



The Waisman Laboratory
for Brain Imaging and Behavior



University of Wisconsin
**SCHOOL OF MEDICINE
AND PUBLIC HEALTH**

BMI/STAT-768 Statistical Methods for Medical Image Analysis

Moo K. Chung

Department of Biostatistics and Medical Informatics
Waisman Laboratory for Brain Imaging and Behavior
University of Wisconsin-Madison

www.stat.wisc.edu/~mchung

Introduction to MATLAB programming

Unit Objectives

1. Get familiar with Matlab
2. 5 components of programming
3. Combining 5 components in solving complex examples.

MATLAB

MATLAB (matrix laboratory) is a numerical computing environment and fourth-generation programming language.

A proprietary programming language developed by MathWorks.

MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

MATLAB

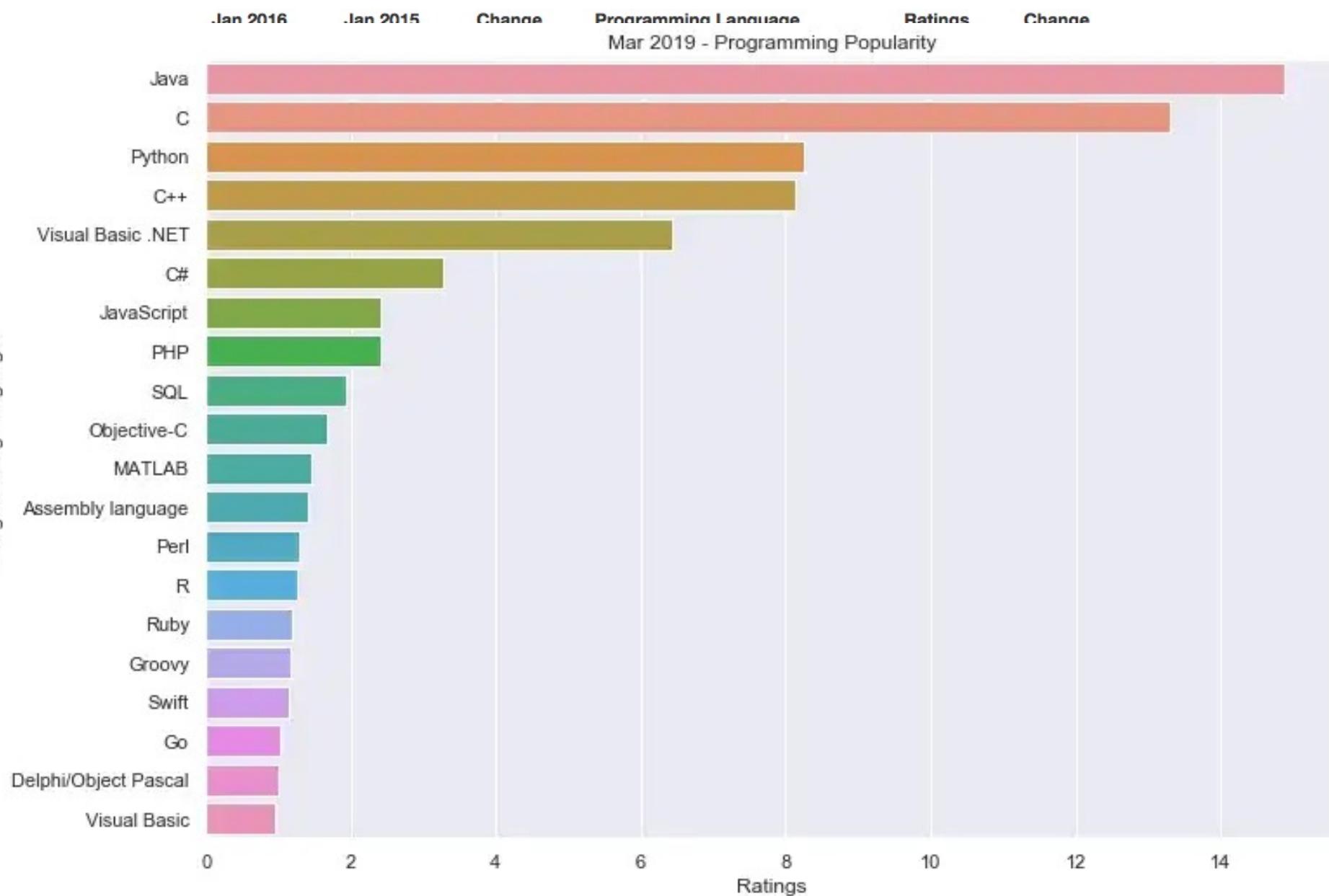
Dr. Cleve Moler, the chairman of the computer science department at the University of New Mexico, developed MATLAB in the late 1970s to give his students access to LINPACK without them having to learn Fortran.

LINPACK: linear algebra package written in C/Fortran. It is now widely used amongst scientists involved in image processing and medical imaging.

MATLAB use LINPACK for linear algebra computation

Mainly used in numerical analysis, scientific computation, image computation

Ranking of programming languages for data science

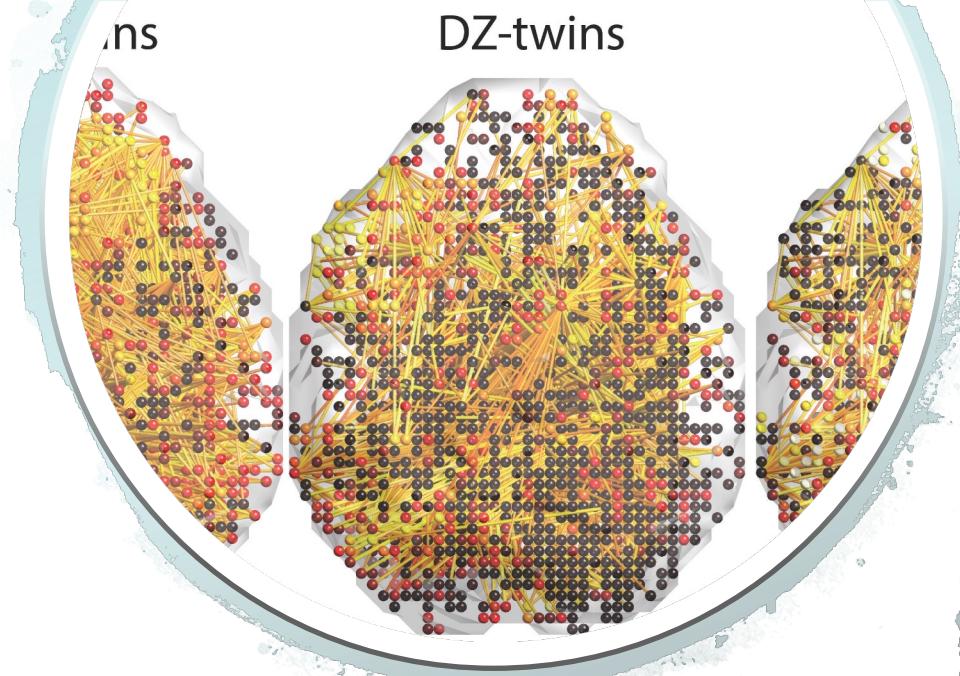


Interfacing with other languages

- MATLAB can call functions and subroutines written in the C, Fortran, Python, Perl, Java, ActiveX.
- A wrapper function is created allowing MATLAB data types to be passed and returned.
- The dynamically loadable object files created by compiling such functions are termed "MEX-files"
- MEX: MATLAB executable

