Lecture 10 Vector codes

- Matlab compiler
- Vector codes
  - Hyper-plane fitting
  - Distances between points
  - Product of two matrices

Hanoi Tower Play Panel

PlayHanoiTower.m
PlayHanoiTower.fig

mcc

Use mcc instruction to compile a matlab function

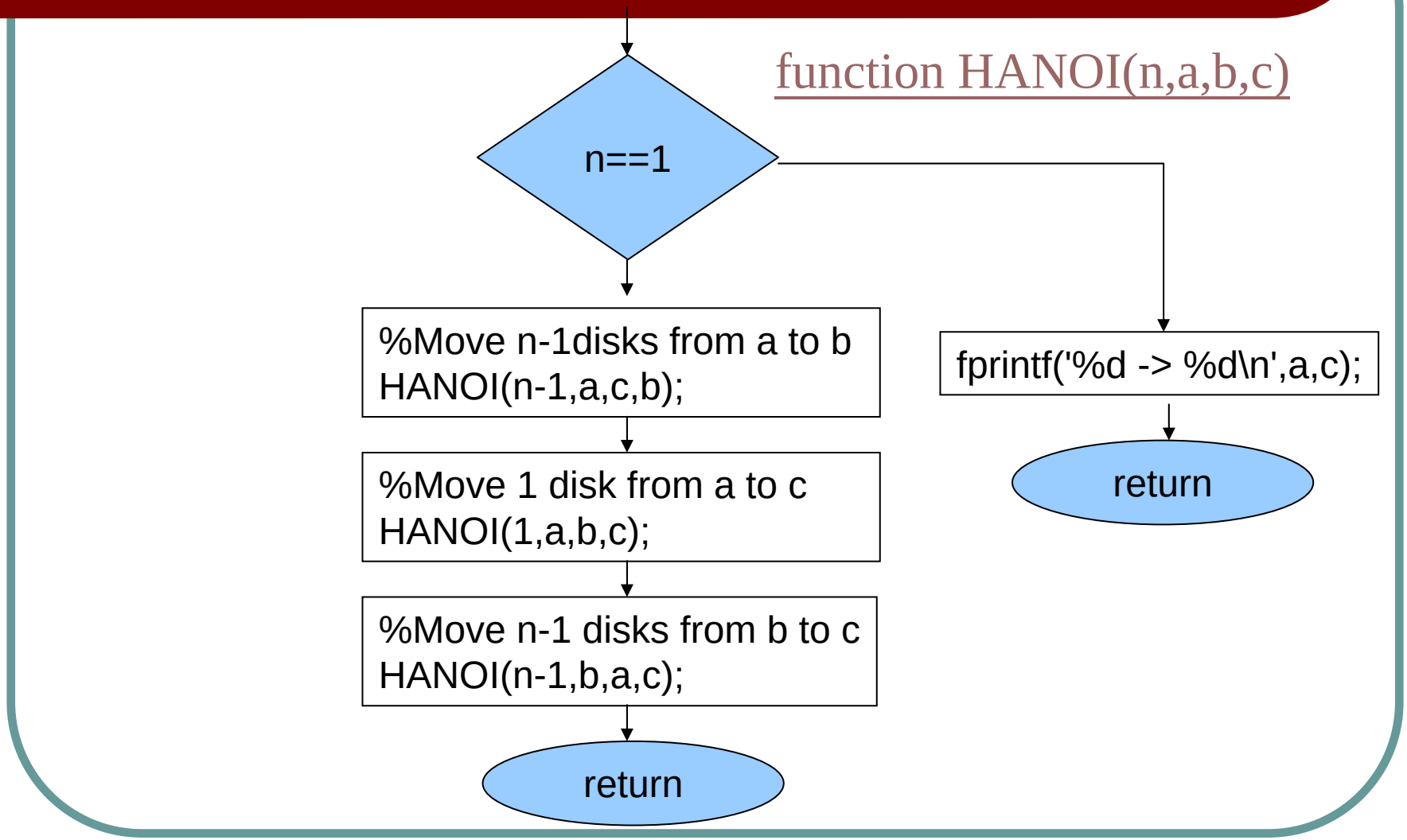
```
mcc –m PlayHanoiTower.m
```

HANOI TOWER

HANOI.m

```
1  function HANOI(n,a,b,c)
2  if n==1
3      fprintf('%d -> %d\n',a,c);
4  elseif n>1
5      HANOI(n-1,a,c,b);
6      HANOI(1,a,b,c);
7      HANOI(n-1,b,a,c);
8  end
9  return
```

