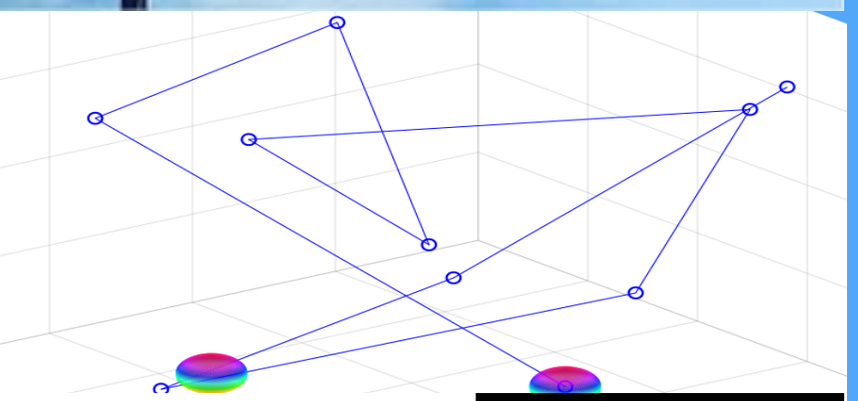
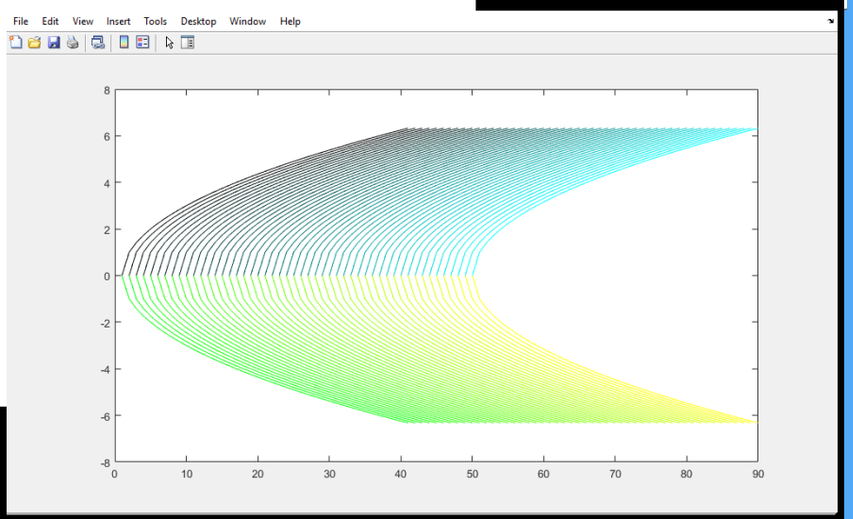
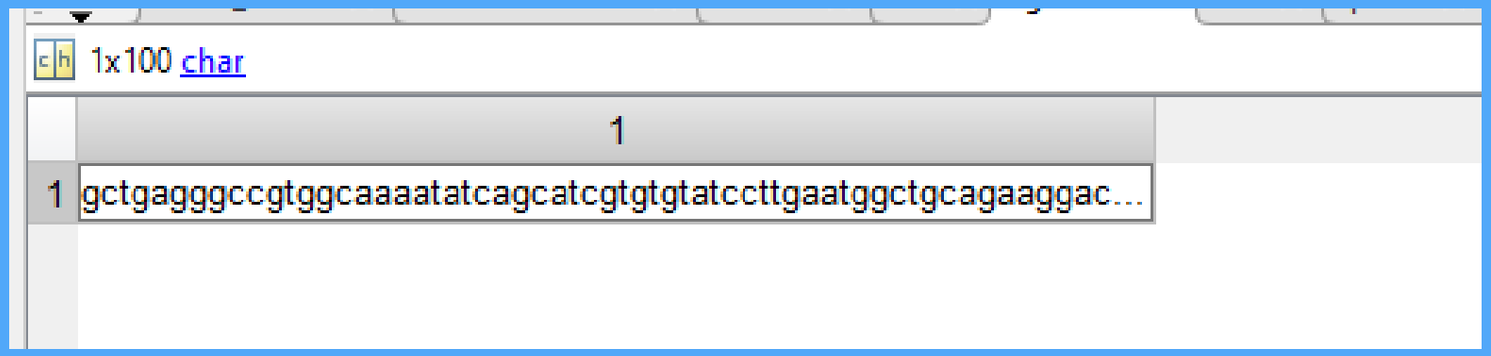
Introduction to Scientific Computation