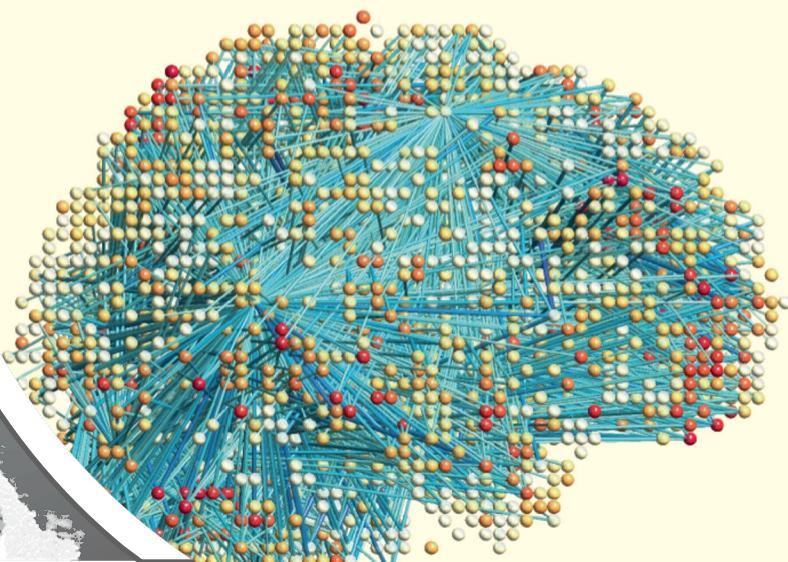
Syntax

- Mostly C-like scripting language.
- Variables are defined using the assignment operator, =. MATLAB is an inferred typed language because variables can be assigned without declaring their type.
- Variable type can be easily changed.
- Very forgiving to syntax errors. Multiple ways to write a function
→ **Fast prototyping language.**
- Baseline tool for image processing and image analysis.



Chung et al. 2015 arxiv.org/abs/1509.04771

BRAIN NETWORK ANALYSIS



Medical Image
Visualization in MATLAB

Advanced programming only needs 5 things

Almost all algorithms can be written using 1) variables, 2) functions, 3) iterations, 4) conditional statements and 5) random numbers.

Examples

- Optimization: Iterations on functions
- Matrix inversion: Iterations on matrix variables
- Partial differential equation: Iterations on a system of linear equations.

Question: *Can you come up with an algorithm that uses more than these five?*

Variables

```
>> x = 17
```

```
x =
```

```
17
```

```
>> x = 'hat'
```

```
x =
```

```
hat
```

```
>> x = [3*4, pi/2]
```

```
x =
```

```
12.0000 1.5708
```

```
>> y = 3*sin(x)
```

```
y =
```

```
-1.6097 3.0000
```

- Variable assignment is intuitive and easiest.
- Biggest source of programming error.

Iterations

Iteration is the act of repeating a process with the aim of approaching a desired result.

Each repetition of the process is also called an "iteration", and the results of one iteration are used as the starting point for the next iteration.

Recursion is the process of repeating items in a self-similar way in a nested fashion. Recursion is a special case of iteration.

Programming suggestions

- Avoid recursion in MATLAB if possible.
Difficult to read and debug codes.
- Avoid using “break” if possible.

break Terminate execution of WHILE or FOR loop. break terminates the execution of FOR and WHILE loops. In nested loops, break exits from the innermost loop only. *Difficult to read*.

Iterations

Iterate forward

```
total=0  
for i=1:100  
    total = total +i  
end
```

Iterate backward

```
total=0  
for i=100:-1:1  
    total = total +i  
end
```

Example 1.

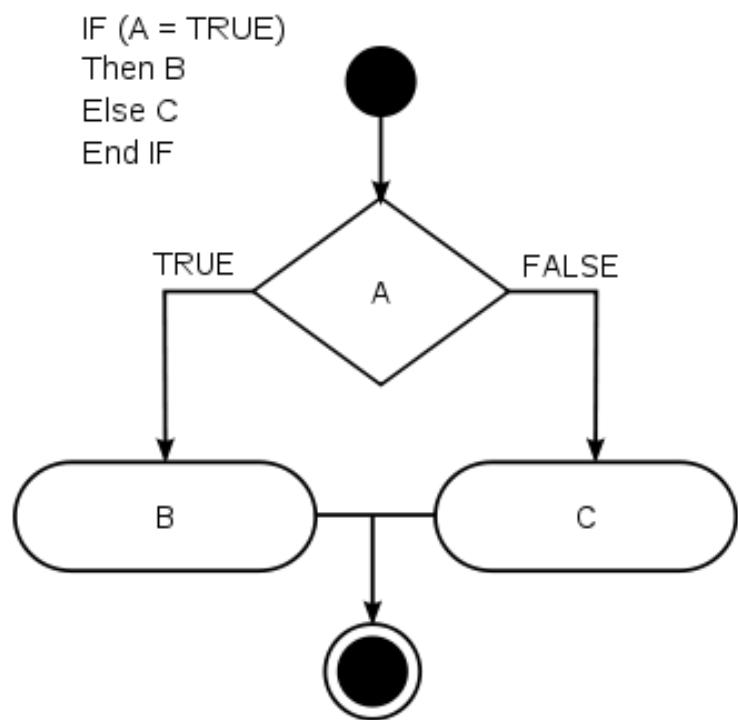
```
total=0  
for i=100:2:1  
    total = total +i  
end
```

Example 2.

```
total=0  
for i=100:-2:1  
    total = total +i  
end
```



Conditional statements



Conditional statements are features of a programming language, which perform different computations or actions depending on whether a programmer-specified boolean condition evaluates to true or false.

