Lecture 5II

- Nonlinear equations
- Nested FOR loops
- Vector codes

Nonlinear system f1

$$x_1^2 + x_2^2 - 4 = 0$$

$$x_1 - x_2 = 0$$

Nonlinear system f2

SOLVE

$$f_1(x_1, x_2) = 2x_1^2 + x_2^2 - 24 = 0$$

$$f_2(x_1, x_2) = x_1^2 - x_2^2 + 12 = 0$$

Nonlinear system f3

$$f_1(x) = exp(x_1) + x_2 * x_3 - 3 = 0$$
$$f_2(x) = \frac{x_1}{x_2} + x_3^2 - log(x_2) = 0$$
$$f_3(x) = \frac{x_1}{x_1 + x_2 + x_3} - sin(x_3) = 0$$

nonlinear functions

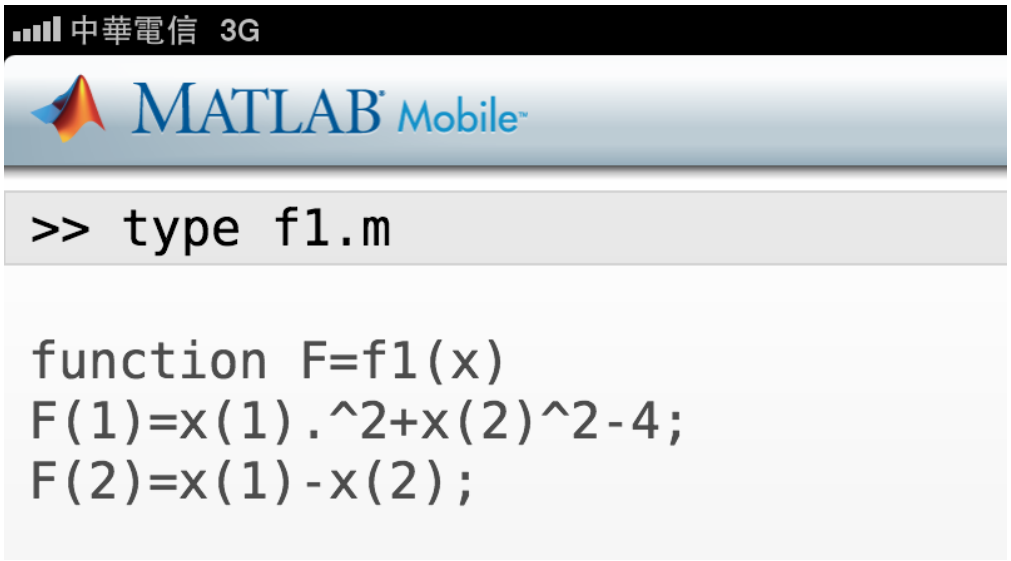
$$\mathrm{F}(x_1, x_2) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix}$$

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 4$$

$$f_2(x_1, x_2) = x_1 - x_2$$

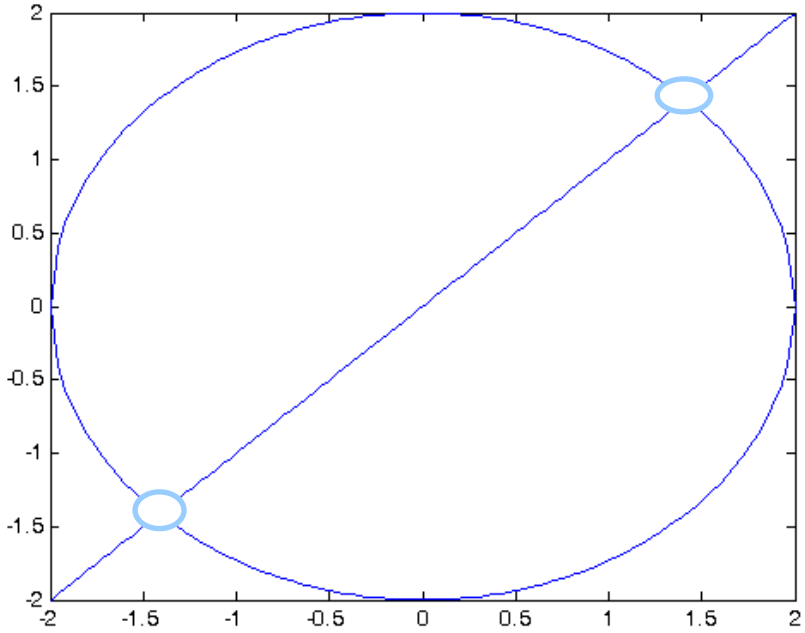
Function evaluation

```
function F = f1(x)
    F(1) =  x(1).^2 + x(2)^2-4;
    F(2) =  x(1) - x(2);
return
```



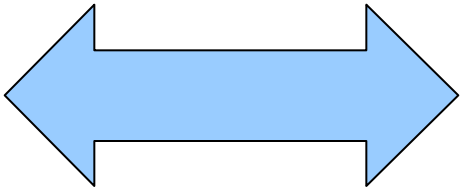
Geometric interpretation

```
x=linspace(-2,2);
plot(x,x); hold on;
plot(x,sqrt(4-x.^2))
plot(x,-sqrt(4-x.^2))
```