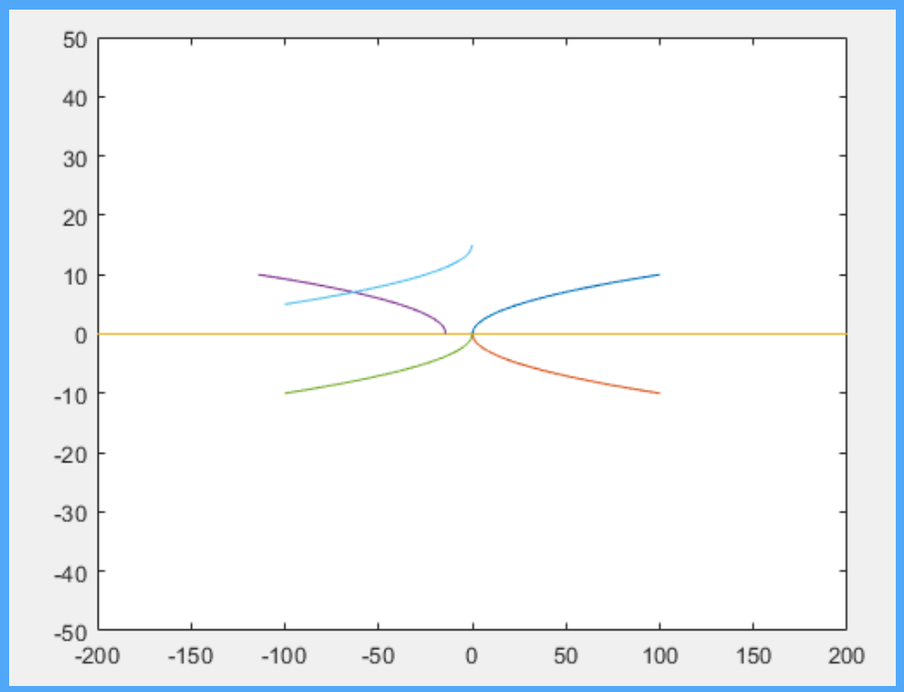
Halil Bayraktar

Lecture 5 - Flow control







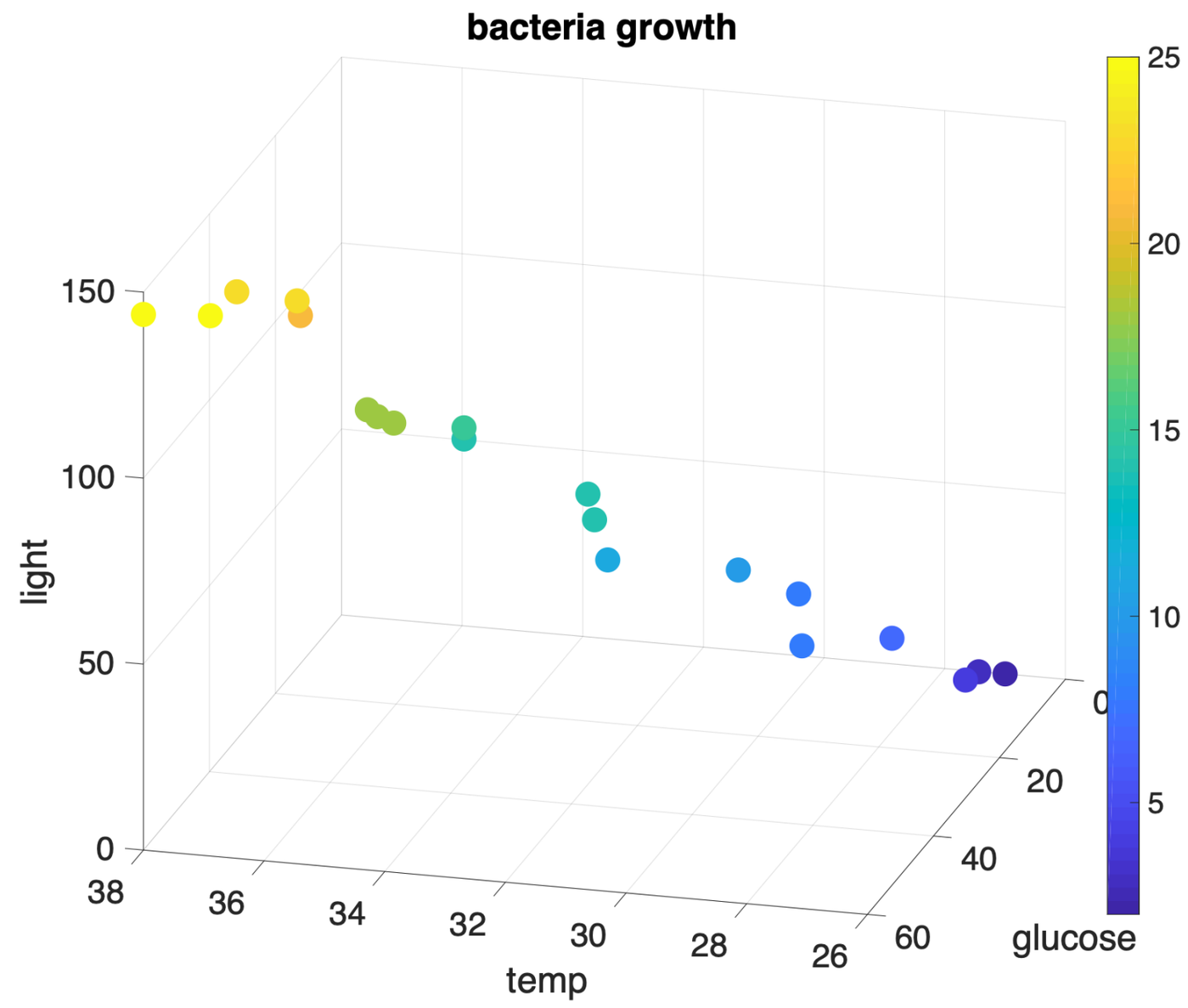


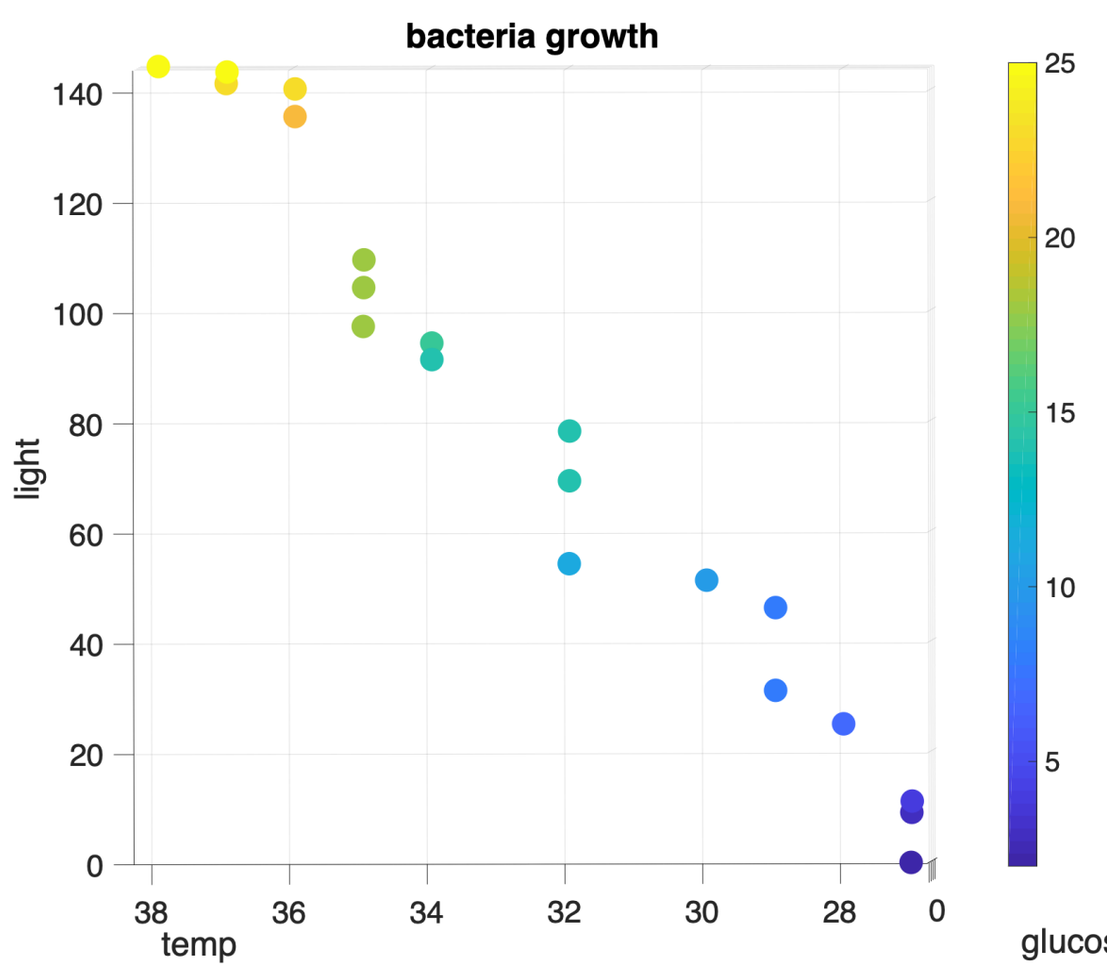
```
%%
se='ggg'
k=1
for i=1:98
  s1=genex(i:i+2)
  if se==s1
    pos(k,1)=i
    k=k+1
  end

end

genex(1,76:end)
```

Today

For loop and while loops

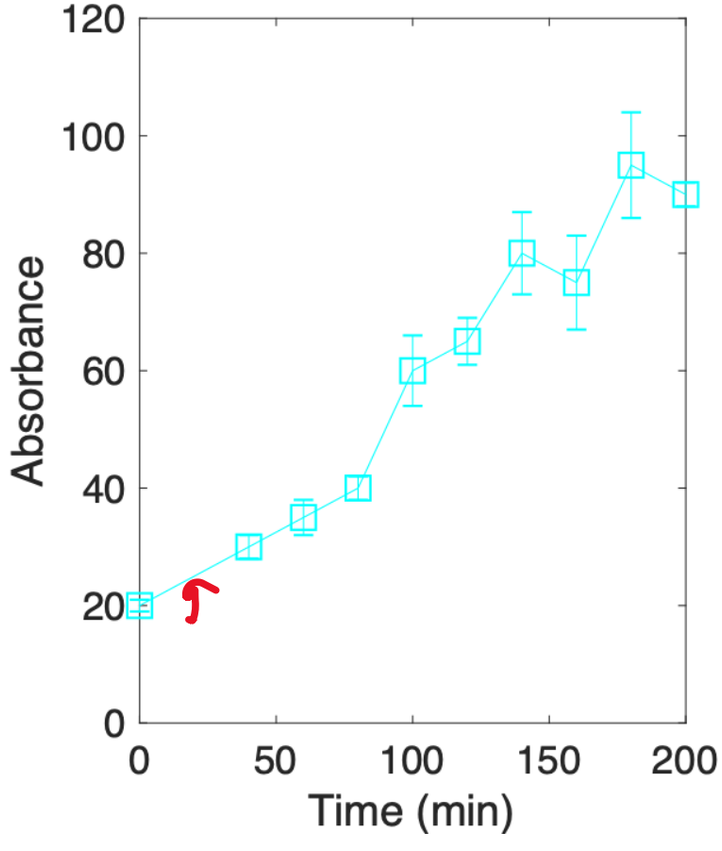
4D Plot

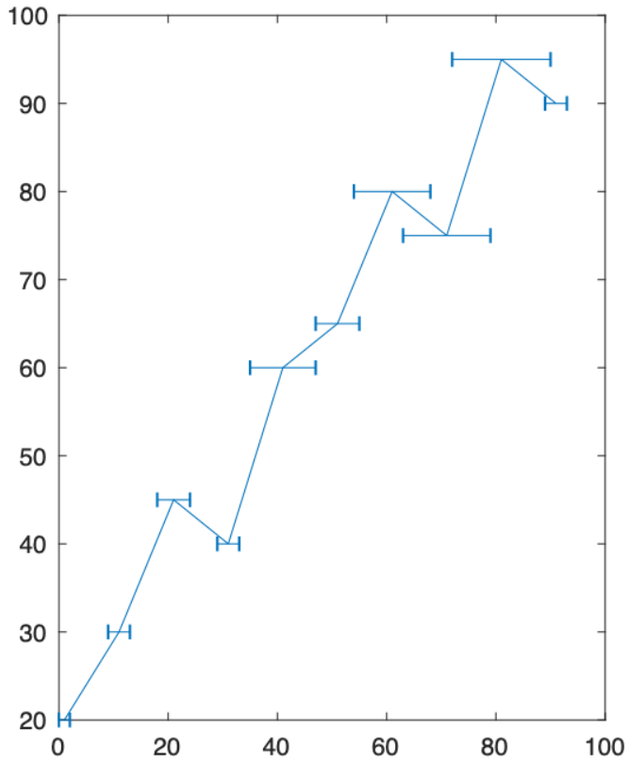

bacteria growth
light
temp
glucose
150
100
50
0
38
36
34
32
30
28
26
0
20
40
60
25
20
15
10
5

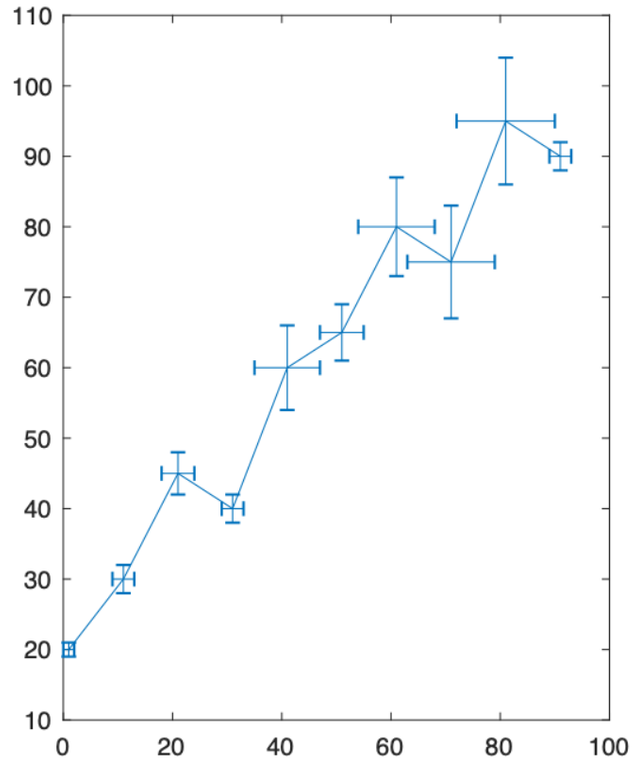

bacteria growth
light
temp
glucose
140
120
100
80
60
40
20
0
38
36
34
32
30
28
0
25
20
15
10
5

Error bar plots

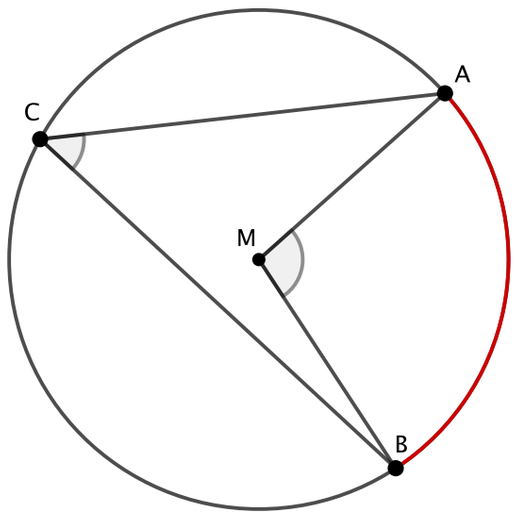
```
figure(2)
subplot(1,3,1)
time = [0 40 60 80 100 120 140 160 180 200];
protein = [20 30 35 40 60 65 80 75 95 90];
err = [1 2 3 2 6 4 7 8 9 2];
errorbar(time,protein,err,'-sc','capsize',8,'Markersize',14)
set(gca,'Fontsize',16,'fontname','arial')
xlabel('Time (min)')
ylabel('Absorbance')
saveas(figure(2),'absvstime','tif')
saveas(figure(2),'absvstime','png')
saveas(figure(2),'absvstime','jpeg')
```