Conditional statements

```
>> 3>2
```

```
ans =
```

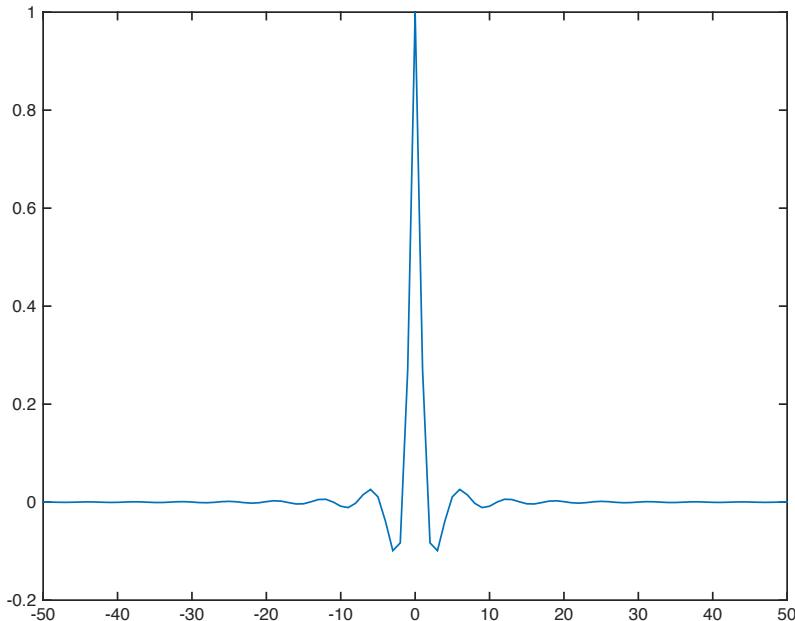
```
1
```

```
>> 3<2
```

```
ans =
```

```
0
```

Example. Finding maximum of a 1D function

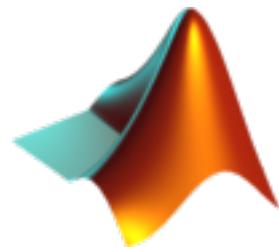


$$y = \frac{\cos x}{x^2 + 1}, x \in [-50, 50]$$

```
x=-50:50  
y=cos(x)./(x.^2+1)
```

```
n=length(x)  
ind=1;  
temp=y(ind)
```

```
for i=2:n  
    if y(i)>temp  
        ind=i;  
        temp=y(i);  
    end  
end
```



Functions

MATLAB supports lambda calculus by introducing function handles or function references, which are implemented in .m files.

When creating a MATLAB function, the name of the file should match the name of the function in the file.

Lambda calculus: (λ -calculus) is a formal system in mathematical logic for expressing computation based on function abstraction and application using variable binding and substitution. First introduced by mathematician Alonzo Church in the 1930s. Lambda calculus is a universal model of computation equivalent to a Turing machine.

Functions

Syntax

```
output = name(input)
```

First line of file name.m

```
function output = name(input)
```

```
function [xmax, mymax] = max2(x,y);
```

```
n=length(x)
```

```
ind=1;
```

```
temp=y(ind);
```

```
for i=2:n
```

```
    if y(i)>temp
```

```
        ind=i;
```

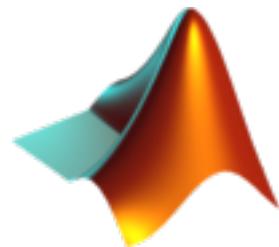
```
        temp=y(i);
```

```
    end
```

```
end
```

```
xmax=x(ind);
```

```
mymax=temp;
```



Random number generators

A random number generator (RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that can not be reasonably predicted better than by a random chance.

Almost all computational algorithms that can produce long sequences of apparently random results, which are in fact completely determined by a seed value. The entire seemingly random sequence can be reproduced if the seed value is known. This type of random number generator is often called a pseudorandom number generator.

Random number generators

Discrete uniform random numbers

$n=10$

$x = \text{unidrnd}(n, [1 \ 10])$

$x =$

4 2 3 7 5 4 9 6 6 10

Quiz 1. Can you generate continuous random numbers between [0 1] using `unidrnd`?

Random number generators

Poorman's continuous uniform distribution

$n=10000$

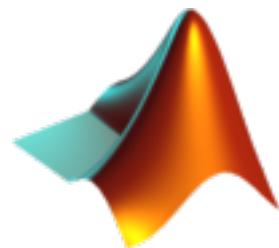
$x = \text{unidrnd}(n, [1 10])$

x/n

$x =$

0.6020 0.2630 0.6541 0.6893 0.7482
0.4506 0.0839 0.2290 0.9134 0.1524

Quiz 2. Is there a number in $[0, 1]$ you can't generate with the above procedure?



Histogram: probability distribution

A histogram is a graphical tool for understanding the distribution of numbers. It is an estimate of the probability distribution of the data and was first introduced by Karl Pearson.

To construct a histogram, the first step is to "bin" the range of values—that is, divide the entire range of values into a series of intervals—and then count how many values fall into each interval.

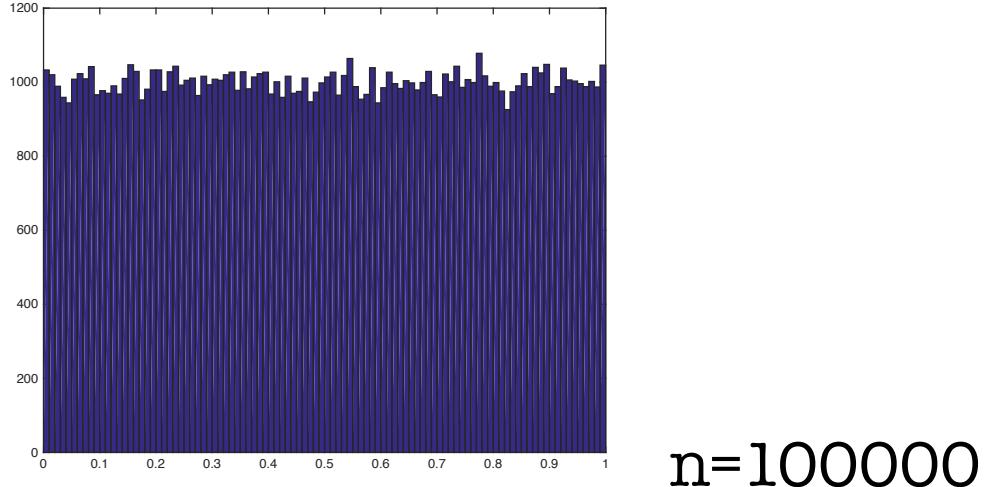
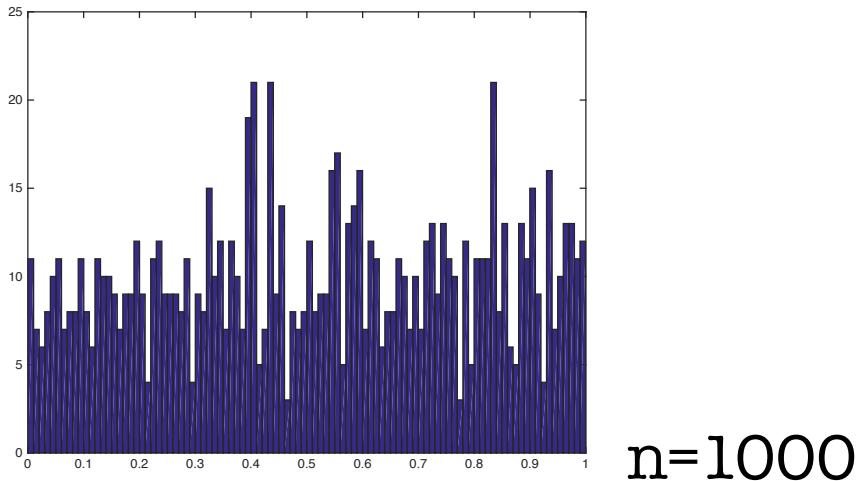
Quiz 3. *Write MATLAB codes for displaying the histogram of given data without using the built-in function hist.*