Least square of nonlinear functions

$$\min_{x_1, x_2} \sum_i f_i(x_1, x_2)^2$$

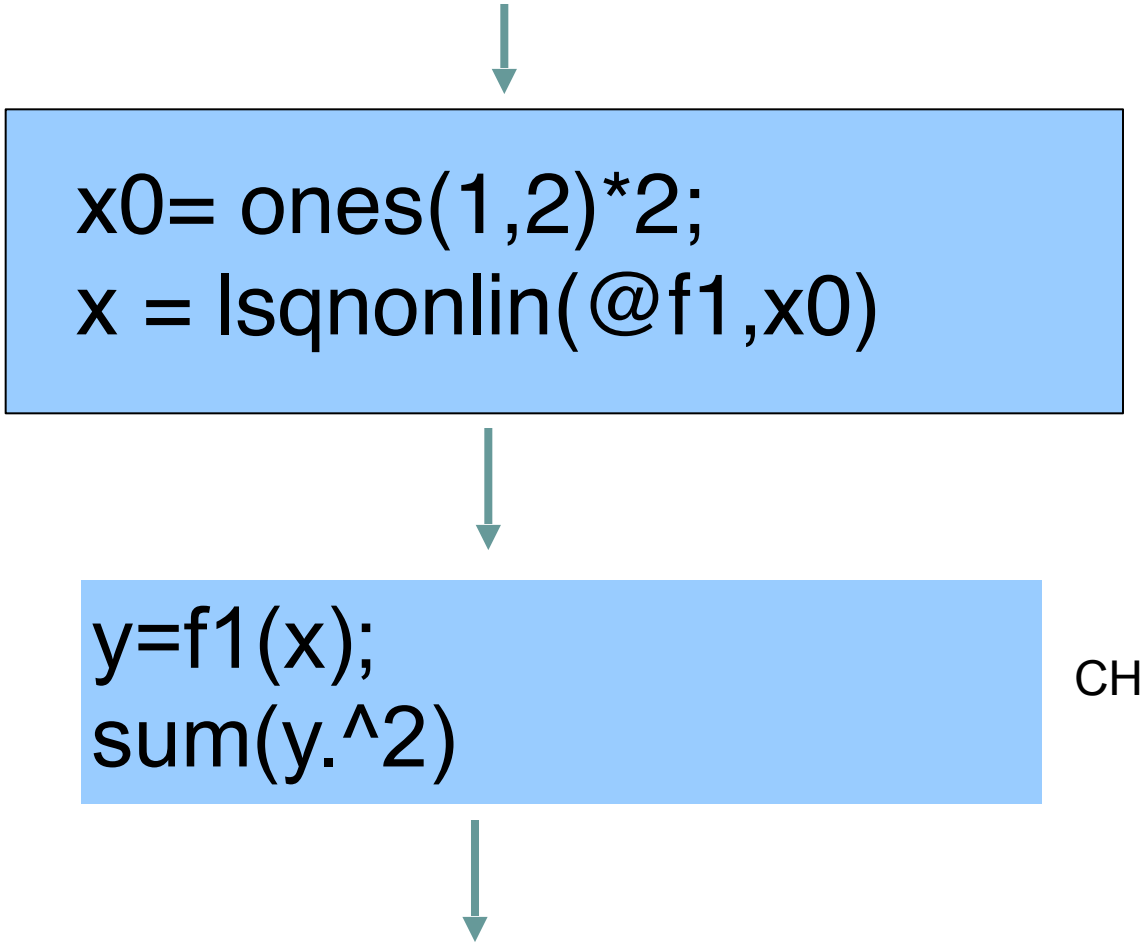


SOLVE

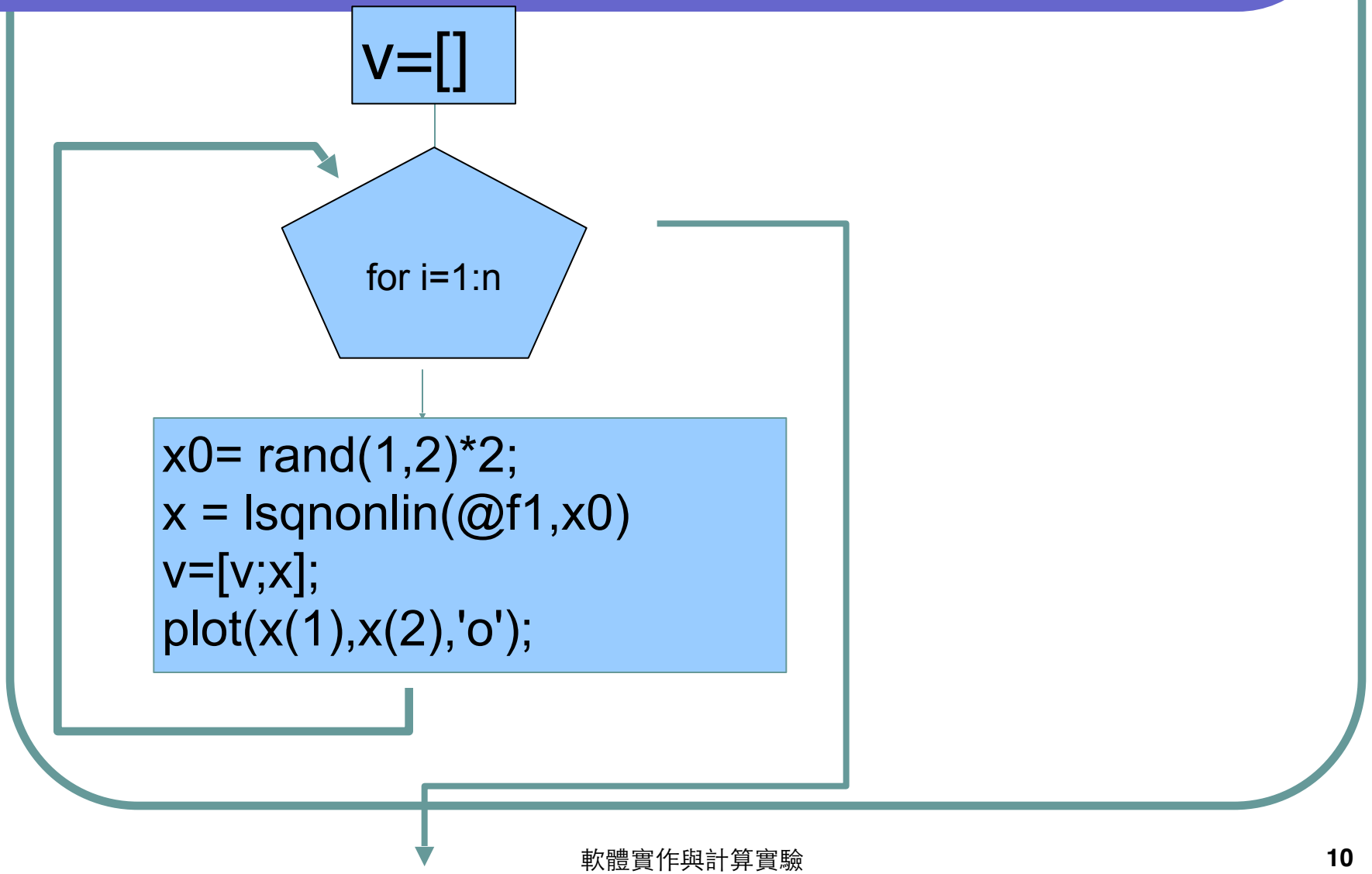
$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 4 = 0$$

$$f_2(x_1, x_2) = x_1 - x_2 = 0$$

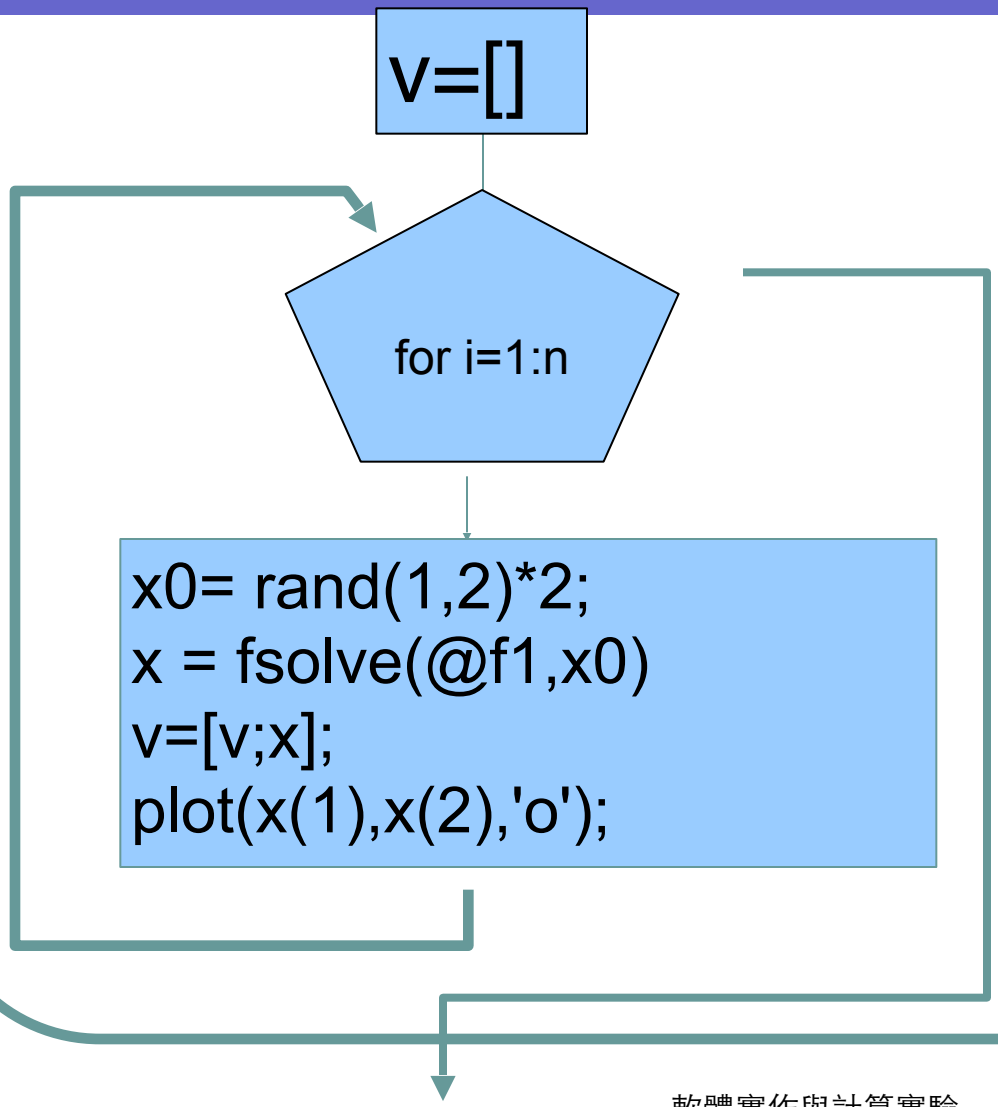
Find a zero



Multiple roots



Multiple roots



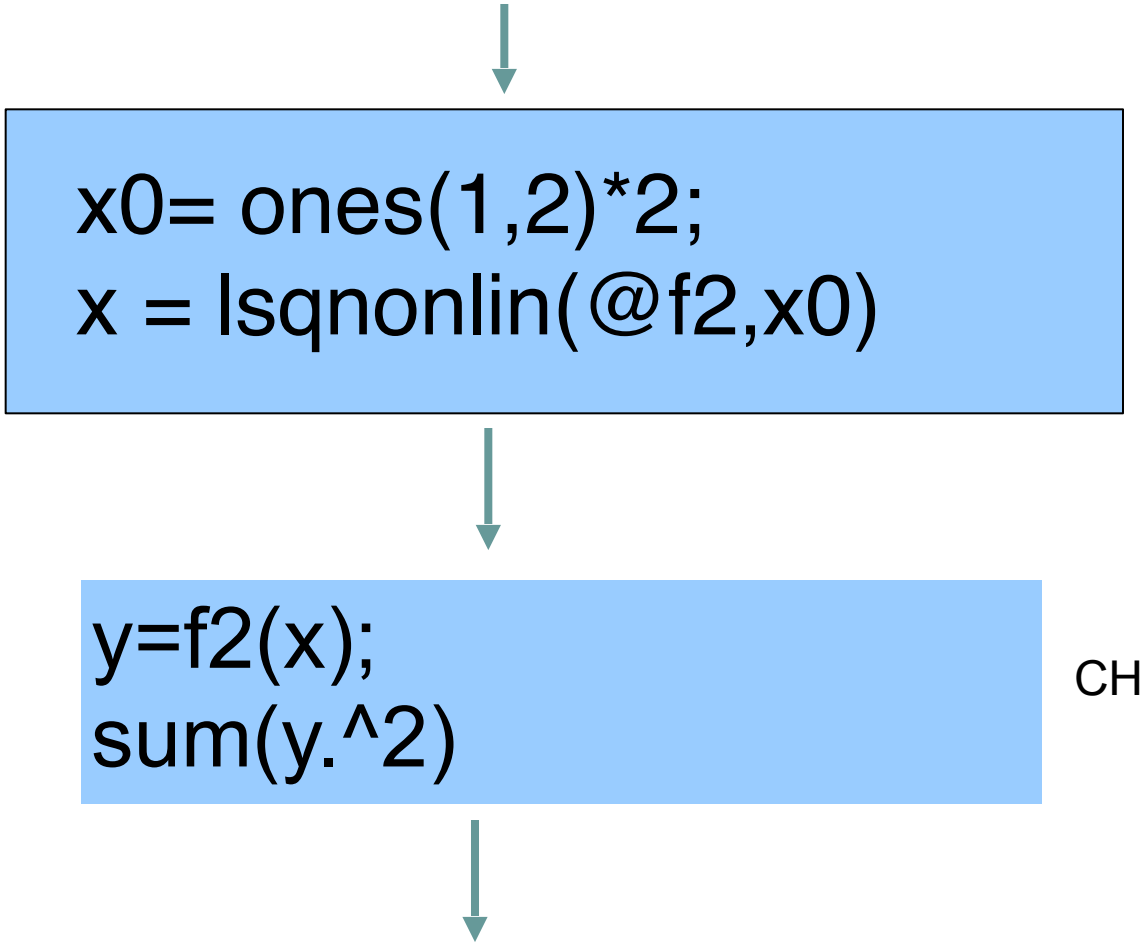
Function evaluation

```
function F = f2(x)
   F(1) =  2*x(1).^2 + x(2)^2-24;
   F(2) =  x(1).^2 - x(2).^2+12;
return
```