Flow chart: move n disks from a to c



Hyper-plane fitting

Sampling

```
d=5; N=1000;
x=rand(N,d)*2-1;
a= rand(d,1);b=rand(1,1);
y=x*a+rand(N,1)*0.2-0.1;
```

Mesh

plot_2D.m

Minimization

$$E(\mathbf{a}, b) = \sum_i (y_i - \mathbf{a}^{\mathrm{T}}\mathbf{x}_i - b)^2$$

$$\mathbf{a} = [a_1, \ldots, a_d]^T$$

$$\mathbf{x}_i = [x_{i1}, \ldots, x_{id}]^T$$

Derivation of normal equations

$$\frac{\partial E}{\partial a_m} = 0 \text{ for m} = 1,..., \text{d}$$

$$\frac{\partial E}{\partial b} = 0$$

Derivation

$$E(\mathbf{a}, b) = \sum_i (y_i - \mathbf{a}^{\mathrm{T}}\mathbf{x}_i - b)^2$$

$$\frac{\partial E}{\partial a_m} = -\sum_i (y_i - \mathbf{a}^{\mathrm{T}}\mathbf{x}_i - b)x_{im} = 0$$

$$\Rightarrow$$

$$\sum_i (\mathbf{a}^{\mathrm{T}}\mathbf{x}_i + b)x_{im} = \sum_i y_i x_{im}$$

$$\sum_i (\sum_n a_n x_{in} + b)x_{im} = \sum_i y_i x_{im}$$

Derivation

$$\sum_i (\sum_n a_n x_{in} + b) x_{im} = \sum_i y_i x_{im}$$

$$\Rightarrow$$

$$\sum_n (\sum_i x_{in} x_{im}) a_n + (\sum_i x_{im}) b = \sum_i y_i x_{im}$$

Derivation

$$E(\mathbf{a}, b) = \sum_i (y_i - \mathbf{a}\mathbf{x}_i - b)^2$$

$$\frac{\partial E}{\partial b} = -\sum_i (y_i - \mathbf{a}\mathbf{x}_i - b) = 0$$

$$\Rightarrow$$

$$\sum_i (\mathbf{a}\mathbf{x}_i + b) = \sum_i y_i$$

$$\sum_i (\sum_n a_n x_{in} + b) = \sum_i y_i$$

Derivation

$$\sum_i \left(\sum_n a_n x_{in} + b\right) = \sum_i y_i$$

$$\Rightarrow$$

$$\sum_n \left(\sum_i x_{in}\right) a_n + Nb = \sum_i y_i$$

Normal equations

$$\sum_n \left(\sum_i x_{in} x_{im}\right) a_n + \left(\sum_i x_{im}\right) b = \sum_i y_i x_{im}$$

$$\sum_n \left(\sum_i x_{in}\right) a_n + Nb = \sum_i y_i$$

d=3

$$\begin{pmatrix} \sum_i x_{i1}x_{i1} & \sum_i x_{i2}x_{i1} & \sum_i x_{i3}x_{i1} & \sum_i x_{i1} \\ \sum_i x_{i1}x_{i2} & \sum_i x_{i2}x_{i2} & \sum_i x_{i3}x_{i2} & \sum_i x_{i2} \\ \sum_i x_{i1}x_{i3} & \sum_i x_{i2}x_{i3} & \sum_i x_{i3}x_{i3} & \sum_i x_{i3} \\ \sum_i x_{i1} & \sum_i x_{i2} & \sum_i x_{i3} & N \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix} = \begin{pmatrix} \sum_i y_i x_{i1} \\ \sum_i y_i x_{i2} \\ \sum_i y_i x_{i3} \\ \sum_i y_i \end{pmatrix}$$