Random number generators

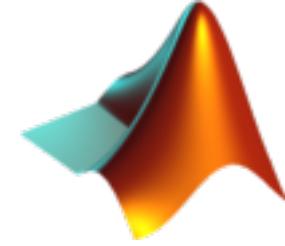
```
x=unifrnd(0,1, [1 10])
```

```
x =  
0.8258 0.5383 0.9961 0.0782 0.4427  
0.1067 0.9619 0.0046 0.7749 0.8173
```

```
x=unifrnd(0,1, [1 n]);  
figure; hist(x,100)
```



Random number generators

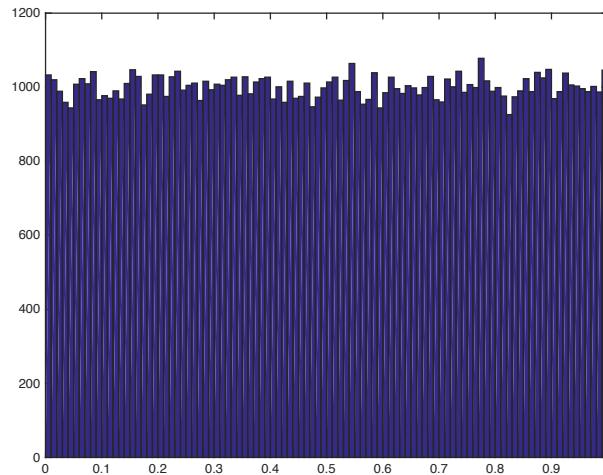


Quiz 3. Generate random numbers with bell-shaped histogram using the uniform random numbers.

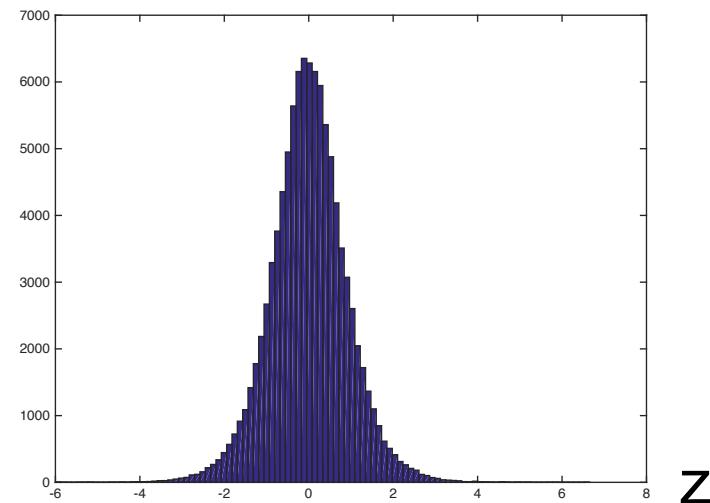
Fisher's z-transform:

$$z = \frac{1}{2} \log \frac{1+x}{1-x} = \operatorname{arctanh} x$$

Inverse hyperbolic arctan



x



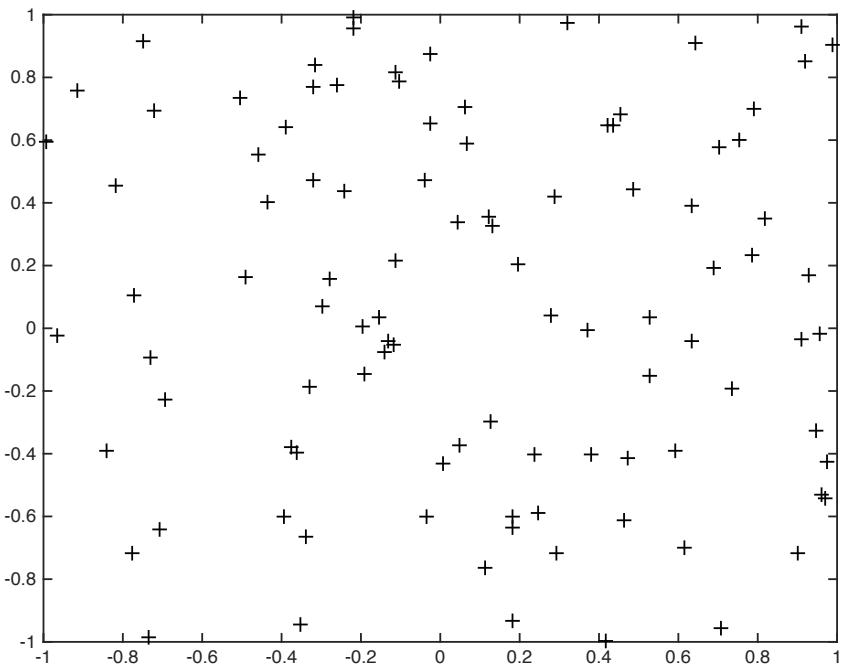
z

Advanced Example I

Sampling + random numbers + plotting

Data visualization

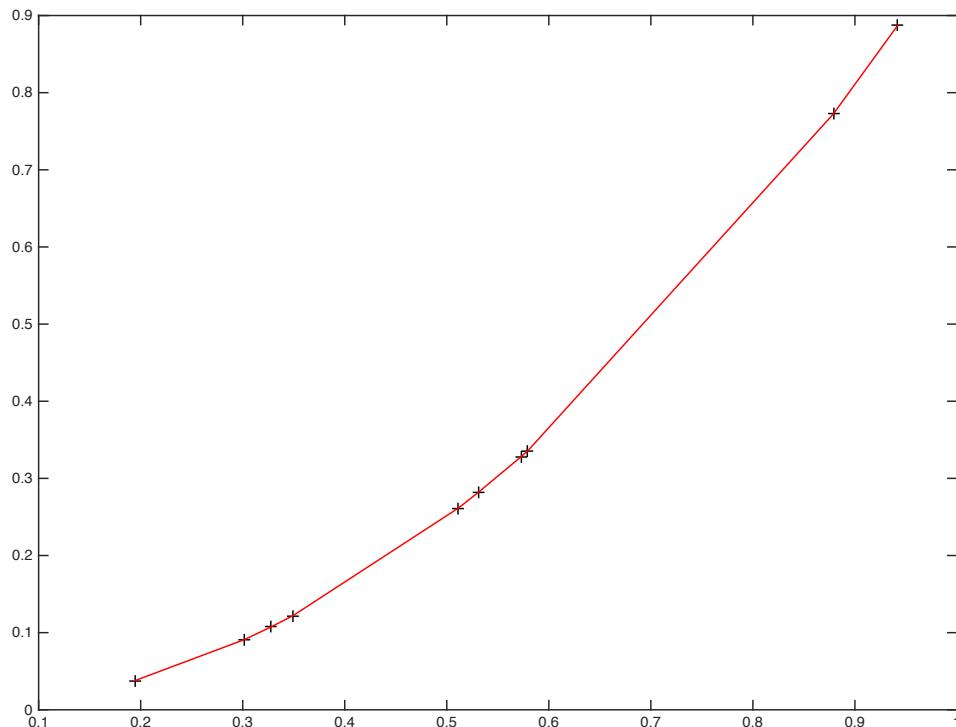
Plotting scatter points



```
x=unifrnd(-1,1,[1, 100])  
y=unifrnd(-1,1,[1,100])  
figure; plot(x,y,'+k')
```