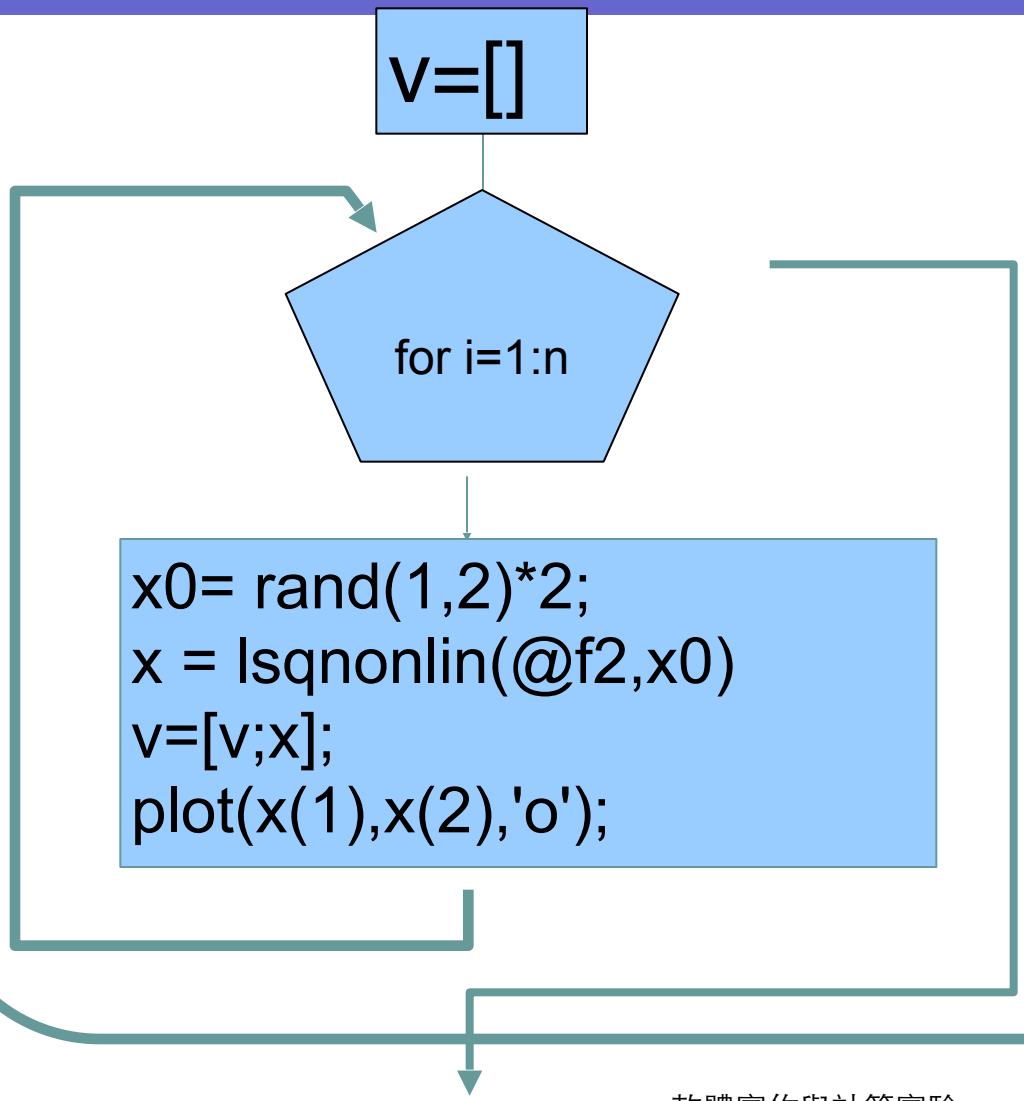
$$f_1(x_1, x_2) = 2x_1^2 + x_2^2 - 24 = 0$$

$$f_2(x_1, x_2) = x_1^2 - x_2^2 + 12 = 0$$

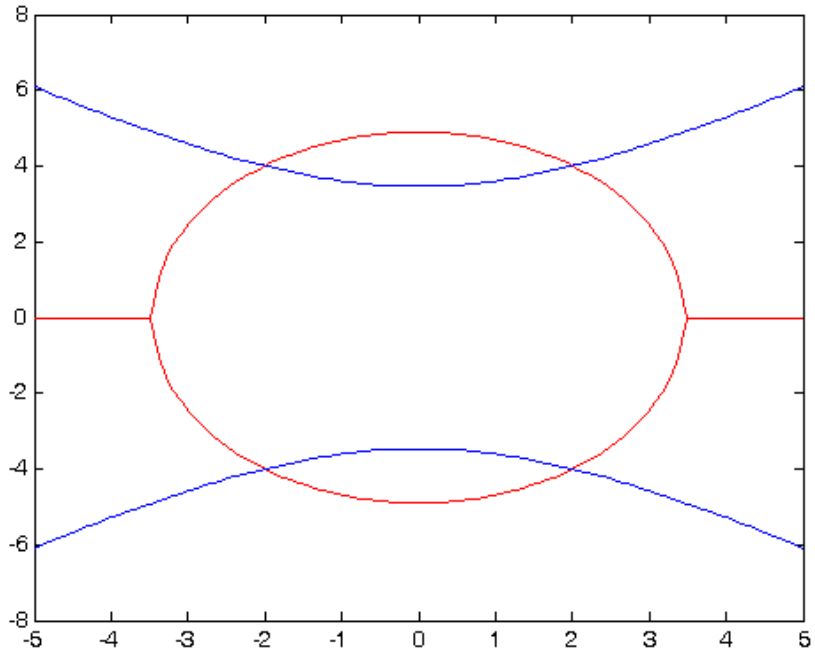
Find a zero

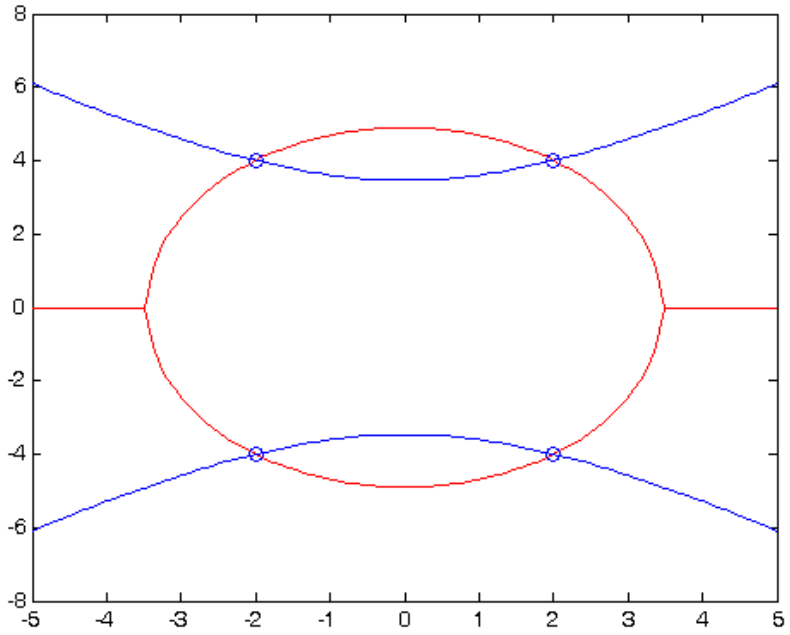


Multiple roots



```
x=linspace(-5,5);
plot(x,sqrt(24-2*x.^2),'r');hold on;
plot(x,-sqrt(24-2*x.^2),'r')
plot(x,sqrt(12+x.^2),'b')
plot(x,-sqrt(12+x.^2),'b')
```