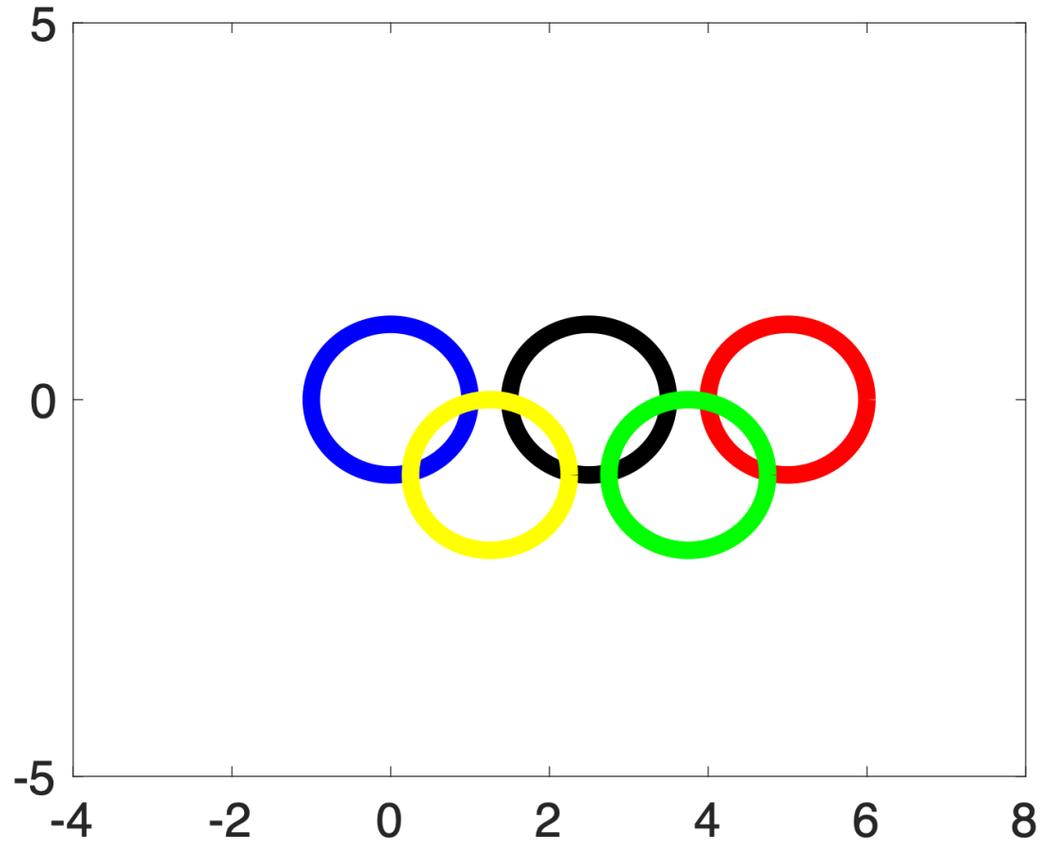






Geometry of a circle

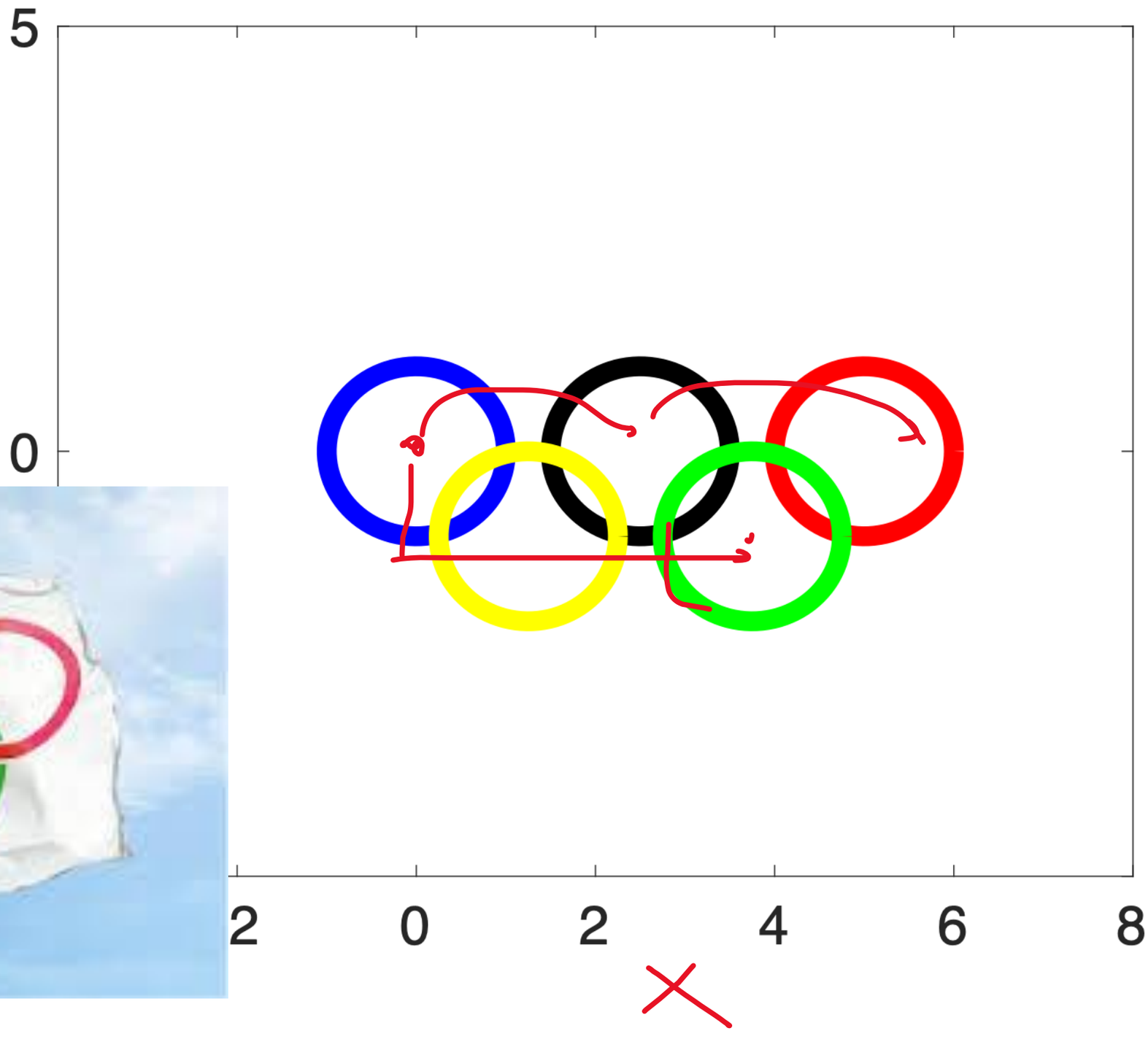




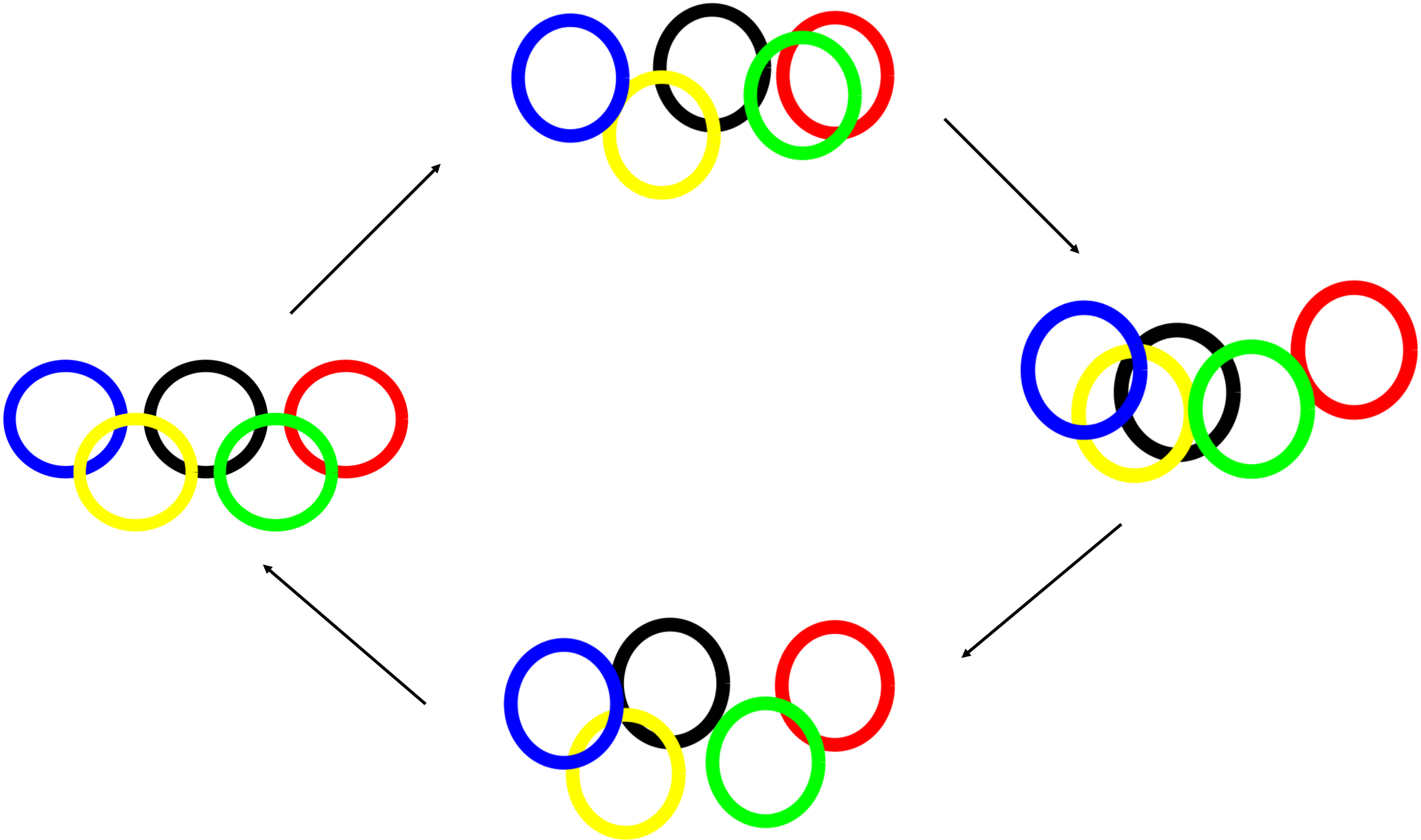

```
% circle cos sin
r=1
angle = linspace(0,pi*2,360) % radians
 angle2=0:0.1:6.30
 %
close all
x=r.*cos(angle2(1,:))  % cosd
y=r.*sin(angle2(1,:))
% sind
```

Olympic Rings with Matlab

```
figure(1)
plot(x+0,y+0,'-b','linewidth',8)
set(gca,'Fontsize',22)
hold on
plot(x+2.5,y+0,'-k','linewidth',8)
hold on
plot(x+5,y+0,'-r','linewidth',8)
hold on
plot(x+1.25,y-1.0,'-y','linewidth',8)
hold on
plot(x+3.75,y-1.0,'-g','linewidth',8)
%xticks([-1:0.2:1])
%yticks([-1:0.2:1])
axis([-4 8 -5 5])
```



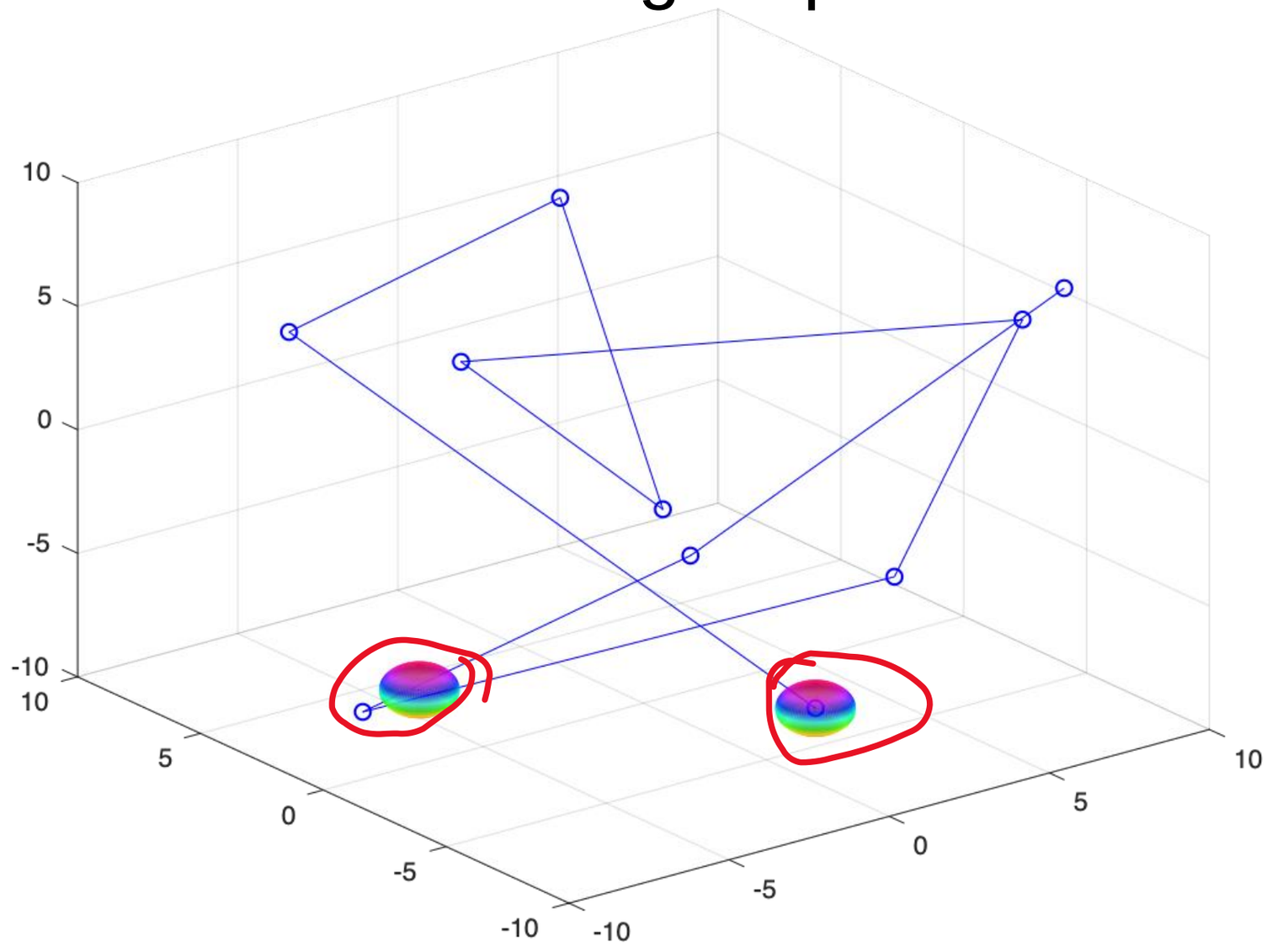
Lets move/jiggle the rings!!



