q)
    fprintf ('Error: vector b has a different dimension than n; q = %2.0f \n', q);
end
n1=n+1;
for k=1:n-1
    for i=k+1:n
        factor=ExA(i,k)/ExA(k,k);
        ExA(i,k:n1)= ExA(i,k:n1)-factor*ExA(k,k:n1);
    End
End
x=zeros(n,1);
x(n)=ExA(n,n1)/ExA(n,n);
for i=n-1:-1:1
    x(i) = (ExA(i,n1)-ExA(i,i+1:n)*x(i+1:n))/ExA(i,i);
end

```

```

>> A=[1 1 1 0 0 0;
0 -1 0 1 -1 0;
0 0 -1 0 0 1;
0 0 0 1 -1;
0 10 -10 0 -15 -5;
5 -10 0 -20 0 0]
A =
    1     1     1     0     0     0
    0    -1     0     1    -1     0
    0     0    -1     0     0    -1
    0     0     0     1     1    -1
    0    10   -10     0   -15    -5
    5   -10     0   -20     0     0
b =
    [0  0  0  0  0  200]
>> b=b'
b =
    0
    0
    0
    0
    0
    200
>> x = GaussNaive(A,b)
x =
    NaN
    NaN
    NaN
    NaN
    NaN
    NaN

```

Pivoting

- If a coefficient along the diagonal is 0 (problem: division by 0) or close to 0 (problem: round-off error) then the Gauss elimination causes problems.
- Partial pivoting → determine the coefficient with the largest absolute value in the column below the pivot element. The rows can then be switched so that the largest element is the pivot element.
- Complete pivoting → check also the rows to the right of the pivot element are also checked and switch columns.

```

function x = GaussPivot(A,b)
% GaussPivot: Gauss elimination pivoting
%   x = GaussPivot(A,b): Gauss elimination with pivoting.
% input:
%   A = coefficient matrix
%   b = right hand side vector
% output:
%   x = solution vector

[m,n]=size(A);
if m~=n, error('Matrix A must be square'); end
nb=n+1;
Aug=[A b];
% forward elimination
for k = 1:n-1
    % partial pivoting
    [big,i]=max(abs(Aug(k:n,k)) );
    ipr=i+k-1;
    if ipr~=k
        Aug([k,ipr],:)=Aug([ipr,k],:);
    end
    for i = k+1:n
        factor=Aug(i,k)/Aug(k,k);
        Aug(i,k:nb)=Aug(i,k:nb)-factor*Aug(k,k:nb);
    end
end
% back substitution
x=zeros(n,1);
x(n)=Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
    x(i)=(Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end

```

```

function x=GaussPartialPivot(A,b)
ExtendedA=[A b];
[m,n]=size(A);
q=size(b);
if (m~=n)
    fprintf ('Error: input matrix is not square; n = %3.0f, m=%3.0f \n', n,m);
End
if (n~=q)
    fprintf ('Error: vector b has a different dimension than n; q = %2.0f \n', q);
end
n1=n+1;
for k=1:n-1
    [largest,i]=max(abs(ExtendedA(k:n,k)));
    nrow = i +k -1;
    if nrow ~=k
        ExtendedA([k,nrow],:) = ExtendedA([nrow,k],:);
    end
end
for k=1:n-1
    for i=k+1:n
        factor=ExtendedA(i,k)/ExtendedA(k,k);
        ExtendedA(i,k:n1)= ExtendedA(i,k:n1)-factor*ExtendedA(k,k:n1);
    End
End
x=zeros(n,1);
x(n)=ExtendedA(n,n1)/ExtendedA(n,n);
for i=n-1:-1:1
    x(i) = (ExtendedA(i,n1)-ExtendedA(i,i+1:n)*x(i+1:n))/ExtendedA(i,i);
end

```

A =

1	1	1	0	0	0
0	-1	0	1	-1	0
0	0	-1	0	0	-1
0	0	0	0	1	-1
0	10	-10	0	-15	-5
5	-10	0	-20	0	0

```
>> k=4; n=6; A(k:n,k)
```

```
ans =
```

```
0
0
-20
```

```
>> k=4; n=6; A(2,k:6)
```

```
ans =
```

```
1 -1 0
```

```
>> k=4; n=6; [largest,i]=max(abs(A(k:n,k))); nrow = i +k -1, largest, i
```

```
nrow =6
```

```
largest =20
```

```
i =3
```


Tridiagonal system solver

```
function x = Tridiag(e,f,g,r)
% Tridiag: Tridiagonal equation solver banded system
%   x = Tridiag(e,f,g,r): Tridiagonal system solver.
% input:
%   e = subdiagonal vector
%   f = diagonal vector
%   g = superdiagonal vector
%   r = right hand side vector
% output:
%   x = solution vector
n=length(f);
% forward elimination
for k = 2:n
    factor = e(k)/f(k-1);
    f(k) = f(k) - factor*g(k-1);
    r(k) = r(k) - factor*r(k-1);
end
% back substitution
x(n) = r(n)/f(n);
for k = n-1:-1:1
    x(k) = (r(k)-g(k)*x(k+1))/f(k);
end
```