8
6
4
2
0
-2
-4
-6
-8
-5
-4
-3
-2
-1
0
1
2
3
4
5

SOLVE

$$f_1(x_1, x_2) = x_1^2 - 2x_2^2 - 2 = 0$$

$$f_2(x_1, x_2) = x_1 x_2 - 2 = 0$$

Function evaluation

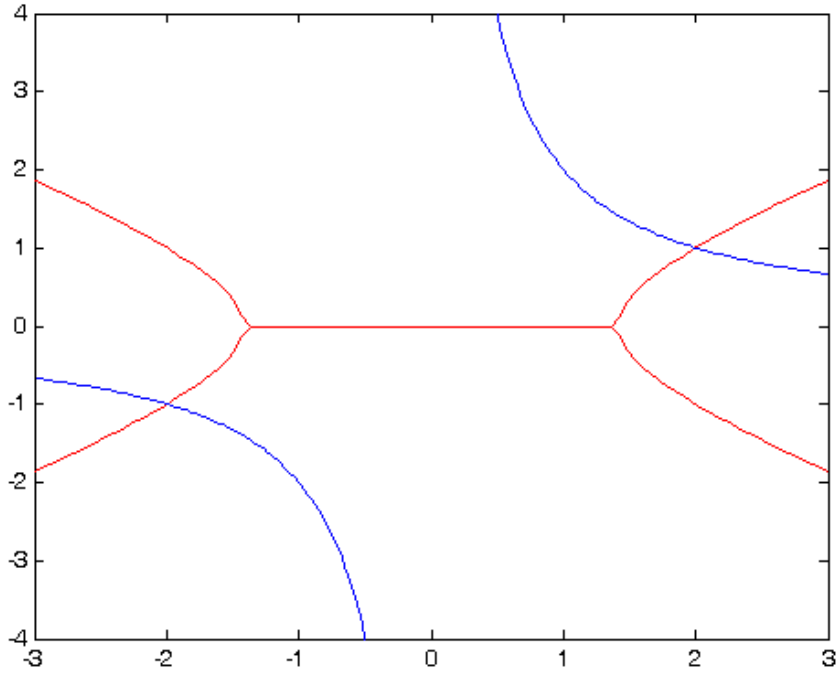
```
function F = myfun3(x)
   F(1) =  x(1).^2 -2* x(2)^2-2;
   F(2) =  x(1).*x(2) -2;
return
```

SOLVE

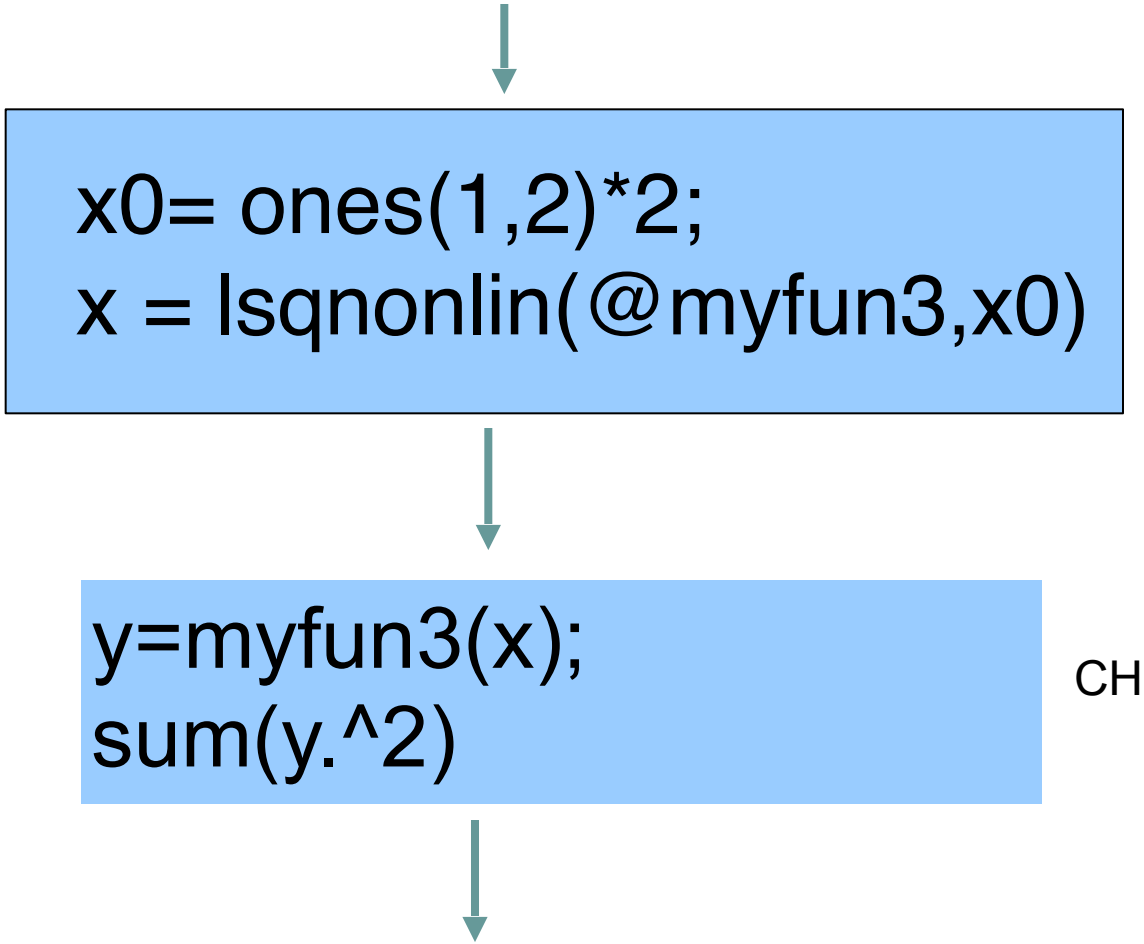
$$f_1(x_1, x_2) = x_1^2 - 2x_2^2 - 2 = 0$$

$$f_2(x_1, x_2) = x_1 x_2 - 2 = 0$$

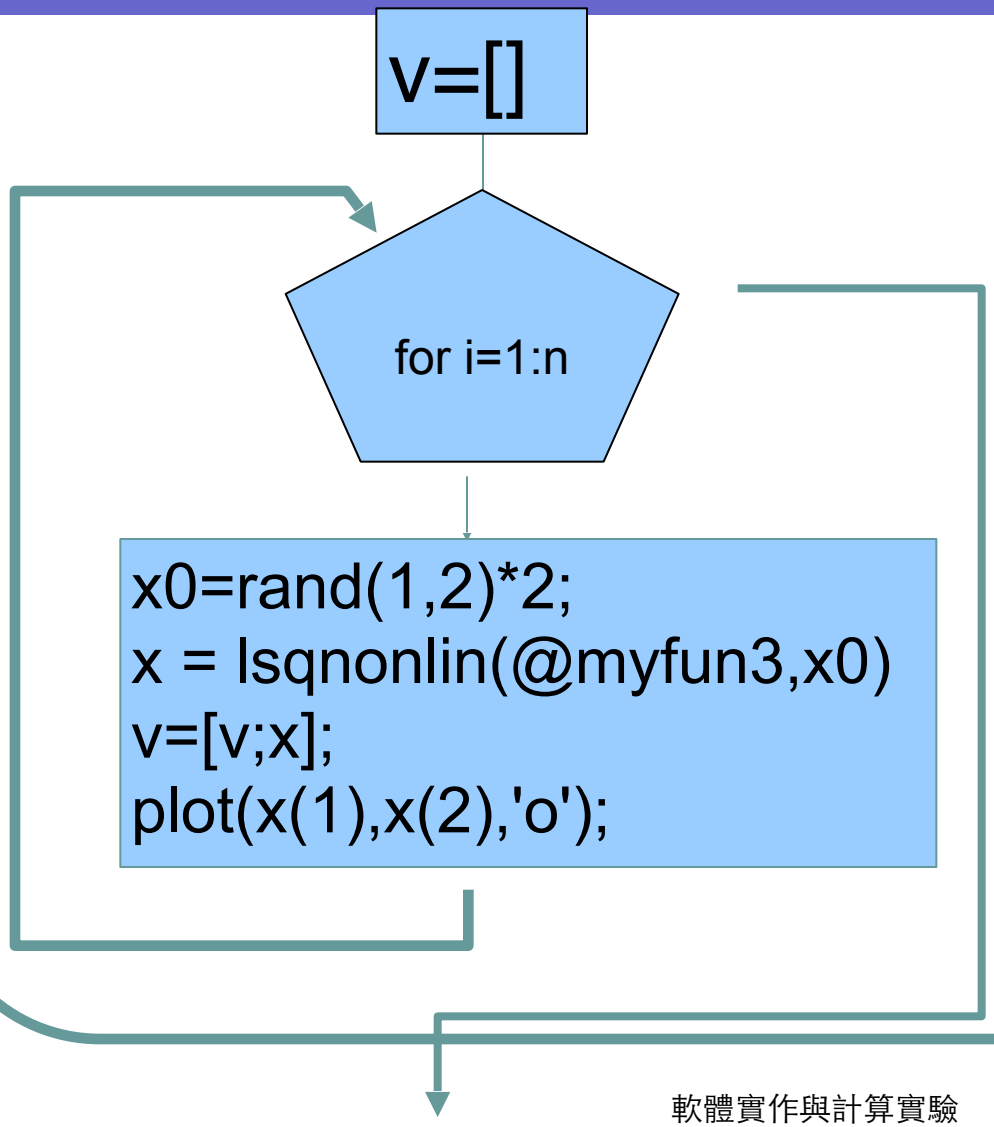
```
x=linspace(-3,3);
x=x(find(abs(x) > 0.1));
plot(x,sqrt((x.^2-2)/2),'r');hold on;
plot(x,-sqrt((x.^2-2)/),'r')
x=linspace(0.5,3);
plot(x,2./x,'b');
x=linspace(-0.5,-3);
plot(x,2./x,'b')
```