3-Dimensional random movement with Matlab
Create a shape of circle and sphere

```
clear all
x=linspace(0,10,5)

N = 100;
angle1 = linspace(0,pi,N);
angle2 = linspace(0,pi*2,N);
[th, ph] = meshgrid(angle1,angle2);
R =1*ones(size(th))
z =R.*cos(th);
x= R.*sin(th).*cos(ph);
y= R.*sin(th).*sin(ph);
figure(2)
s=surf(x,y,z)
colormap(hsv) %jet, winter, summer, hsv, hot
s.EdgeColor='none'
alpha(0.5)
axis([-10 10 -10 10 -10 10])
```

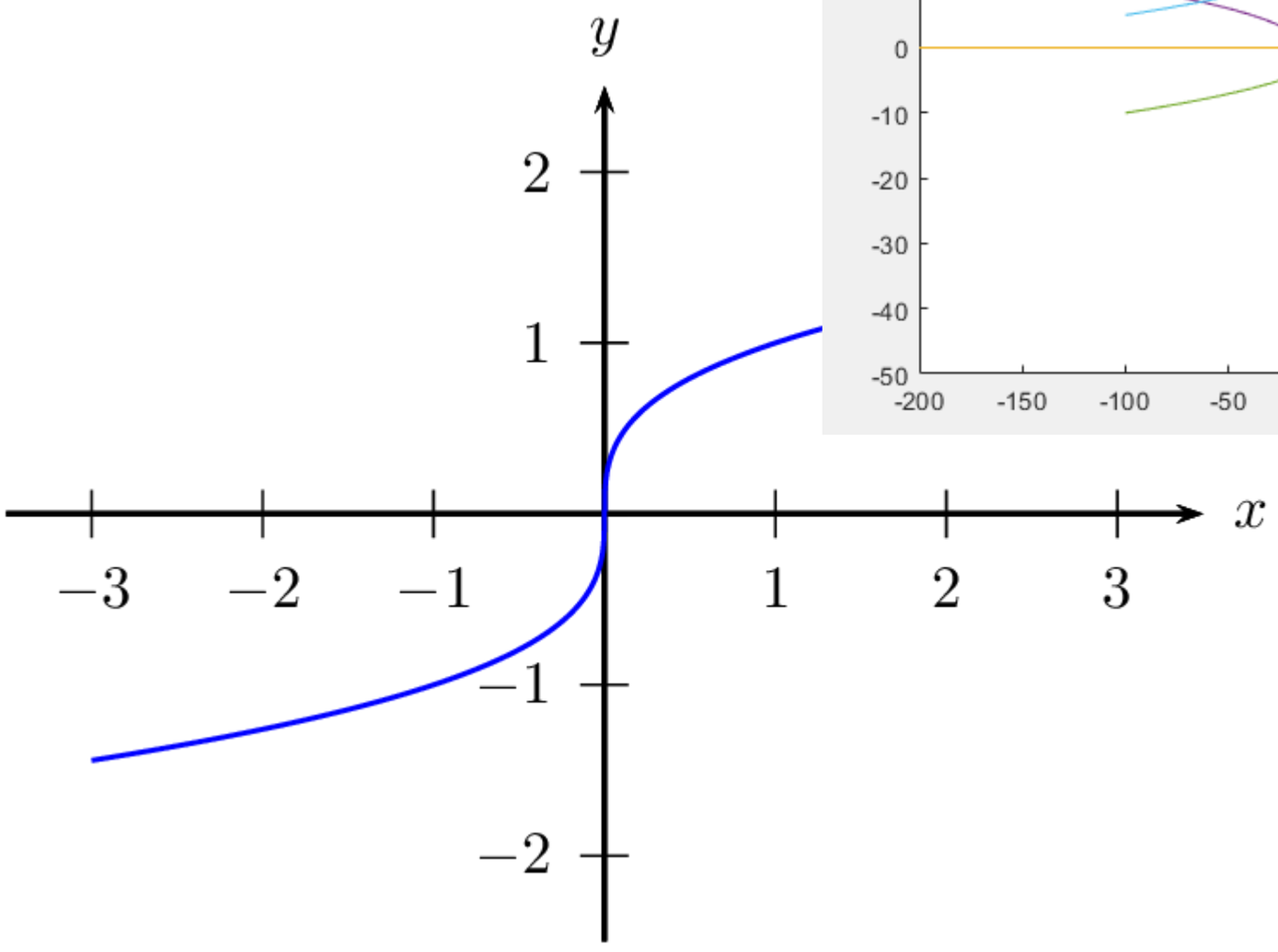
Animating a sphere

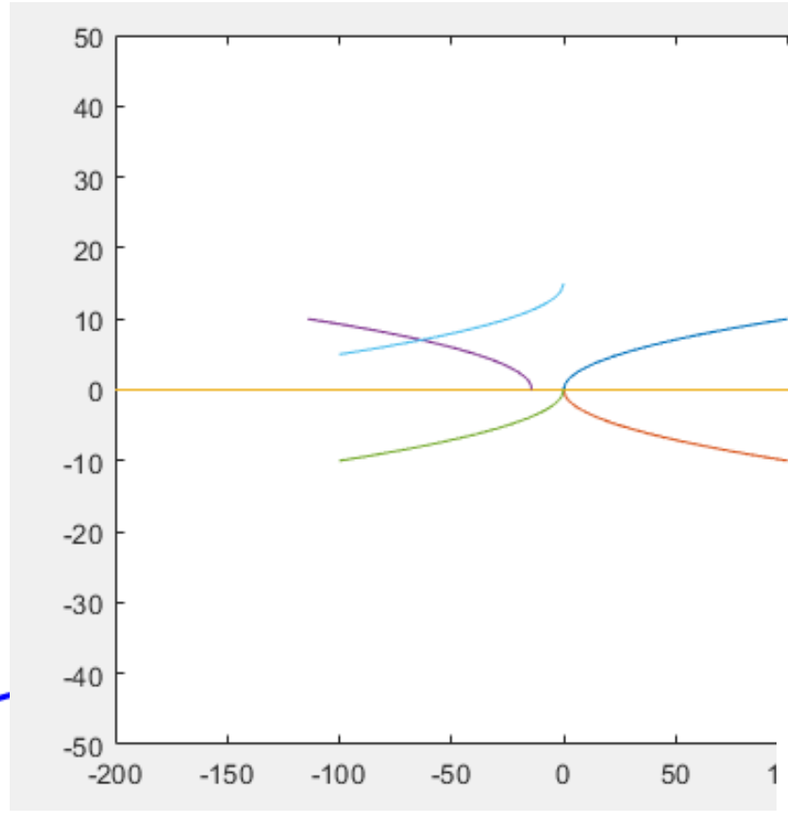


Other geometric functions with matlab

f(x)=a*(y)^0.5 or

f(x)=a*(y)^2

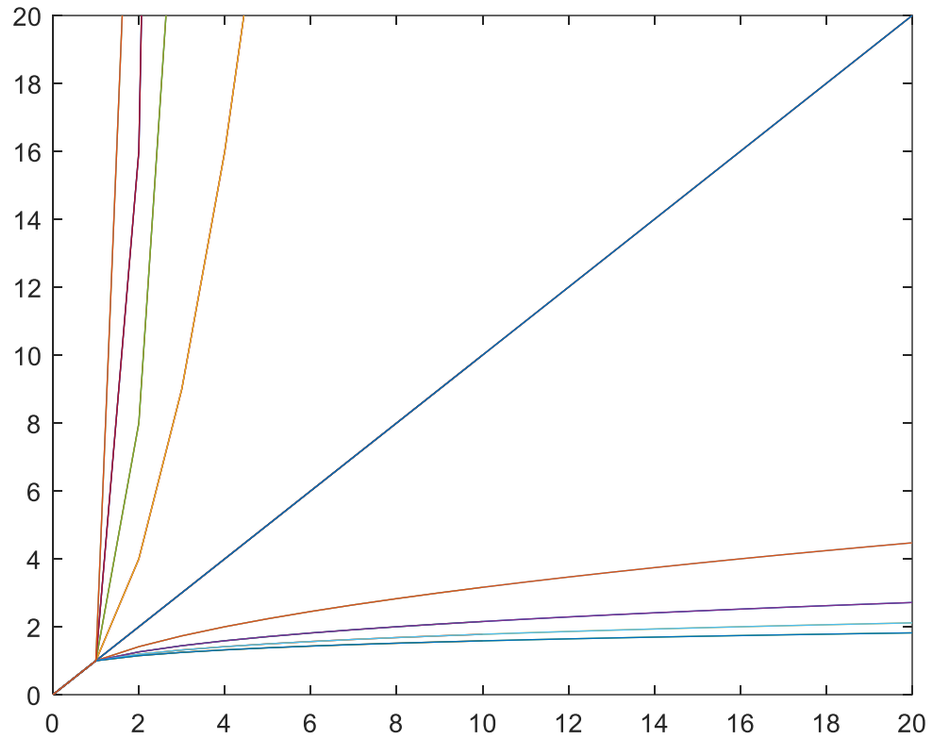





```
%%
x=0:1:100
y=x.^0.5
figure(1)
plot(x,y)
axis([-200 200 -50 50])
hold on
plot(x,-1*y)
hold on
xl=-200:1:200
yl=zeros(1,401)
plot(xl,yl)
hold on
x=-100:1:0
y=(x.*-1).^0.5
figure(1)
plot(x-0,y)
hold on
figure(1)
plot(x-20,(-1*y)-20)
hold on
```