Data visualization

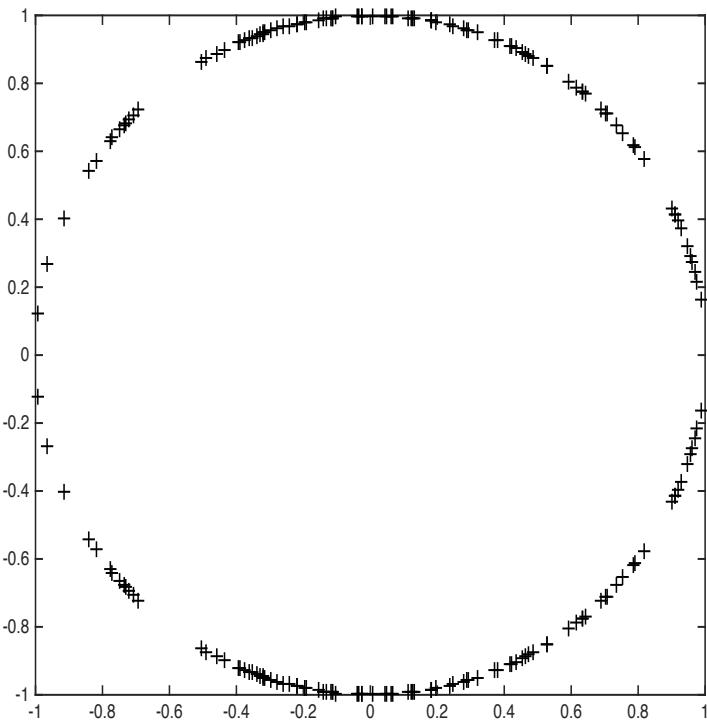
Plotting explicit curves



```
x=unifrnd(0,1,[1, 10])  
x=sort(x);  
y=x.^2;  
figure; plot(x,y,'+k')  
hold on; plot(x,y,'r')
```

Uniform sampling

Plotting explicit curves: a circle with radius 1

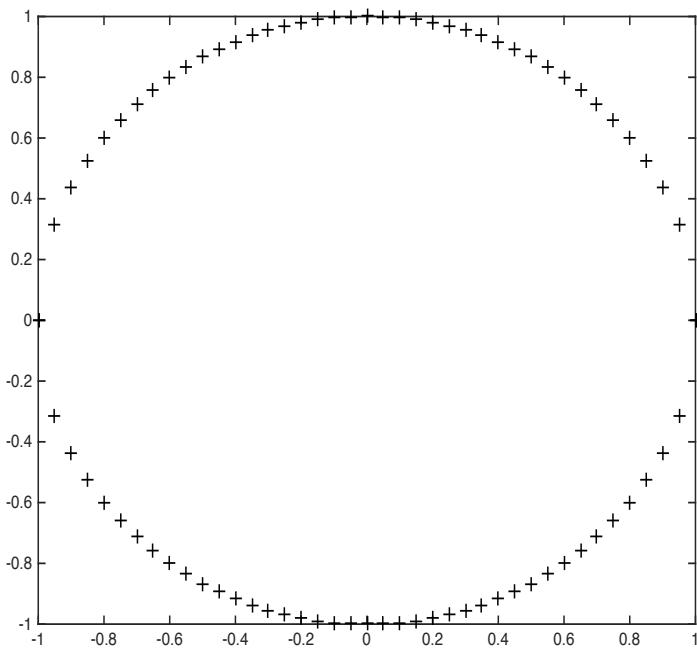


```
x=unifrnd(-1,1,[1, 100])  
y=sqrt(1-x.^2)  
figure; plot(x,y,'+k')  
hold on; plot(x,-y,'+k')
```

Quiz. Can you draw points uniformly sampled points on a circle?

Uniform sampling

Plotting explicit curves: a circle with radius 1

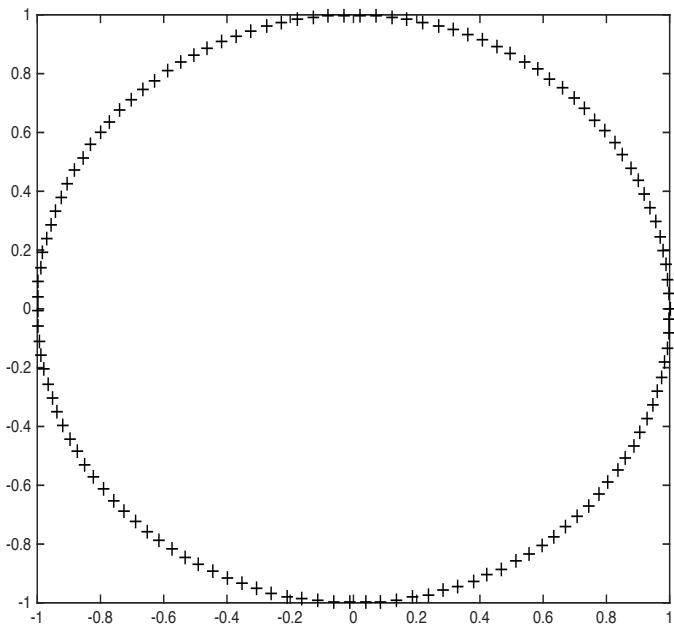


```
x=-1:0.05:1  
y=sqrt(1-x.^2);  
figure; plot(x,y,'+k')  
hold on; plot(x,-y,'+k')
```

Quiz 2. Still not uniform enough. Why?

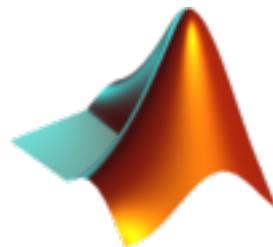
Uniform sampling

Plotting implicit curves: a circle with radius 1



```
theta=0:0.05:2*pi  
x=cos(theta)  
y=sin(theta)  
figure; plot(x,y,'+k')
```

Uniform sampling has to be done with respect to the manifold (parameter space) **Bertrand Russel's paradox (1889)**



Advanced Example 2

Convergence + Speeding up computation

Computation of Pi

pi=3.141592....