Find a zero

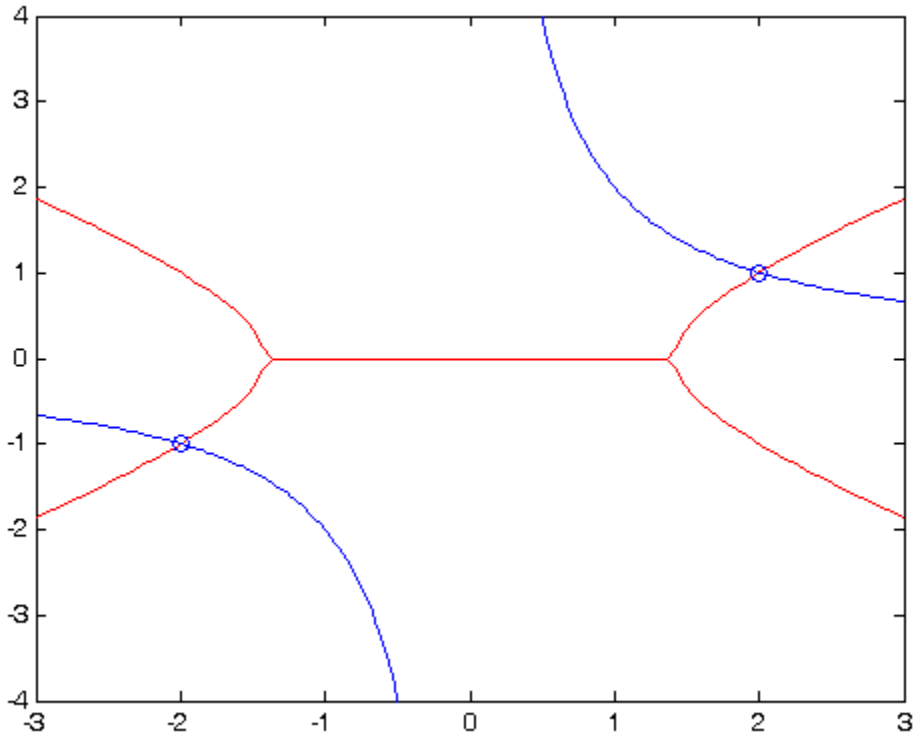


Multiple roots



demo_lsq_hyperbola.m

two_hyperbola.m



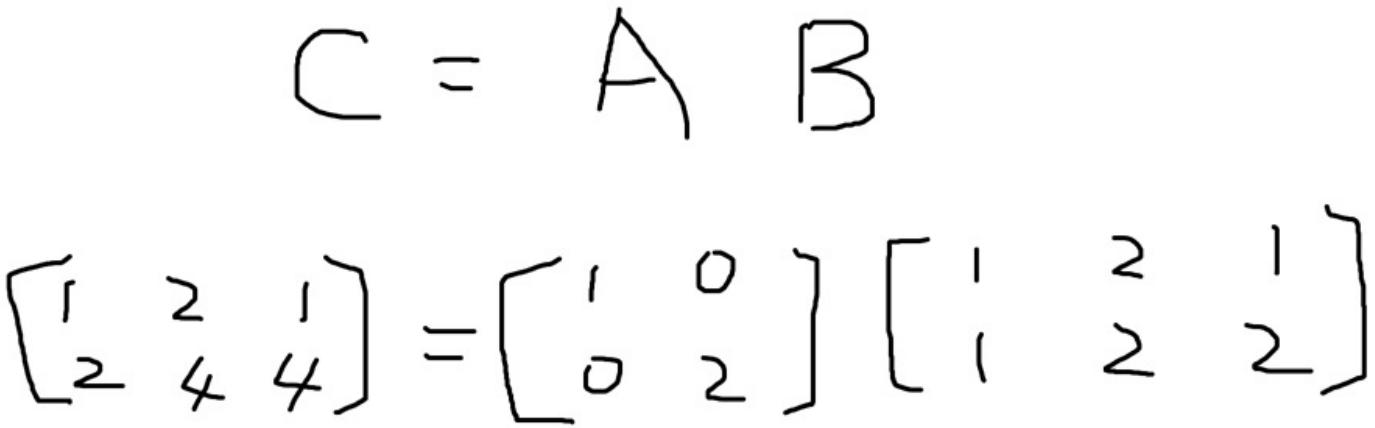
Nonlinear system f3

$$f_1(x) = exp(x_1) + x_2 * x_3 - 3 = 0$$
$$f_2(x) = \frac{x_1}{x_2} + x_3^2 - log(x_2) = 0$$
$$f_3(x) = \frac{x_1}{x_1+x_2+x_3} - sin(x_3) = 0$$

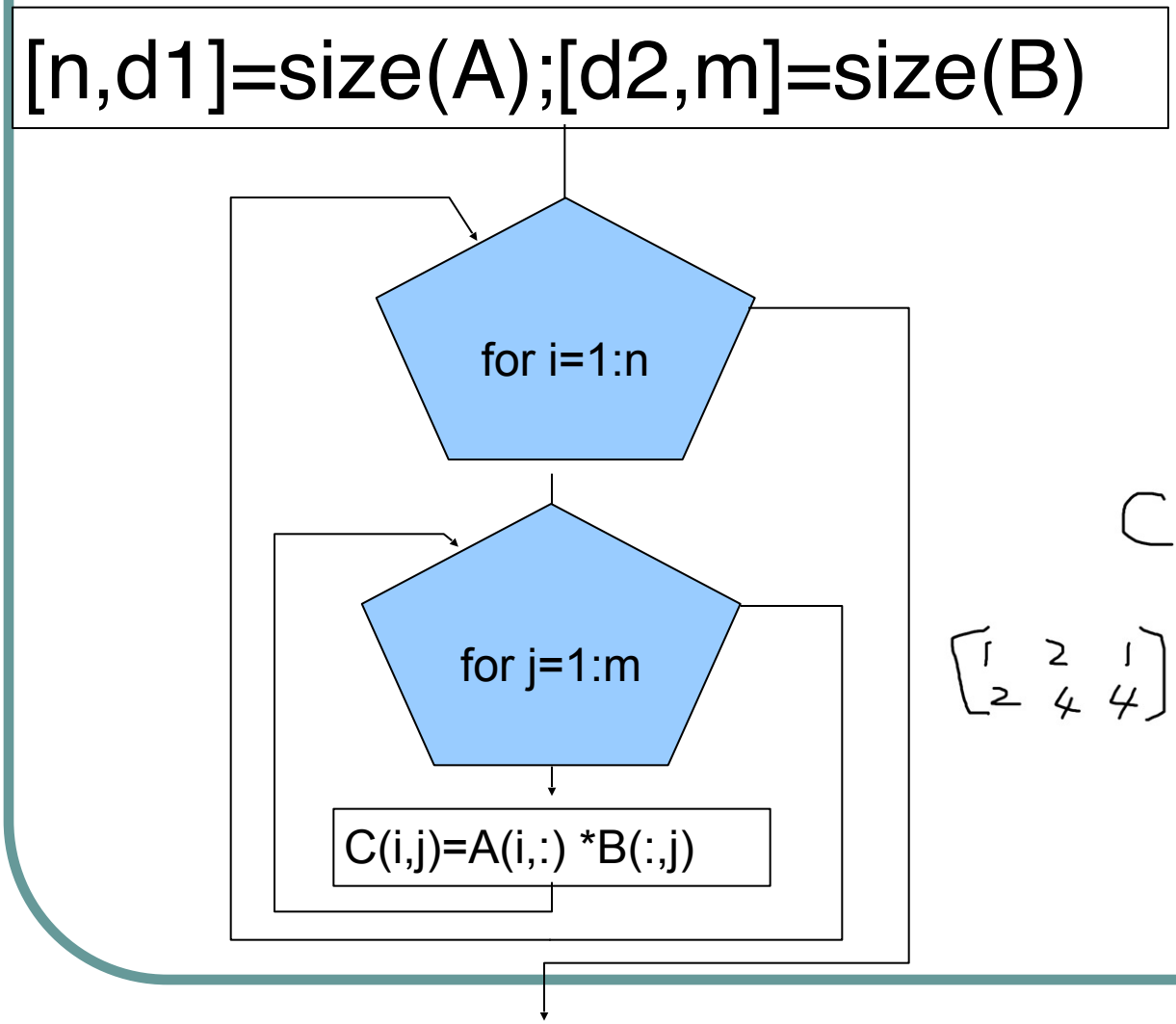
```
function F=f3(x)
F(1)=exp(x(1))+x(2)*x(3)-3;
F(2)=x(1)/x(2)+x(3).^2-log(x(2));
F(3)=x(1)/(x(1)+x(2)+x(3))-sin(x(3));
```

Matrix Multiplication

- C=A*B
- C(i,j)=A(i,:) *B(:,j) for all i,j



Nested for-loops



$$C = A\,B$$

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

```
[n,d1]=size(A);[d2,m]=size(B);
for i=1:n
for j=1:m
    C(i,j)=A(i, :) *B(:,j);
end
end
```

CPU time

```
A=rand(2000,100);B=rand(100,2000);
[n,d1]=size(A);[d2,m]=size(B);
tic
C=A*B;
toc
```

```
>> A=rand(2000,100);B=rand(100,2000);
[n,d1]=size(A);[d2,m]=size(B);
tic
C=A*B;
toc
```

```
tic
[n,d1]=size(A);[d2,m]=size(B);
for i=1:n
for j=1:m
    C(i,j)=A(i, :) *B(:,j);
end
end
toc
```

MATLAB Mobile

```
>> tic
[n,d1]=size(A);[d2,m]=size(B);
for i=1:n
for j=1:m
    C(i,j)=A(i, :) *B(:,j);
end
end
toc
```