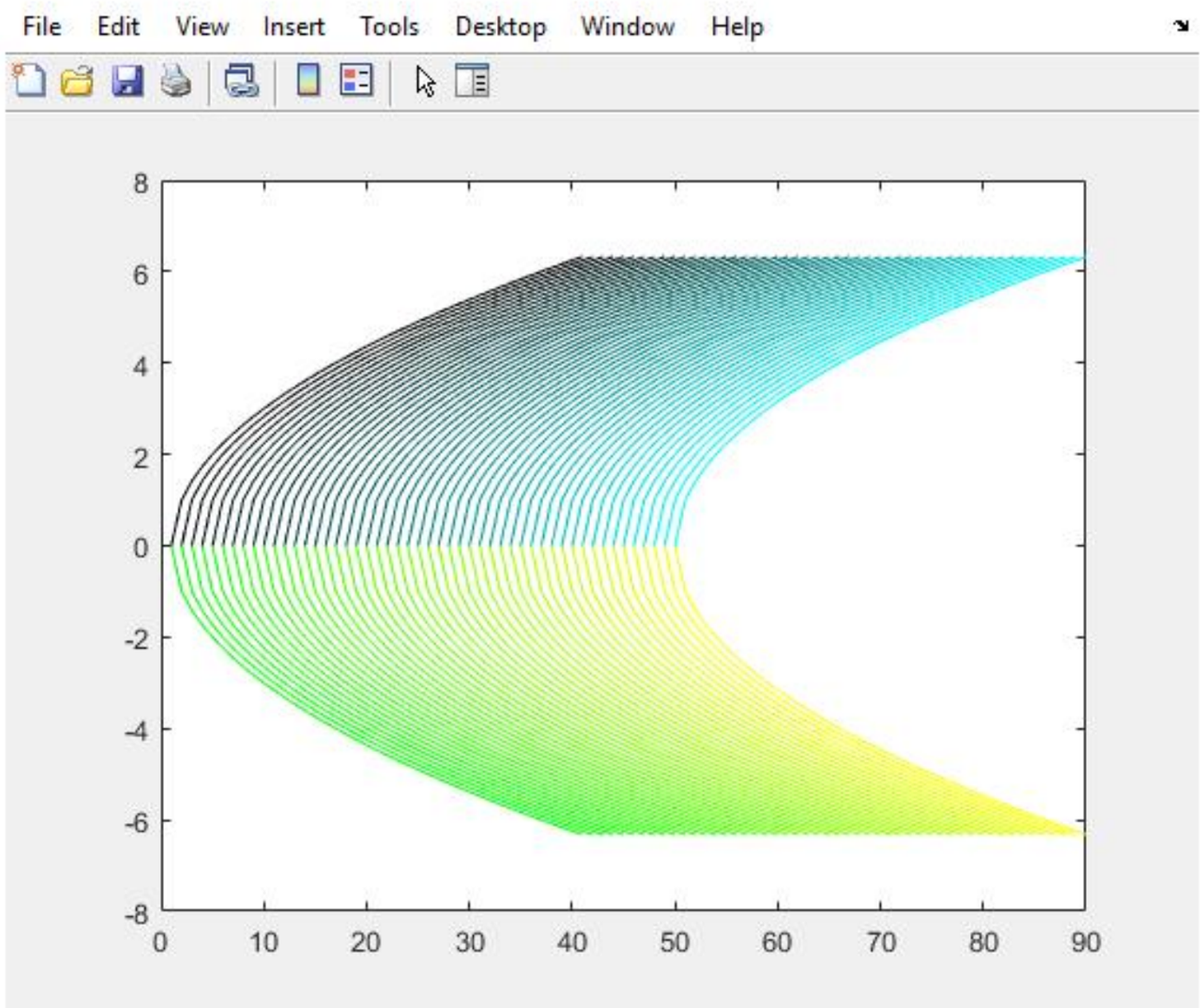
For loop

```
% for loops
x=1:1:100
y=reshape(x,10,10)

for i=1:10
    y(i,1)=i+5

end
```




```
%%
for i=1:5
    x=0:1:20
y=x.^(1/i)
y1=x.^(i)
figure(1)
plot(x,y)
hold on
plot(x,y1)
axis([0 20 0 20])
hold on
end
```

```
% repeating with for loops

for i=1:100;
    disp(i)
    fprintf('%f',i)
     disp('matlab')
     x(i,1)=i
     x(i,2)=sqrt(i)

end
```

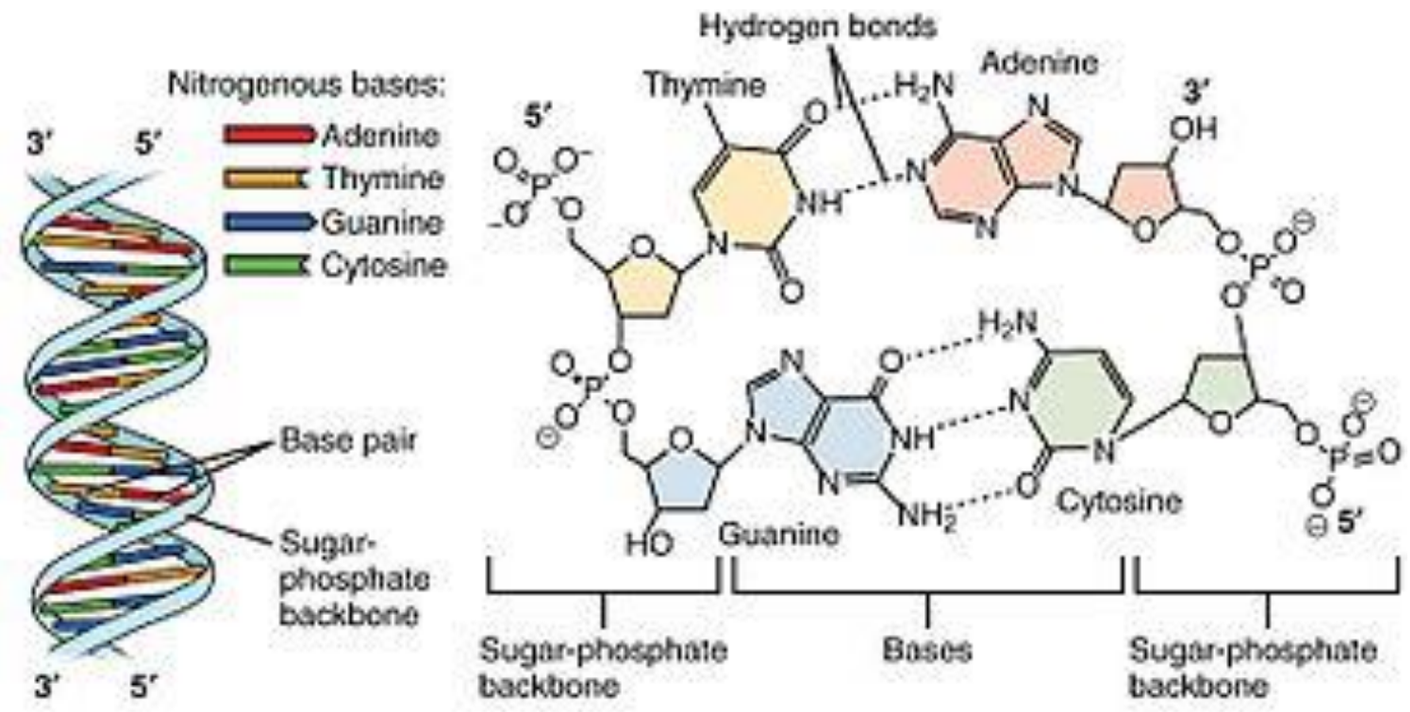


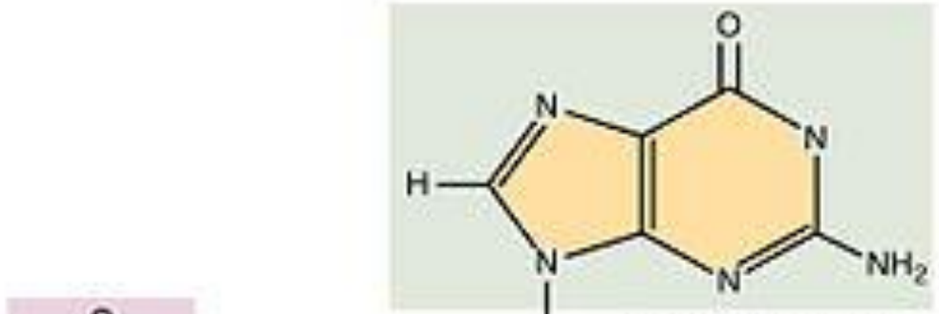

```
x=0:1:40
y=x.^0.5
z=linspace(0,1,50)
figure(1)
for i=1:50
    plot(x+i,y,'-r', 'color',[0 z(i) z(i)])
    hold on
    plot(x+i,-1*y,'color',[z(i) 1 0])

end
```

A genome sequence with for loop

We need 4 letters

```
%%
letters=['a','g','c','t']
genex=''
for i=1:100
    a=randi([1,4],1,1)
     genex(i)=letters(a)


end
```





Searching a sequence

Method 1
With if statement

```
%%
letters=['a','g','c','t']
genex=''
for i=1:100
    a=randi([1,4],1,1)
     genex(i)=letters(a)

end
size(genex)
%%
se='ggg'
k=1
for i=1:98
  s1=genex(i:i+2)
  if se==s1
     pos(k,1)=i
     k=k+1
  end

end

genex(1,76:end)
```

Method 2
With find function

```
%%
nucl=['a','g','c','t']
genex=''
pArr={}
for j=1:10;
for i=1:1000;
    r=randi([1,4],1,1);
    genex(i)=nucl(r);

end
pArr{j,1}=genex;
end
```

```
%%
se='aaaa'
clear reg
reg=[]
k=1
for i=1:997;
  r= find(genex(1,i:i+3)==se);
     reg(k,1)=length(r)
     k=k+1

end
%%
 [r,c,l] =find(reg==4)
%%
```

Running time of for loops

1 loop
Average speed was **0.9 ms**

```
tic
for  i=0:2000;
l(i,1)=sqrt(i);
l(i,2)=i.^2;

end
toc
```

2 loops
Average loop time is **100 ms**

```
tic
for i=1:2000;
for j=1:2000;
k(i,1)=sqrt(i);
k(j,2)=j.^2;
end
end
toc
```

Do not use many for loops if not necessary

Infinite loop with while loop


```
x=5
y=1
while x==5
    disp(y^2)
    y=y+1
end
```

Infinite loop saturates CPU (central processing unit)

```
x=5
y=1
while x==5
    disp(y^2)
    y=y+1
end
```

Before

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	PID
kernel_task	5.6	4:11:31.84	136	183	0
WindowServer	3.5	4:16:36.34	6	50	131
Activity Monitor	3.0	1:51.45	5	4	20792
airportd	1.7	17:22.41	9	4	61
Wi-Fi	1.5	12:45.71	6	1	398
locationd	1.4	9:15.71	9	1	89
Spotlight Networking	1.3	16.86	14	2	736
hidd	1.1	1:02:26.66	5	0	103
sysmond	0.8	53.25	3	0	20390
https://b2lab.wordpress.com	0.7	13.50	9	19	21313
python	0.5	20:59.41	22	89	545
[illegible]	0.4	17:31.62	19	78	546
MATLAB	0.4	1:23.52	99	60	21303
Dropbox	0.3	10:03.07	217	4	3397

System diagnostics options

After

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	PID
MATLAB	223.2	2:09.04	100	152	21303
WindowServer	30.4	4:16:46.03	6	123	131
kernel_task	11.2	4:11:35.96	136	5,445	0
screencapture	6.1	0.34	4	0	21345
airportd	3.2	17:23.38	8	9	61
Wi-Fi	2.5	12:46.36	6	1	398
locationd	2.1	9:16.40	9	3	89
Activity Monitor	2.0	1:52.99	6	1	20792
hidd	1.4	1:02:27.42	5	0	103
launchd	1.0	12:27.31	4	0	1
https://b2lab.wordpress.com	0.6	13.96	8	19	21313
sysmond	0.6	53.68	3	1	20390
quicklookd	0.4	0.22	7	2	21186
python	0.3	20:59.66	22	85	545

Cryptography

The science of writing secret codes is called **cryptography**.