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix}$$

Function Ce(x,y)

A. Head: function C=Ce(x,y)
B. Body:
   a. [N,d]=size(x);
   b. Set C to zeros(d+1,d+1)
   c. for m=1:d
      for n=1:d
         for i=1:N
         C(m,n) = C(m,n) + x(i,m)*x(i,n)
   d. for m=1:d
         for i=1:N
          Add C(m,d+1) = C(m,d+1) + x(i,m)*y(i)
   e. For m=1:d
         for i=1:N
         C(d+1,n) = C(d+1,n) + x(i,n)
   f. C(d+1,d+1) = N

Scalar code

Ce.m

Vector code I

```
A=[x ones(N,1)];
C=A'*A;e=A'*y;
u=inv(C)*e;
```

Vector code I

```
A=[x ones(N,1)];
C=A'*A;e=A'*y;
u=C/e;
```

Demo_hp

demo_hp.m

d=100; N=1000

absolute difference = 0.000
cpu time: 0.234 0.047 in sec
time ratio- scalar to vector : 5.000

demo_hp(500,2000)

absolute difference = 0.000
cpu time: 12.391 0.656 in sec
time ratio- scalar to vector : 18.881

Vector code vs. sequential code

- For-loop
  - Inefficient
  - Time consuming
- Speed up
  - Vector code

Distances between high dimensional data points

MyDistance

- Head: D=MyDisrtance(x)
- Body:
  - [N,d]=size(x); D=zeros(N,N);
  - For i=1:N

    For j=1:N

        for k=1:d

            D(i,j)=D(i,j)+(x(i,k)-x(j,k))^2;

    D(i,j)=sqrt(D(i,j));

Scalar code

Mydistance.m

$$D_{ij} = (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i^T - \mathbf{x}_j^T)$$

$$= \mathbf{x}_i\mathbf{x}_i^T - 2\mathbf{x}_i\mathbf{x}_j^T + \mathbf{x}_j\mathbf{x}_j^T$$

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

```
A=sum(X.^2,2)*ones(1,N);
C=A';
B=X*X';
DD=A-2*B+C;
```

Execution file

demo_distance.m

demo_distance.exe

Example

```
>> demo_distance
Data size:1000
Dimension:20
cpu time: 14.922 0.125 in sec
time ratio- scalar to vector : 119.375
Absolute difference:0.000
Press a key to return
```

Matrix multiplication

- A, B denote matrices respectively with size pxq and qxr

$$\mathbf{a}_i : \text{the ith row of matrix A}$$

$$\mathbf{b}_j : \text{the jth column of matrix B}$$

$$[c_{ij}] : \text{the product of matrices A and B}$$

Scalar rule

$$c_{ij} = \sum_{k=1}^{q} a_{ik} b_{kj}$$

Matrix multiplication

1. Head: C=myMM(A,B)
2. Body
   a) [p,q]=size(A);[q,r]=size(B);C=zeros(p,r);
   b) for i=1:p
      for j=1:r
      for k=1:q
            C(i,j)=C(i,j)+ A(i,k)*B(k,j)

Example: matrix multiplication

```
p=500;q=1000;r=500;
A=rand(p,q);
B=rand(q,r);
for i=1:p
    for j=1:r
        sv=0;
        for k=1:q
            sv=sv+A(i,k)*B(k,j);
        end
        C(i,j)=sv;
    end
end
```

Vector code

- Matlab built-in instruction

```
C=A*B
```

Demo_product

demo_product.m

cputime

- Instruction: cputime

  demo_cputime.m

P=300

cpu time: 1.766 0.125 in sec
time ratio- scalar to vector : 14.125
absolute difference:0.000000

P=500

cpu time: 8.750 0.531 in sec
time ratio- scalar to vector : 16.471
absolute difference:0.000000

P=800

cpu time: 39.016 2.203 in sec
time ratio- scalar to vector : 17.709
absolute difference:0.000000