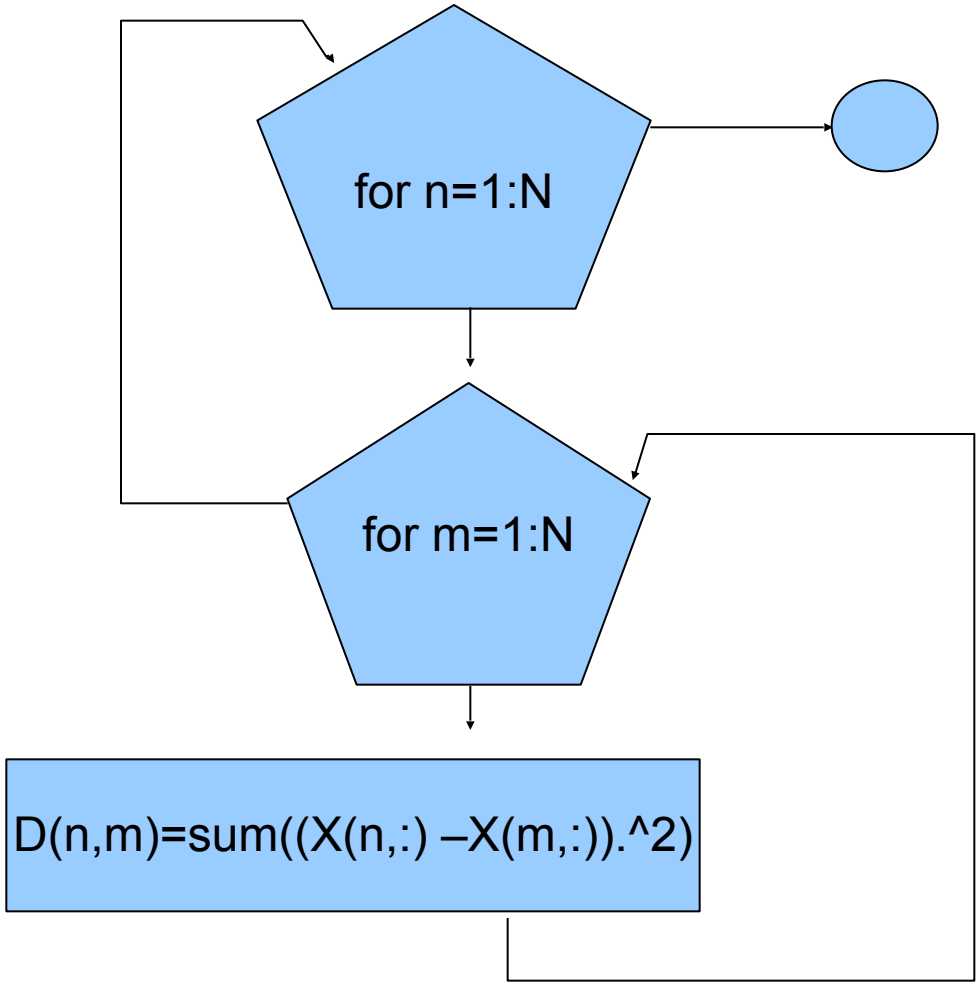
```
Elapsed time is 12.259549 seconds.
```

Calculation of Cross distances

- Given N points X: Nx200
- D: NxN
- D(i,j) denotes the distance between X(i,:) and X(j,:)
- Given X, find D

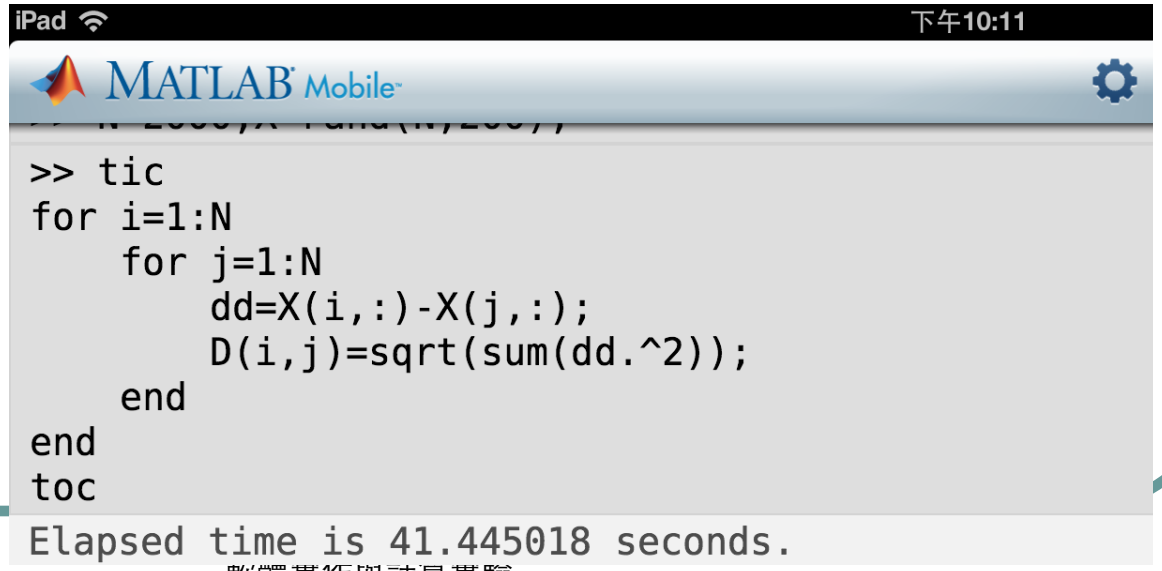
```
>> N=2000;X=rand(N,200);
```

Nested loops for cross distances



Matlab codes for nested codes

```
tic
for i=1:N
    for j=1:N
        dd=X(i,:)-X(j,:);
        D(i,j)=sqrt(sum(dd.^2));
    end
end
toc
```



- Straightforward implementation
- Nested for-looping
  - A loop within a loop
  - NxN calculations of distances
- Time consuming for large N and d

Vector codes

- How to calculate cross distances without using for-looping or while-looping ?
- Vector codes are loop-free
- Vector codes for cross distances can significantly improve efficiency against nested looping in computation

$$D_{i,j} = (x_i - x_j)(x_i^T - x_j^T)$$

$$= x_i x_i^T - 2 x_i x_j^T + x_j x_j^T$$

$$= A_{i,j} - 2B_{i,j} + C_{i,j}$$

D : cross distances between N points
Matrix D is decomposed to matrices A, B and C
A : elements in each row are identical
B : multiplication of matrix X and transpose of matrix X
C : elements in each column are identical

$$D_{ij} = (x_i - x_j)(x_i^T - x_j^T)$$

$$= x_i x_i^T - 2 x_i x_j^T + x_j x_j^T$$

$$= A_{ij} - 2B_{ij} + C_{ij}$$

```
tic
N=size(X,1);
A=sum(X.^2,2)*ones(1,N);
C=ones(N,1)*sum(X.^2,2)';
B=X*X';
D=sqrt(A-2*B+C);
toc
```

iPad

MATLAB Mobile

```
Elapsed time is 41.445018 seconds.
>> tic
N=size(X,1);
A=sum(X.^2,2)*ones(1,N);
C=ones(N,1)*sum(X.^2,2)';
B=X*X';
D=sqrt(A-2*B+C);
toc
Elapsed time is 1.712473 seconds.
```

Check equivalence

```
>> DD=sqrt(A-2*B+C);
>> sum(sum(abs(D-DD)))

ans =

     0
```

Cross distances between matrices X and Y

demo_distance2.m

```
 7 - for i=1:N
 8 -     for j=1:M
 9 -         dd=X(i,:)-Y(j,:);
10 -         D(i,j)=sqrt(sum(dd.^2));
11 -     end
12 - end
13 - ss2=cputime;
14 - A=sum(X.^2,2)*ones(1,M);
15 - C=ones(N,1)*sum(Y.^2,2)';
16 - B=X*Y';
17 - DD=sqrt(A-2*B+C);
18 - sum(sum(abs(DD-D)))
```