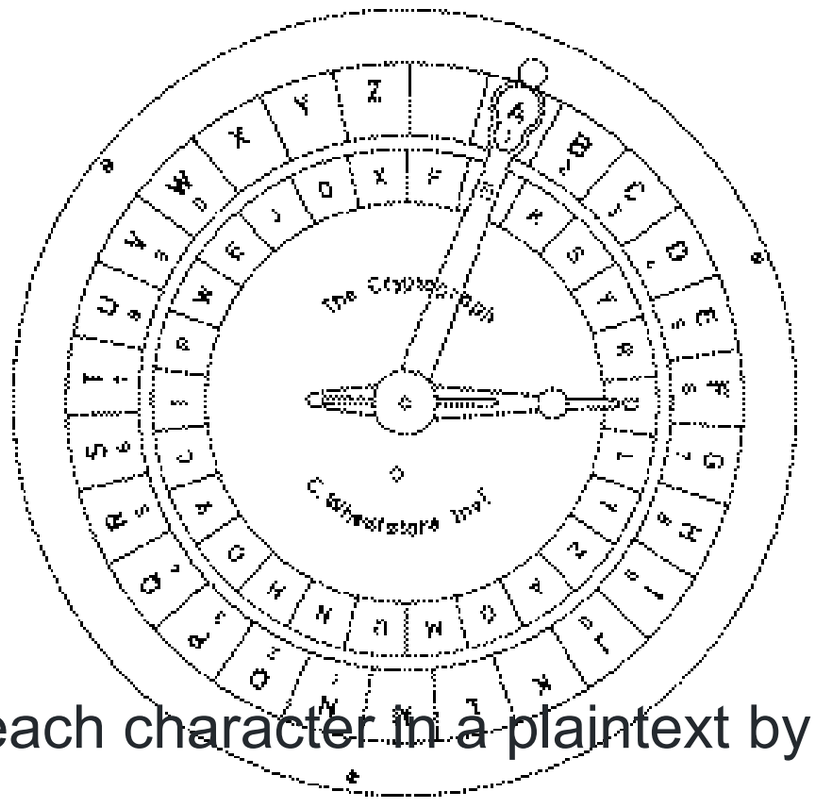
The message called **plaintext is encrypted**

Encrypt and Decrypt Messages
Programming and Cryptography

EGYPT HIEROGLYPHICS

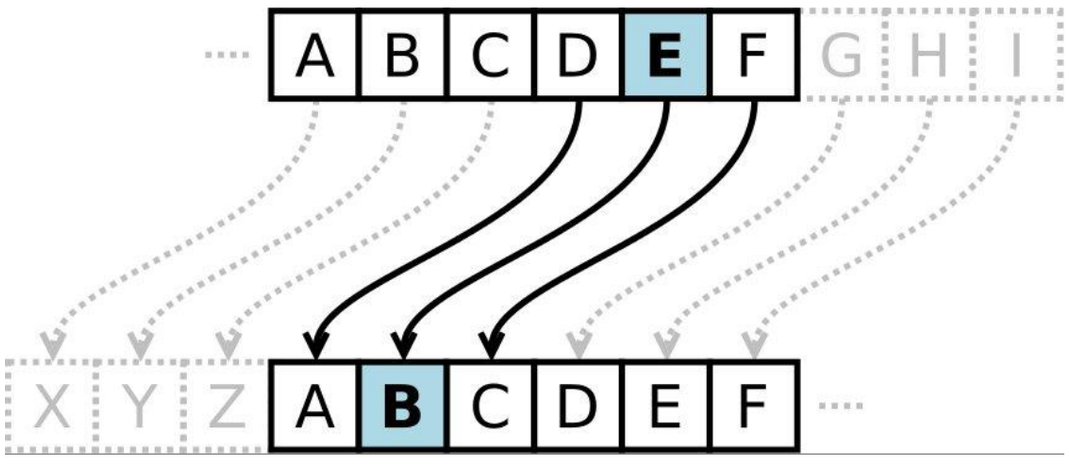
Caeser Cypher

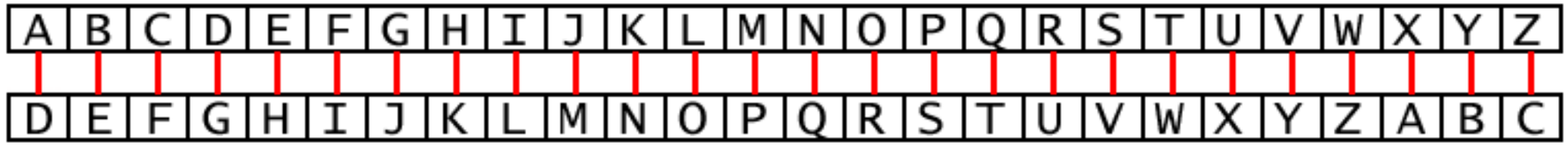
Caesar Cipher was one of the earliest ciphers ever invented.



A Caesar cipher involves shifting each character in a plaintext by three letters forward or backw

At the end or beginning of the alphabet, the cipher mapping wraps around the end,





Ceaser send the following message to his commander.

['p', 'r', 'y', 'h', 'q', 'r', 'u', 'w', 'k', 'z', 'd', 'l', 'w', 'i', 'l', 'u', 'h']

What is the message=?

Encrypting Messages

Decrypting Messages

['p', 'r', 'y', 'h', 'q', 'r', 'u', 'w', 'k', 'z', 'd', 'l', 'w', 'i', 'l', 'u', 'h']

['m', 'o', 'v', 'e', 'n', 'o', 'r', 't', 'h', 'w', 'a', 'i', 't', 'f', 'i', 'r', 'e']

```
%# Ceaser said:
caeser='movenorthwaitfire'

size(caeser,2)
%%
% uint8 takes a letter and returns a number
for i=1:size(caeser,2)
    k=caeser(i)
    t{i}=uint8(k)

    display(t)

end
%%
% char takes number and returns a letter
key=1
for i=1:size(caeser,2)
    dec{i}=char(t{i}+key)


end

%%
% commander receives the message
% also knows what the key is
key=-1v
for i=1:size(caeser,2)
    message{i}=char(dec{i}+key)

end
```

Char=returns the character

Uint8=returns binary number

Uint8 and char functions

representing each letter as a number called an **ordinal**,

A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

capital letters “A” through “Z” have the ASCII numbers 65 through 90.

lowercase letters “a” through “z” have the ASCII numbers 97 through 122

Universal ASCII Character Table

ASCII was developed a long time ago and now the non-printing characters are rarely used for t

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Vigenere Cipher

A Vigenere cipher is similar to a Caesar cipher. Every characte

Plain text: MOVENORTH

Key= CEASER

Encrypted text: P

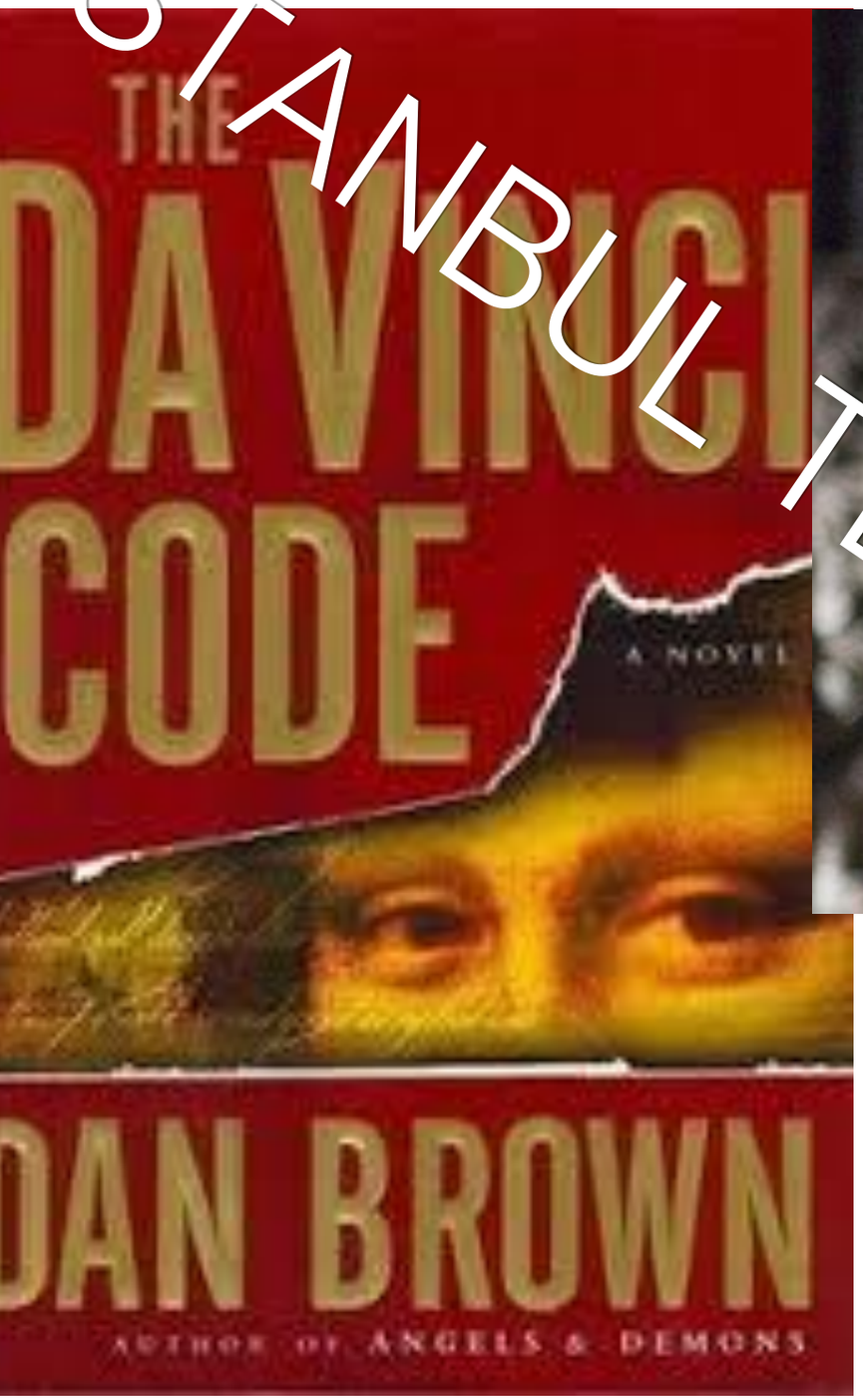
M+C=P

12+3=15

O+E=?

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Vigenere Cipher



Key=APPLE

cryptex

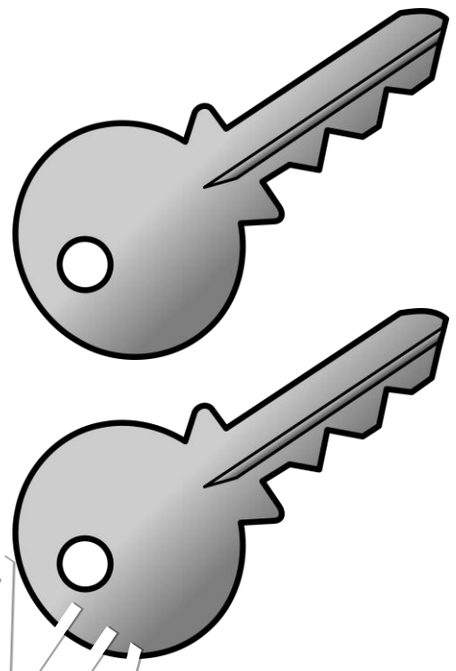
Modern Cryptosystem

They use two keys, public and private key

['m', 'o', 'v', 'e', 'n', 'o', 'r', 't', 'h', 'w', 'a', 'i', 't', 'f', 'i', 'r', 'e']