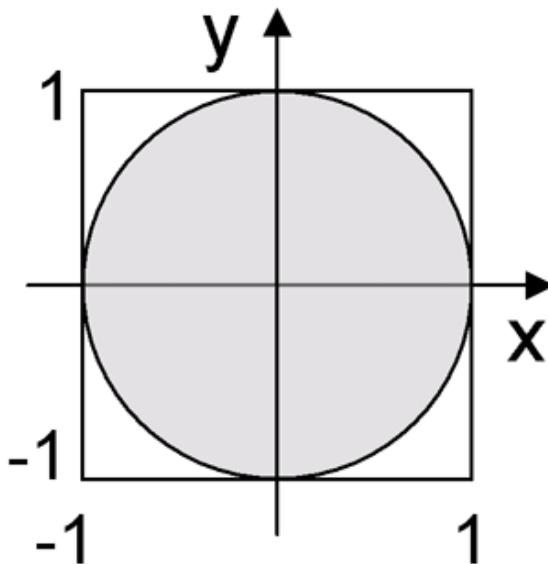
Using iterations and uniform random number generator, we can compute the digits of **pi** as accurately as possible.

How?

*Suggestion: Monte-Carlo simulation
(rejection sampling)*

Computation of Pi

Math fact: area of the circle = π



Goal: estimate the area of the circle

Algorithm:

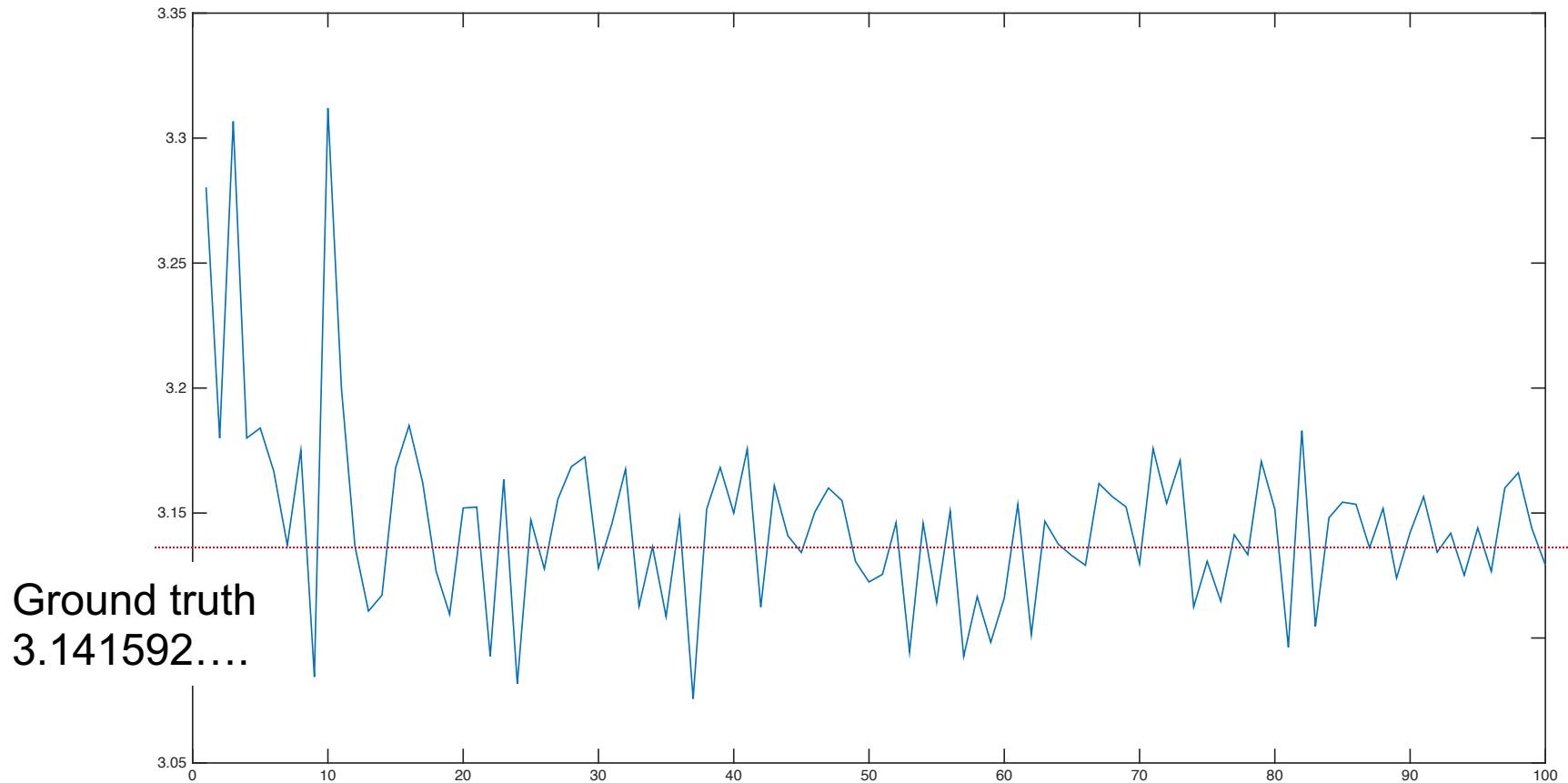
- 1) Uniformly sample n points (x, y) from $[0, 1]^2$.
- 2) Count the number of points within the circle. Call it p .
- 3) Area of circle = $4 * p/n = \pi$.

```
OurPi=[];
for n=1:10000
    n
    x=unifrnd(0,1,[1, n]);
    y=unifrnd(0,1,[1, n]);
    r=sqrt(x.^2 + y.^2);
    p=length(find(r<=1));
    OurPi = [OurPi 4*p/n];
end
```

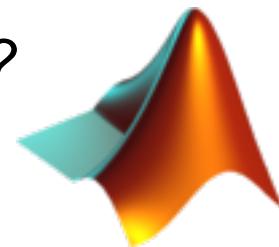
```
figure; plot(OurPi)
```

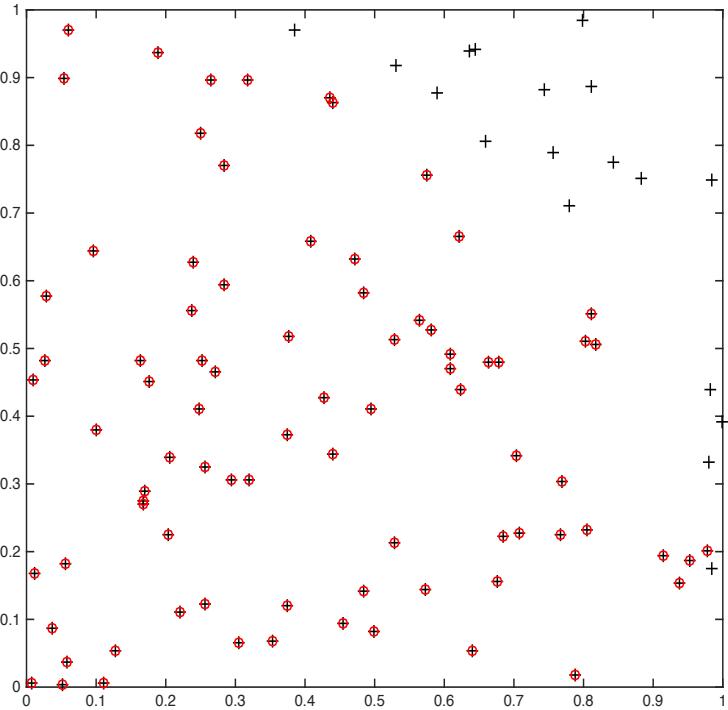
We don't need `sqrt` here. $r^2 \leq 1$ is equivalent to $r \leq 1$

Convergence rate of the Monte-Carlo simulation



Quiz. This dose not seem to converge. What's wrong?
Can you speed up the convergence?