Constrained optimization

$$-2 \le x_1 \le 10$$

$$0 \le x_2 \le 10$$

Inequality constraint $x_2 \le x_1$

Nonlinear equality $x_1^2 + x_2^2 = 13$

Constrained optimization

$$\min_{x_1, x_2} (x_1^2 + x_2^2 - 13)^2$$

$$-2 \le x_1 \le 10$$

$$0 \le x_2 \le 10$$

$$x_2 \le x_1$$

Constraints

- Lower bound and upper bound
- Equality constraint
- Inequality constraint

Constraints

Lower bound $[-2,0] \leq [x_1, x_2]$

Upper bound $[x_1, x_2] \leq [10,10]$

Inequality constraints

$$x_2 \leq x_1$$

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq 0$$

Demo_conop2

demo_conop2.m

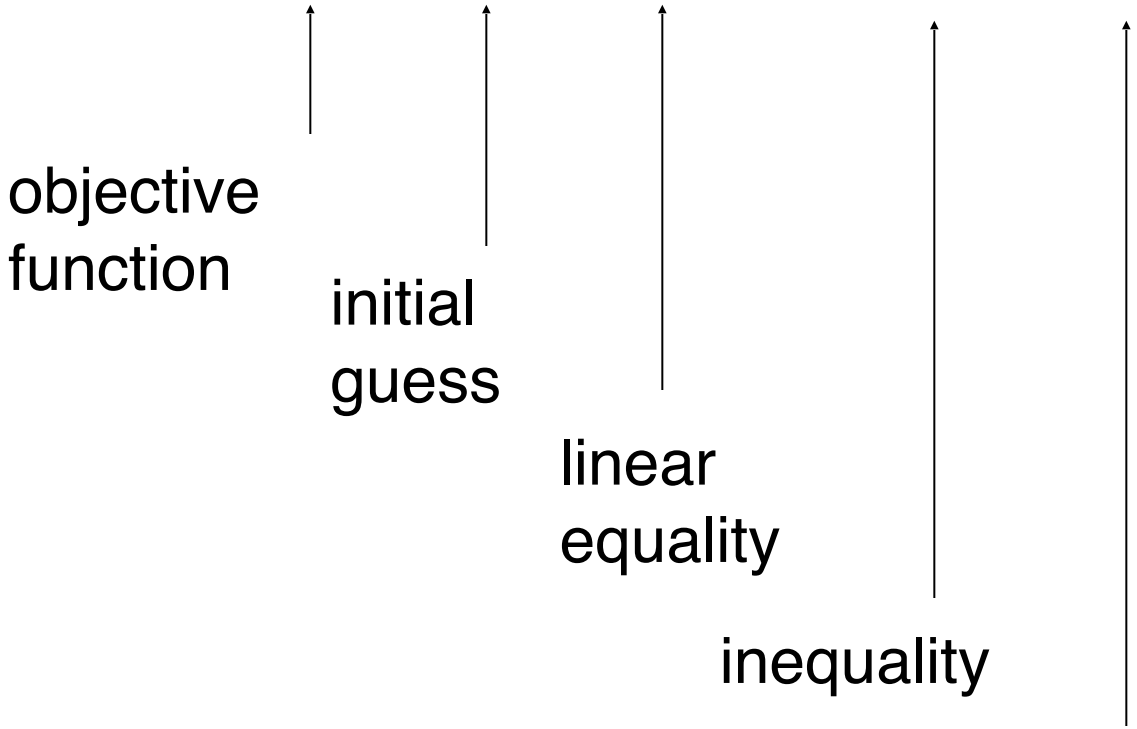
```
function demo_conop2()
lb =[-2 0];
ub =[10,10];
Aeq=[-1 1];beq=[0];
A=[];b=[];
x = fmincon(@fun,[1 1],A,b,Aeq,beq,lb,ub)
x(1).^2+x(2).^2
return
function y = fun(x)
y = (x(1).^2+x(2).^2-13).^2;
return
```

Objective function

```
function y = fun(x)
y = (x(1).^2+x(2).^2-13).^2;
return
```

Calling fmincon

```
x = fmincon(@fun,[1 1],A,b,Aeq,beq,lb,ub)
```



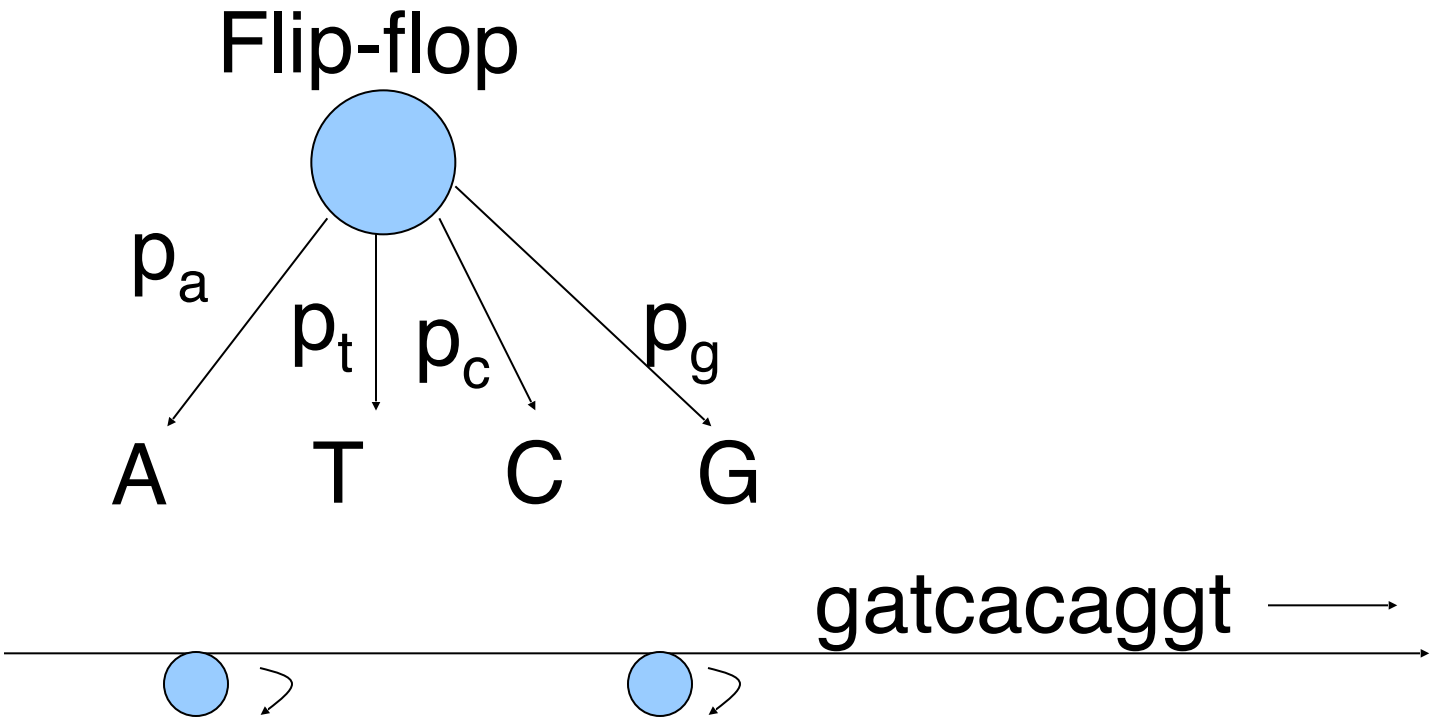
Random variable

- Sample space S={A,T,C,G}
- X is a discrete random variable with sample space S.

$$\Pr(X = A) = p_a \qquad \Pr(X = T) = p_t$$
$$\Pr(X = C) = p_c \qquad \Pr(X = G) = p_g$$

Generative model

$$p_a + p_t + p_c + p_g = 1$$

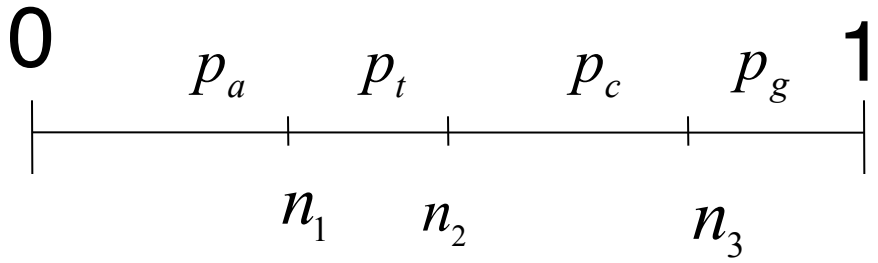


Partition

$$n_1 = p_a$$

$$n_2 = p_a + p_t$$

$$n_3 = p_a + p_t + p_c$$



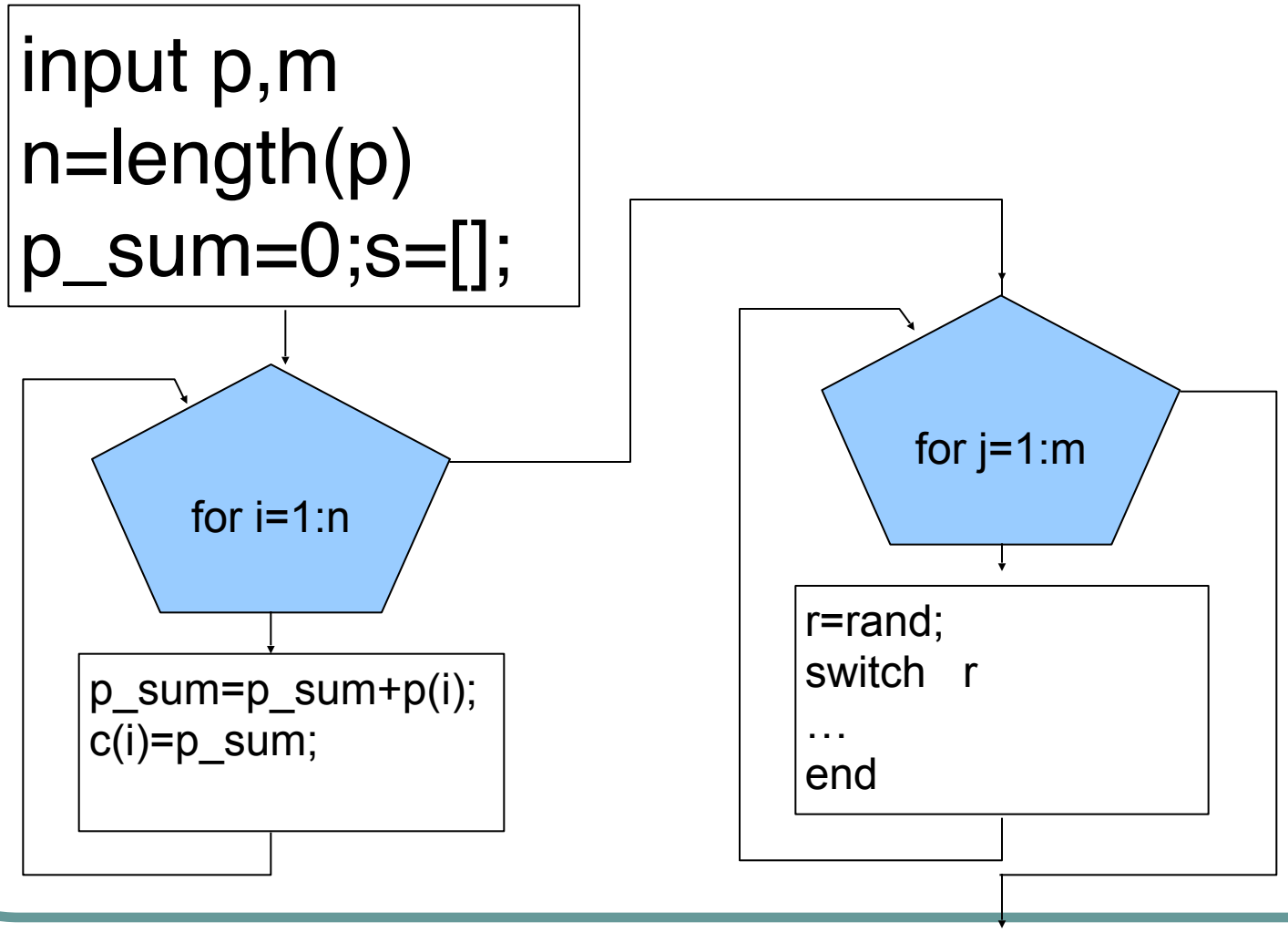
Knots n_1, n_2 and n_3 partition [0,1]
into four intervals respectively with lengths,
p_a, p_t, p_c and p_g

```
r=rand;
Tag= (r<=c(1))+ 2*(r<=c(2) & r > c(1))+3*(r<=c(3) & r > c(2))+4*(r<=c(4) & r > c(3))
switch Tag
case  1
        s=[s 'A'];
case 2
        s=[s 'T'];
case 3
        s=[s 'C'];
case 4
        s=[s 'G'];
end
```