Encrypt

public key



4h3289h2nfaknfa98wn0

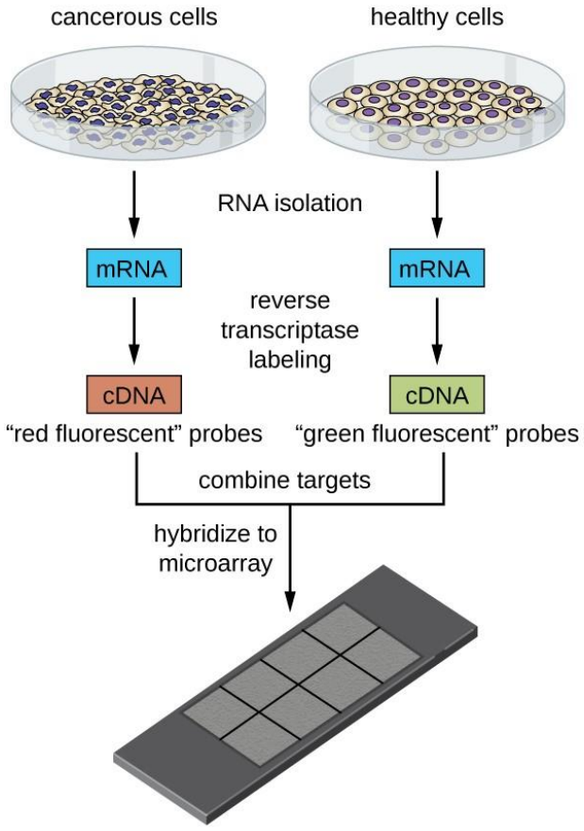
Decrypt

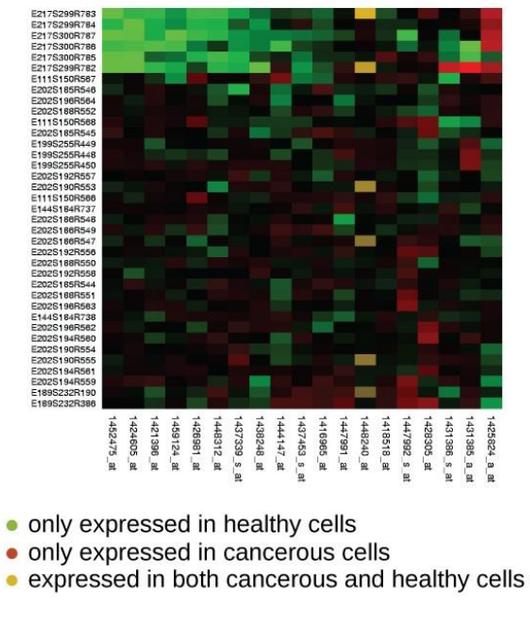
private key

['m', 'o', 'v', 'e', 'n', 'o', 'r', 't', 'h', 'w', 'a', 'i', 't', 'f', 'i', 'r', 'e']

Compare data with logic operators

```
geneA=randi(100,1000,1)
geneB=randi(100,1000,1)
geneC=randi(100,1000,1)
```

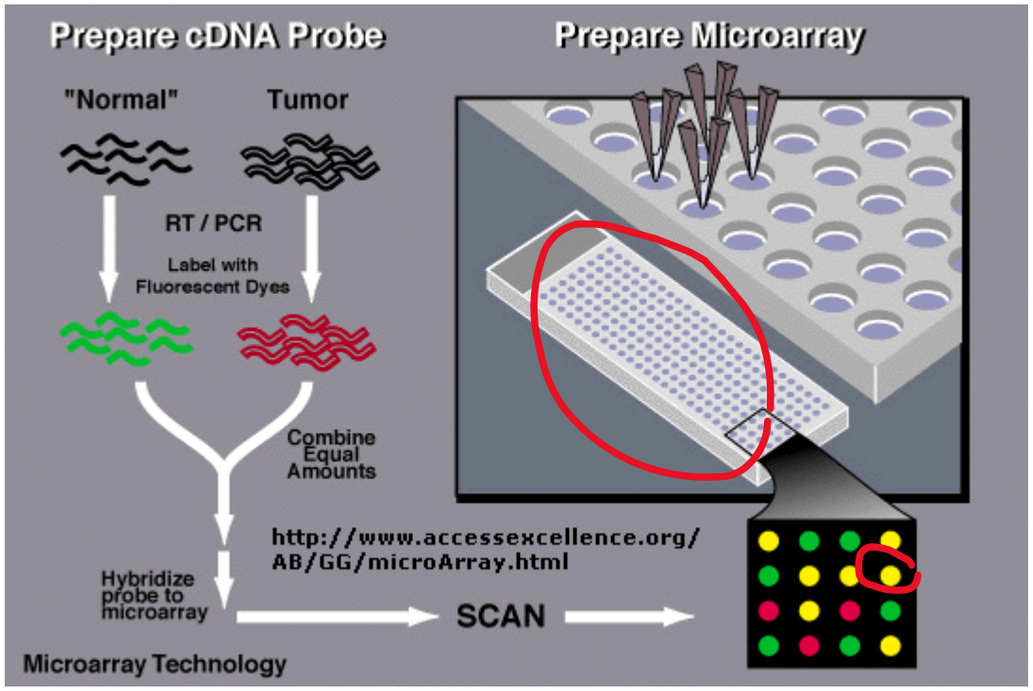




- only expressed in healthy cells
- only expressed in cancerous cells
- expressed in both cancerous and healthy cells

```
%
```

```
x=find(geneA>90 & geneB>90
& geneC<90)
geneA(x,1)
geneB(x,1)
geneC(x,1)
```



2d microarray data

Variables - data1

corr_coef_y | allmeansdata | p_value_o | data1

27130x34 table

	1 Gene	2 ARNA	3 TRNA	4 ARNA1	5 TRNA1	6 ARNA2	7 TRNA2	8 ARNA3	9 TRNA3	10 TRNA4	11 TRNA5
1	"LOC1024...	0	1.0045	4.0185	0.9162	2.9799	3.3377	1.3212	2.1511	1.0805	
2	"ZBTB42"	27.8394	37.1676	55.2547	30.2348	42.7112	32.2643	54.1705	48.3991	35.6578	50.26
3	"FCAMR"	1.1136	0	1.0046	0	0.9933	0	0	0	0	
4	"ZNF503-...	41.2024	35.1586	40.1853	16.4917	35.7582	32.2643	60.7767	23.6618	47.5438	40.84
5	"NFU1"	111.3578	123.5573	118.5465	133.7663	91.3821	86.7800	76.6315	120.4600	100.4903	68.07
6	"ELSPBP1"	0	0	0	0	0	0	0	0	0	
7	"ZRANB3"	190.4218	183.8291	141.6531	169.4984	155.9455	92.3428	118.9109	121.5355	152.3563	102.63
8	"MECR"	259.4637	291.3139	188.8708	237.2977	289.0455	189.1358	257.6403	253.8264	347.9341	271.24
9	"LOC1057...	0	0	0	0	0	0	0	0	0	
10	"LINC003...	2.2272	2.0091	3.0139	4.5810	6.9530	0	0	0	1.0805	4.18
11	"AARSD1"	1.1136	0	0	0	0	2.2251	0	0	0	
12	"DEXI"	485.5200	435.9663	492.2695	308.7619	511.5410	493.9782	478.2860	357.0779	467.8742	569.72
13	"DCHS1"	1.1559e+03	1.0035e+03	1.2116e+03	1.3138e+03	1.3479e+03	1.4652e+03	1.8788e+03	1.4842e+03	1.7224e+03	1.2724e+
14	"PSMD2"	1.2550e+03	1.8232e+03	1.3914e+03	1.4861e+03	1.5545e+03	1.3206e+03	1.5630e+03	1.1282e+03	1.4382e+03	1.5730e+
15	"GABRR1"	3.3407	4.0181	2.0093	1.8324	5.9597	8.9005	6.6062	2.1511	2.1611	4.18
16	"PKNOX2"	780.6181	676.0491	640.9550	244.6274	522.4672	406.0857	486.2134	287.1680	504.6126	799.07
17	"TIPARP"	309.5747	294.3275	372.7184	721.0553	238.3881	354.9078	395.0484	379.6641	298.2293	273.34
18	"ADAM20"	113.5849	91.4123	74.3427	89.7883	100.3216	160.2091	104.3773	73.1364	63.7519	97.39
19	"LOC2847...	0	0	0	0	0	0	0	0	0	
20	"MIR4715"	0	0	0	0	0	1.1126	0	0	0	

Table to array conversion of data with for loop

Variables - data1

corr_coef_y | allmeansdata | p_value_o | data1

27130x34 table

	1 Gene	2 ARNA	3 TRNA	4 ARNA1	5 TRNA1	6 ARNA2	7 TRNA2	8 ARNA3	9 TRNA3	10 TRNA4	11 TRNA5
1	"LOC1024...	0	1.0045	4.0185	0.9162	2.9799	3.3377	1.3212	2.1511	1.0805	
2	"ZBTB42"	27.8394	37.1676	55.2547	30.2348	42.7112	32.2643	54.1705	48.3991	35.6578	50.26
3	"FCAMR"	1.1136	0	1.0046	0	0.9933	0	0	0	0	
4	"ZNF503-...	41.2024	35.1586	40.1853	16.4917	35.7582	32.2643	60.7767	23.6618	47.5438	40.84
5	"NFU1"	111.3578	123.5573	118.5465	133.7663	91.3821	86.7800	76.6315	120.4600	100.4903	68.07
6	"ELSPBP1"	0	0	0	0	0	0	0	0	0	
7	"ZRANB3"	190.4218	183.8291	141.6531	169.4984	155.9455	92.3428	118.9109	121.5355	152.3563	102.63
8	"MECR"	259.4637	291.3139	188.8708	237.2977	289.0455	189.1358	257.6403	253.8264	347.9341	271.24
9	"LOC1057...	0	0	0	0	0	0	0	0	0	
10	"LINC003...	2.2272	2.0091	3.0139	4.5810	6.9530	0	0	0	1.0805	4.18
11	"AARSD1"	1.1136	0	0	0	0	2.2251	0	0	0	
12	"DEXI"	485.5200	435.9663	492.2695	308.7619	511.5410	493.9782	478.2860	357.0779	467.8742	569.72
13	"DCHS1"	1.1559e+03	1.0035e+03	1.2116e+03	1.3138e+03	1.3479e+03	1.4652e+03	1.8788e+03	1.4842e+03	1.7224e+03	1.2724e+
14	"PSMD2"	1.2550e+03	1.8232e+03	1.3914e+03	1.4861e+03	1.5545e+03	1.3206e+03	1.5630e+03	1.1282e+03	1.4382e+03	1.5730e+
15	"GABRR1"	3.3407	4.0181	2.0093	1.8324	5.9597	8.9005	6.6062	2.1511	2.1611	4.18
16	"PKNOX2"	780.6181	676.0491	640.9550	244.6274	522.4672	406.0857	486.2134	287.1680	504.6126	799.07
17	"TIPARP"	309.5747	294.3275	372.7184	721.0553	238.3881	354.9078	395.0484	379.		
18	"ADAM20"	113.5849	91.4123	74.3427	89.7883	100.3216	160.2091	104.3773	73.		
19	"LOC2847...	0	0	0	0	0	0	0			
20	"MIR4715"	0	0	0	0	0	1.1126	0	0	0	

```
for i=1:200
subdata1(i,:)=table2array(data1(i,2:34));

end
%%
```

200x33 double

	1	2	3	4	5
1	0	1.0045	4.0185	0.9162	2.979
2	27.8394	37.1676	55.2547	30.2348	42.711
3	1.1136	0	1.0046	0	0.993
4	41.2024	35.1586	40.1853	16.4917	35.758
5	111.3578	123.5573	118.5465	133.7663	91.382
6	0	0	0	0	0
7	190.4218	183.8291	141.6531	169.4984	155.945
8	259.4637	291.3139	188.8708	237.2977	289.045
9	0	0	0	0	0
10	2.2272	2.0091	3.0139	4.5810	6.953
11	1.1136	0	0	0	0
12	485.5200	435.9663	492.2695	308.7619	511.541
13	1.1559e+03	1.0035e+03	1.2116e+03	1.3138e+03	1.3479e+03
14	1.2550e+03	1.8232e+03	1.3914e+03	1.4861e+03	1.5545e+03
15	3.3407	4.0181	2.0093	1.8324	5.959
16	780.6181	676.0491	640.9550	244.6274	522.467
17	309.5747	294.3275	372.7184	721.0553	238.388
18	113.5849	91.4123	74.3427	89.7883	100.321
19	0	0	0	0	0
20	0	0	0	0	0
21	1.3274e+03	1.6374e+03	1.3000e+03	1.2341e+03	1.2416e+03
22	1.3285e+03	1.1422e+03	977.5067	1.1333e+03	1.3072e+03
23	150.3330	145.6569	190.8800	127.3528	113.234