>> ourPi

ourPi =

3.4000

figure; axis square;

n=100

x=unifrnd(0,1,[1, n]);

y=unifrnd(0,1,[1, n]);

r=sqrt(x.^2 + y.^2);

figure; plot(x,y,'+k')

p=0

for i=1:n

if r(i)<=1

p=p+1;

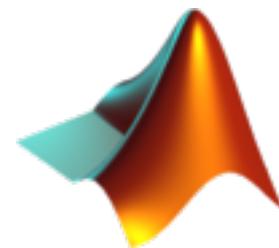
hold on;

plot(x(i),y(i),'or')

end

end

ourPi = 4 * p/n



More efficient code

```
OurPi=[];
x=unifrnd(0,1,[1, 10000]);
y=unifrnd(0,1,[1, 10000]);
r=x.^2 + y.^2;

for n=100:100:10000
temp=r(1:n);
p=length(find(temp<=1));
    OurPi = [OurPi 4*p/n];
end

figure; plot(OurPi)
```

Random numbers
are generated once
outside of the loop.

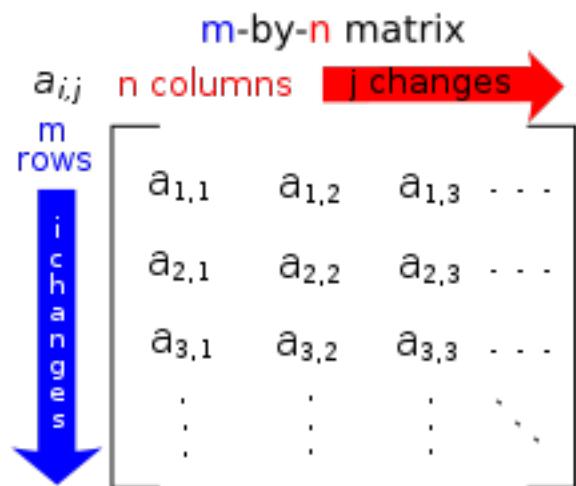
If possible, do **NOT**
use loop in MATLAB
programming.

Advanced Example 3

Matrix computation + least squares estimation

A matrix is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns.

m by n matrix $A_{m \times n} = (a_{ij})$

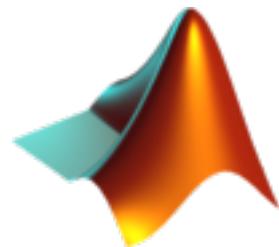


$m=3$; $n=2$;
 $A = \text{zeros}(m,n)$

$A =$

$$\begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix}$$

Quiz. What will happen if you run $A(5,5)=1$?



Matrix algebra

$A+B$ $>>A+B$

AB $>>A*B$

A^2 $>>A^2$ or $A*A$

Hadamard product $A = (A_{ij}), B = (B_{ij})$

$$\mathbf{A} \circ \mathbf{B} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} \circ \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1m} \\ B_{21} & B_{22} & \cdots & B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n1} & B_{n2} & \cdots & B_{nm} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} & A_{12}B_{12} & \cdots & A_{1m}B_{1m} \\ A_{21}B_{21} & A_{22}B_{22} & \cdots & A_{2m}B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1}B_{n1} & A_{n2}B_{n2} & \cdots & A_{nm}B_{nm} \end{pmatrix}$$

Frobenius norm

$\mathbf{A}.*\mathbf{B}$

$$\langle A, B \rangle = \sum_{i,j} A_{ij}B_{ij}$$

Quiz. Write 1-line MATLAB code for computing Frobenius norm.

Kronecker product

Can be used to tile matrices with repeating patterns

$$A = (A_{ij})$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \cdots & A_{1n}\mathbf{B} \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \cdots & A_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}\mathbf{B} & A_{m2}\mathbf{B} & \cdots & A_{mn}\mathbf{B} \end{pmatrix}.$$

```
>>kron(A,B)
```

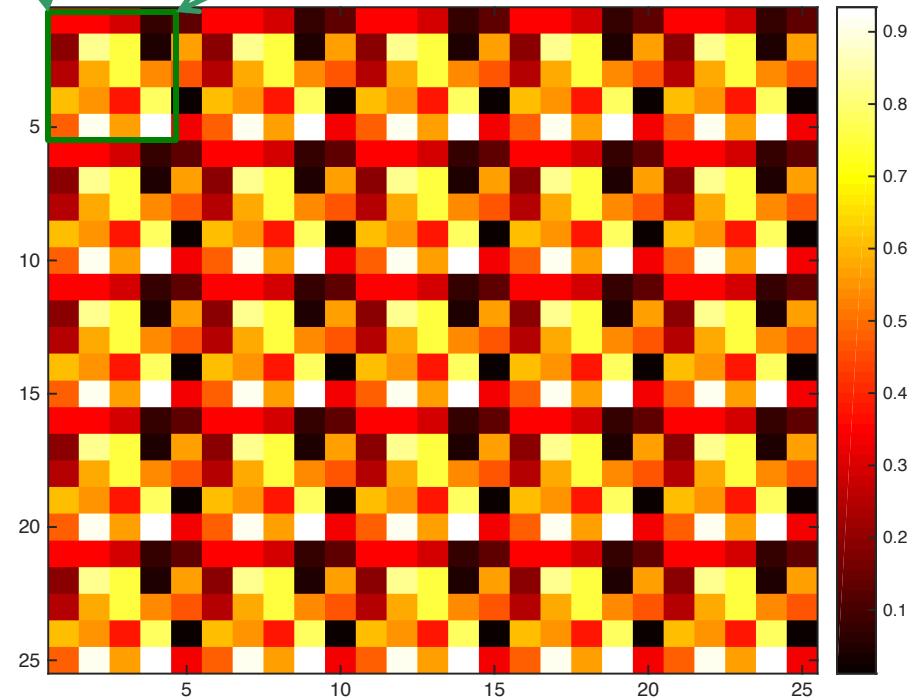
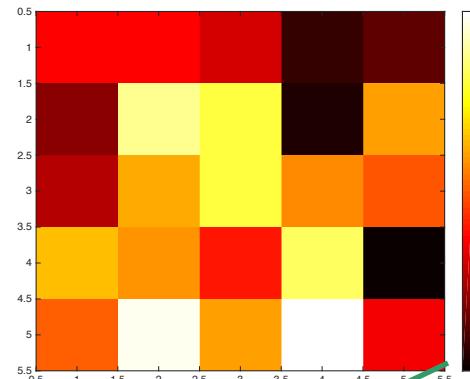
Kroneker matrix product

```
B=unifrnd(0,1,[5 5])
```

```
figure  
imagesc(B)  
colorbar  
colormap('hot')
```

```
A=ones(5,5);  
C=kron(A,B);
```

```
figure  
imagesc(C)  
colorbar  
colormap('hot')
```



Quiz: Can we extend this concept to tensor product?