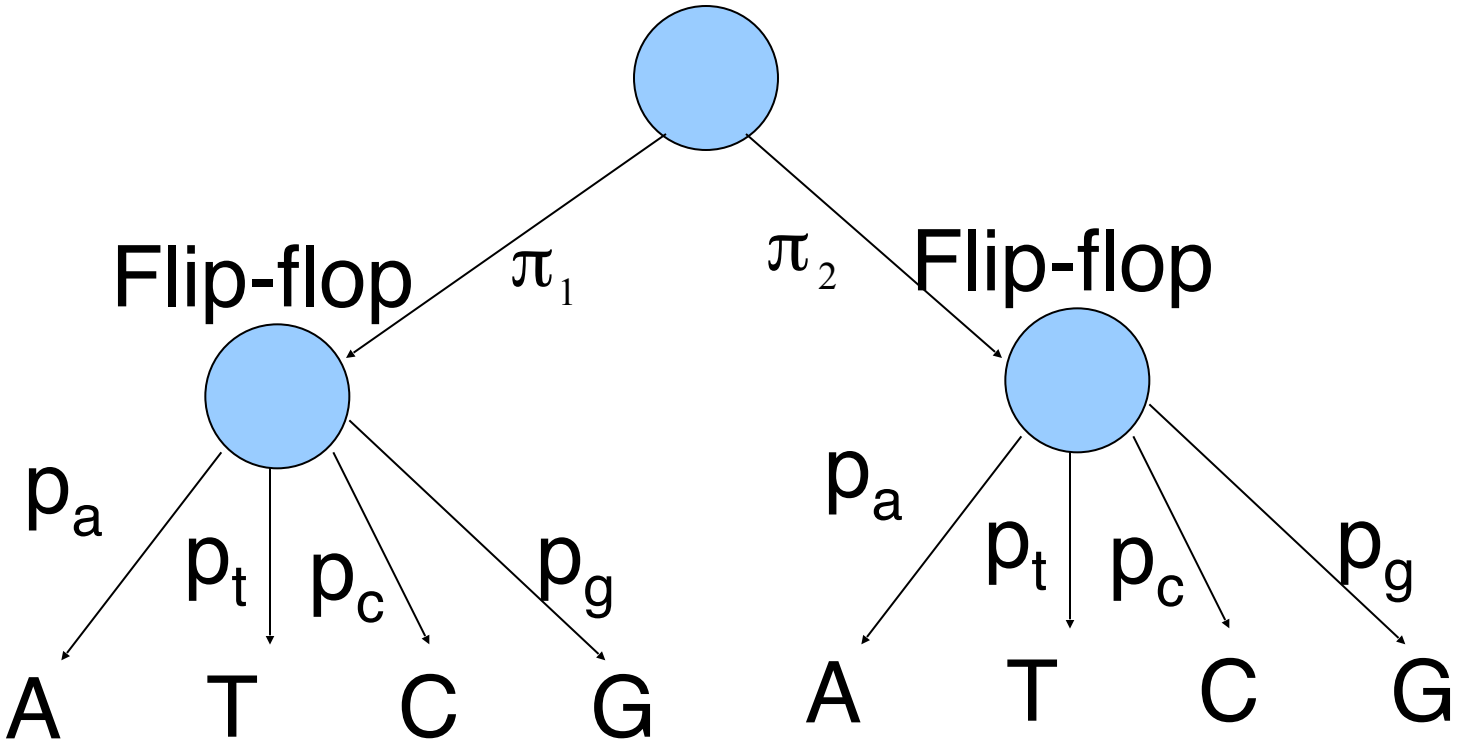
Sequence generation

- Emulation of a generative model for creation of an atcg sequence

FOR Loops


input p,m
n=length(p)
p_sum=0;s=[];
for i=1:n
p_sum=p_sum+p(i);
c(i)=p_sum;
for j=1:m
r=rand;
switch r
…
end

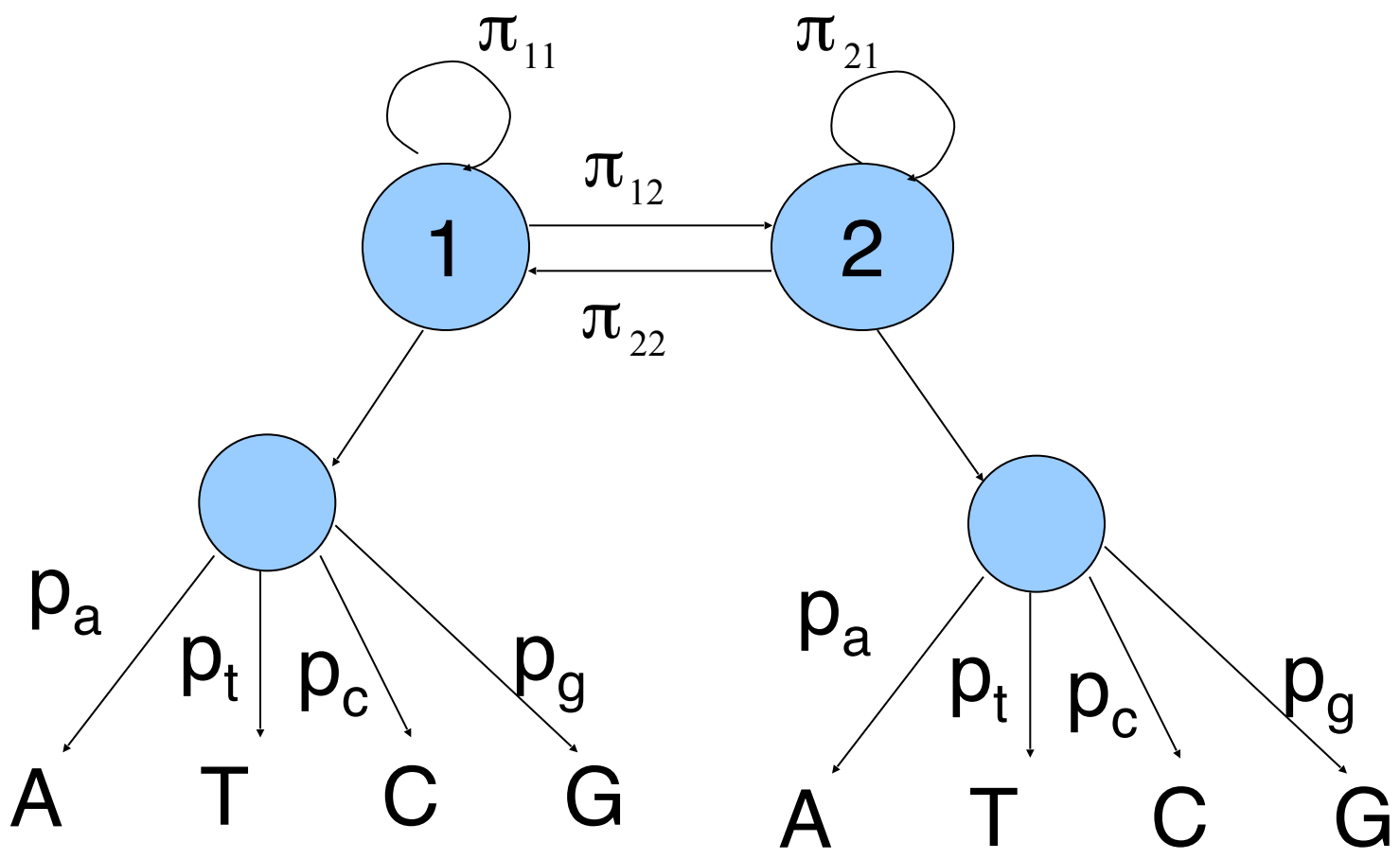
Mixture model



Mixture model

- According to probabilities, π_1 and π_2 each time one of two joined models is selected to generate a character
- Created characters are collected to form an atcg sequence

Hidden Markov model



Transition probability

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix}$$