Data cleanup
Some datasets can include unmeasured values, we should remove than before we analyze the data

200x33 double

	1	2	3	4	5
1	0	1.0045	4.0185	0.9162	2.979
2	27.8394	37.1676	55.2547	30.2348	42.711
3	1.1136	0	1.0046	0	0.993
4	41.2024	35.1586	40.1853	16.4917	35.758
5	111.3578	123.5573	118.5465	133.7663	91.382
6	0	0	0	0	
7	190.4218	183.8291	141.6531	169.4984	155.945
8	259.4637	291.3139	188.8708	237.2977	289.045
9	0	0	0	0	
10	2.2272	2.0091	3.0139	4.5810	6.953
11	1.1136	0	0	0	
12	485.5200	435.9663	492.2695	308.7619	511.541
13	1.1559e+03	1.0035e+03	1.2116e+03	1.3138e+03	1.3479e+03
14	1.2550e+03	1.8232e+03	1.3914e+03	1.4861e+03	1.5545e+03
15	3.3407	4.0181	2.0093	1.8324	5.959
16	780.6181	676.0491	640.9550	244.6274	522.467
17	309.5747	294.3275	372.7184	721.0553	238.388
18	113.5849	91.4123	74.3427	89.7883	100.321
19	0	0	0	0	
20	0	0	0	0	
21	1.3274e+03	1.6374e+03	1.3000e+03	1.2341e+03	1.2416e+03
22	1.3285e+03	1.1422e+03	977.5067	1.1333e+03	1.3072e+03
23	150.3330	145.6569	190.8800	127.3528	113.234

Data normalization
Scale values between 0 and 1

```
%%
xn=randi([1,10],10,10)
yn=(xn-min(min(xn)))./(max(max(xn))-min(min(xn)))
```

10x10 double

	1	2	3	4	5	6	7
1	9	2	6	9	6	5	
2	6	8	4	6	9	5	
3	5	10	2	6	5	6	
4	9	4	2	3	7	9	
5	1	10	2	8	4	1	
6	6	4	8	6	7	2	
7	7	2	5	6	3	10	
8	8	4	2	5	8	10	
9	3	6	8	8	8	5	
10	1	7	2	1	10	9	
11							
12							

10x10 double

	1	2	3	4	5	6	7
1	0.8889	0.1111	0.5556	0.8889	0.5556	0.4444	
2	0.5556	0.7778	0.3333	0.5556	0.8889	0.4444	0.666
3	0.4444	1	0.1111	0.5556	0.4444	0.5556	0.333
4	0.8889	0.3333	0.1111	0.2222	0.6667	0.8889	0.444
5	0	1	0.1111	0.7778	0.3333	0	0.444
6	0.5556	0.3333	0.7778	0.5556	0.6667	0.1111	0.666
7	0.6667	0.1111	0.4444	0.5556	0.2222	1	0.444
8	0.7778	0.3333	0.1111	0.4444	0.7778	1	
9	0.2222	0.5556	0.7778	0.7778	0.7778	0.4444	0.333
10	0	0.6667	0.1111	0	1	0.8889	0.333
11							

Data normalization

Dividing values by the max value in the data set

```
for i=1:128
subdata1normalized(i,:)=subcleandata2(i,:)/subcleandata2max(i,1)
end
```

	1	2	3	4	5	6	7
	27.8394	37.1676	55.2547	30.2348	42.7112	32.2643	54.17
	41.2024	35.1586	40.1853	16.4917	35.7582	32.2643	60.77
	111.3578	123.5573	118.5465	133.7663	91.3821	86.7800	76.63
	190.4218	183.8291	141.6531	169.4984	155.9455	92.3428	118.91
	259.4637	291.3139	188.8708	237.2977	289.0455	189.1358	257.64
	485.5200	435.9663	492.2695	308.7619	511.5410	493.9782	478.28
	1.1559e+03	1.0035e+03	1.2116e+03	1.3138e+03	1.3479e+03	1.4652e+03	1.8788e+
	1.2550e+03	1.8232e+03	1.3914e+03	1.4861e+03	1.5545e+03	1.3206e+03	1.5630e+
	780.6181	676.0491	640.9550	244.6274	522.4672	406.0857	486.21
0	309.5747	294.3275	372.7184	721.0553	238.3881	354.9078	395.04
1	113.5849	91.4123	74.3427	89.7883	100.3216	160.2091	104.37
2	1.3274e+03	1.6374e+03	1.3000e+03	1.2341e+03	1.2416e+03	1.4953e+03	1.1904e+
3	1.3285e+03	1.1422e+03	977.5067	1.1333e+03	1.3072e+03	1.1738e+03	969.78
4	150.3330	145.6569	190.8800	127.3528	113.2343	140.1830	142.69
5	66.8147	36.1631	46.2131	39.3969	31.7851	52.2905	31.70
6	153.6738	171.7747	161.7457	154.8391	164.8851	141.2956	170.43
7	146.9923	150.6796	110.5095	95.2856	85.4224	74.5418	101.73
8	197.1033	381.7216	150.6948	362.8182	73.5030	70.0915	114.94
9	20.0444	12.0544	21.0973	6.4135	20.8590	66.7538	29.06
0	1.4610e+03	747.3707	816.7656	209.8115	345.6627	269.2404	948.64
1	228.2835	198.8971	337.5563	285.8567	128.1336	133.5076	153.26
2	40.0888	17.0770	20.0926	35.7321	21.8522	5.5628	17.17
3	209.3527	167.7566	239.1023	240.0464	182.7642	287.0414	204.79
4	8.9086	21.0951	11.0509	77.8776	16.8858	17.8010	19.81
5	178.1725	245.1055	286.3200	564.3838	412.2127	369.3711	363.33

128x30 double

	1	2	3	4	5	6	7
1	0.2646	0.3532	0.5251	0.2873	0.4059	0.3066	0.5
2	0.6779	0.5785	0.6612	0.2713	0.5884	0.5309	
3	0.6220	0.6902	0.6622	0.7472	0.5105	0.4848	0.4
4	0.6342	0.6122	0.4718	0.5645	0.5194	0.3075	0.3
5	0.5136	0.5767	0.3739	0.4697	0.5722	0.3744	0.5
6	0.6946	0.6237	0.7043	0.4417	0.7318	0.7067	0.6
7	0.6152	0.5341	0.6449	0.6993	0.7174	0.7799	
8	0.4944	0.7183	0.5482	0.5855	0.6124	0.5203	0.6
9	0.6377	0.5522	0.5236	0.1998	0.4268	0.3317	0.3
10	0.2816	0.2678	0.3391	0.6560	0.2169	0.3229	0.3
11	0.7090	0.5706	0.4640	0.5604	0.6262	1	0.6
12	0.6340	0.7821	0.6209	0.5894	0.5930	0.7142	0.5
13	0.9074	0.7801	0.6676	0.7741	0.8928	0.8017	0.6
14	0.7876	0.7631	1	0.6672	0.5932	0.7344	0.7
15	0.9645	0.5220	0.6671	0.5687	0.4588	0.7548	0.4
16	0.5592	0.6251	0.5886	0.5635	0.6000	0.5142	0.6
17	0.8626	0.8842	0.6485	0.5592	0.5013	0.4374	0.5
18	0.3002	0.5814	0.2295	0.5526	0.1120	0.1068	0.1
19	0.3003	0.1806	0.3160	0.0961	0.3125	1	0.4
20	0.6889	0.3524	0.3851	0.0989	0.1630	0.1270	0.4
21	0.5165	0.4500	0.7637	0.6467	0.2899	0.3020	0.3
22	0.8603	0.3665	0.4312	0.7668	0.4689	0.1194	0.3
23	0.7293	0.5844	0.8330	0.8363	0.6367	1	0.7
24	0.0810	0.1919	0.1005	0.7085	0.1536	0.1619	0.1

Mean values of subgroups in the data sets

Finding the mean and mean difference of different groups

128x30 double

	1	2	3	4	5	6	7
1	0.2646	0.3532	0.5251	0.2873	0.4059	0.3066	0.5
2	0.6779	0.5785	0.6612	0.2713	0.5884	0.5309	
3	0.6220	0.6902	0.6622	0.7472	0.5105	0.4848	0.4
4	0.6342	0.6122	0.4718	0.5645	0.5194	0.3075	0.3
5	0.5136	0.5767	0.3739	0.4697	0.5722	0.3744	0.5
6	0.6946	0.6237	0.7043	0.4417	0.7318	0.7067	0.6
7	0.6152	0.5341	0.6449	0.6993	0.7174	0.7799	
8	0.4944	0.7183	0.5482	0.5855	0.6124	0.5203	0.
9	0.6377	0.5522	0.5236	0.1998	0.4268	0.3317	
10	0.2816	0.2678	0.3391	0.6560	0.2169	0.3229	
11	0.7090	0.5706	0.4640	0.5604	0.6262	1	.6
12	0.6340	0.7821	0.6209	0.5894	0.5930	0.7142	0.5
13	0.9074	0.7801	0.6676	0.7741	0.8928	0.8017	0.6
14	0.7876	0.7631	1	0.6672	0.5932	0.7344	0.7
15	0.9645	0.5220	0.6671	0.5687	0.4588	0.7548	0.4
16	0.5592	0.6251	0.5886	0.5635	0.6000	0.5142	0.
17	0.8626	0.8842	0.6485	0.5592	0.5013	0.4374	0.5
18	0.3002	0.5814	0.2295	0.5526	0.1120	0.1068	0.1
19	0.3003	0.1806	0.3160	0.0961	0.3125	1	0.4
20	0.6889	0.3524	0.3851	0.0989	0.1630	0.1270	0.4
21	0.5165	0.4500	0.7637	0.6467	0.2899	0.3020	0.3
22	0.8603	0.3665	0.4312	0.7668	0.4689	0.1194	0.3
23	0.7293	0.5844	0.8330	0.8363	0.6367	1	0.7
24	0.0810	0.1919	0.1005	0.7085	0.1536	0.1619	0.1

	1	2	3	4	5	6	7	
1	0.3897	0.4291	0.2733	-0.0394	0.1163	0.1557	0	
2	0.5872	0.6033	0.4231	-0.0161	0.1641	0.1802	0	
3	0.6022	0.5466	0.7597	0.0557	-0.1575	-0.2132	0	
4	0.4888	0.5547	0.7128	-0.0659	-0.2240	-0.1581	0	
5	0.4866	0.6443	0.6451	-0.1577	-0.1585	-8.7756e-04	0	
6	0.6372	0.6580	0.5865	-0.0208	0.0508	0.0716	0	
7	0.7226	0.7814	0.6386	-0.0588	0.0840	0.1428	0	
8	0.5674	0.6311	0.7329	-0.0637	-0.1655	-0.1018	0	
9	0.4129	0.5800	0.4845	-0.1671	-0.0715	0.0955	0	
10	0.3486	0.3189	0.4546	0.0297	-0.1060	-0.1357	0	
11	0.6298	0.6326	0.5259	-0.0028	0.1039	0.1067	0	
12	0.6255	0.5954	0.7032	0.0301	-0.0777	-0.1078	0	

Have a nice week.