Matrix A is called invertible (nonsingular) if there exists an n -by- n square matrix B such that

$$AB = BA = I_n = \text{eye}(n).$$

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

```
>>B=inv(A)
```

assuming $\det(\mathbf{A})$ is not zero. Matrix inversion is done using Gaussian elimination: $O(n^3)$. The lowest bound: $O(n^2 \log n)$.

Quiz. How to invert really large matrices? You cannot!

Matrix equation

Determined system

$$x+z = 4$$

$$-x+y+z = 4$$

$$x -y + z = 2$$

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

$$b = [4 \ 4 \ 2]'$$

$$X = A \setminus b$$

$$X =$$

Backslash \ uses
Gaussian elimination

$$AX = b$$

$$X = A^{-1}b$$

1
2
3

$$X = \text{inv}(A) * b$$

Matrix equation

Over-determined system

$$x = 4$$

$$-x+y = 4$$

$$x-y = 2$$

Quiz. Mathematically there is no solution. So how to solve this problem?

Least squares estimation (LSE)

We will approximately solve

$$AX = b$$

by minimizing $\|AX-b\|^2$.

Even if there is no solution in $Ax=b$, there is an optimization solution.

The least squares method grew out of the fields of astronomy as mathematicians sought to provide solutions to the challenges of navigating the Earth's oceans during the Age of Exploration.

$$f(X) = (x-4)^2 + (-x+y-4)^2 + (x-y-2)^2 = (x-4)^2 + (x-y-2)^2$$

$$df/dx = 6x - 4y - 4 = 0$$

$$df/dy = -4x + 4y - 4 = 0$$



$$A = [6 \ -4 \\ -4 \ 4]$$

$$b = [4 \ 4]'$$

$$X = A \setminus b$$

$$X =$$

4.0000

5.0000

Least squares method

How to minimize $\|AX-b\|^2$?

Let $f(X) = \|AX - b\|^2$

Minimum is obtained when the matrix derivative is zero:

$$\frac{df(X)}{dX} = 0$$

The solution is given by $X = (A^T A)^{-1} A^T b$ assuming $A^T A$ is invertible. What if $A^T A$ is not invertible?

Quiz Find minimum for general 2x2 matrix.

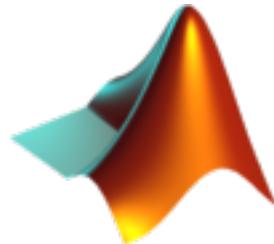
Moore-Penrose (pseudo) inverse

A pseudoinverse A^- of a matrix A is a generalization of the inverse matrix. The most widely known type of matrix pseudoinverse is the Moore–Penrose pseudoinverse, which was independently described by E. H. Moore in 1920, Arne Bjerhammar in 1951 and Roger Penrose in 1955.

If $A^T A$ is invertible, $A^- = (A^T A)^{-1} A^T$.

In general, A^- is defined as a matrix satisfying
 $AA^-A = A$, $A^-AA^- = A^-$
 AA^- and AA^- are symmetric.

```
>>pinv(A)
```



Matrix equation

Over-determined system

$$\begin{aligned}x &= 4 \\ -x+y &= 4 \\ x-y &= 2\end{aligned}$$

Quiz. Check if the MATLAB solution is the least squares estimation of the above system.

$$f(X) = (x-4)^2 + (-x+y-4)^2 + (x-y-2)^2$$

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

$$b = [4 \ 4 \ 2]'$$

pinv(A)

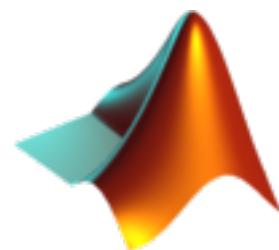
ans =

$$\begin{bmatrix} 1.0000 & -0.5000 & -0.5000 \\ 1.0000 & -0.0000 & -1.0000 \\ 0.0000 & 0.5000 & 0.5000 \end{bmatrix}$$

$$X = \text{pinv}(A) * b$$

X =

$$\begin{bmatrix} 4.0000 \\ 5.0000 \\ 0 \end{bmatrix}$$



Matrix equation

Under-determined system

$$-x+y = 4$$

Mathematically there are infinitely many solutions.
So how to solve this problem?

Can we still use LSE? Yes.

$$f(X) = (-x+y-4)^2$$

$$df/dx = 2x - 2y + 8 = 0$$

$$df/dy = -2x + 2y - 8 = 0$$

$$A = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

$$b = \begin{bmatrix} -8 & 8 \end{bmatrix}'$$

$$X = \text{pinv}(A) * b$$

$$X =$$

$$\begin{bmatrix} -2.0000 \\ 2.0000 \end{bmatrix}$$

*Question: But why we
only get one solution?*

Matrix equation

Under-determined system

$$-x+y = 4$$

Pseudoinverse approach also works.

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

$$b = [4 \ 0]'$$

$$X = \text{pinv}(A) * b$$

$$X =$$

$$\begin{bmatrix} -2.0000 \\ 2.0000 \end{bmatrix}$$

$$A = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$b = [4 \ 4]'$$

$$X = \text{pinv}(A) * b$$

$$X =$$

$$\begin{bmatrix} -2.0000 \\ 2.0000 \end{bmatrix}$$

$$A = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$b = [4 \ -4]'$$

$$X = \text{pinv}(A) * b$$

$$X =$$

$$\begin{bmatrix} -2.0000 \\ 2.0000 \end{bmatrix}$$

Unit Outcome

MATLAB programming requires breaking abstract concept into an algorithm consisting of

- 1) variables,
- 2) functions,
- 3) iterations,
- 4) conditional statements
- 5) random numbers.

Self-assessment Questions

- 1) Uniformly sample points on a unit sphere. Then display them using plot3.m
- 2) By uniformly sampling points on the unit sphere, estimate Pi. Is your algorithm slower/faster than algorithm using a circle. Why?
- 3) Solve the following equations using LSE:
$$\begin{aligned}x^2 &= 4 \\ -x+y &= 4 \\ x-y &= 2\end{aligned}$$