

מבוא ל-MATLAB

מיכאל קרפ

mkarp@technion.ac.il

* ניתן להעתיק/לשכפל באופן חופשי

תכנים

תכני MATLAB

- היכרות וסביבת עבודה
- ביטויים ומערכים ופעולות נומריות
- מבני מידע וארגון נתונים
- פעולות גרפיות בסיסיות
- תכנות וכתובת פונקציות
- פעולות לוגיות ובקרת זרימה
- נושאים מתקדמים :
- קריאת וכתובת קבצים
- הכרת toolboxes
- גרפיקה מתקדמת
- ממשקי משתמש גרפיים (GUI)
- Simulink

יישומים הנדסיים

- פתרון מערכות לינאריות
- גזירה ואינטגרציה
- טרנספורמציות לינאריות
- הצגת תוצאות ניסוייות
- פתרון משוואות דיפרנציאליות
- מתמטיקה עם המנוע הסימבולי
- סטטיסטיקה ושערוך פרמטרים
- ניצול מידע חזותי
- פתרון מערכות לא לינאריות
- בניית ממשקים עצמאיים
- מידול מערכות דינמיות

מבנה הקורס

- 10 שעות, במסגרת 2 מפגשים
- לימוד התנסותי, אין תחליף להתנסות עצמאית
- הוראה פרונטלית + תרגול
- חומר עזר
 - חוברת עזר של דורי פלג
 - <http://www.mathworks.com/>
 - <http://www.math.ucsd.edu/~bdriver/21d-f99/help--files/matlab-primer.html>
 - <http://matlab.wikia.com/wiki/FAQ>
 - <http://web.eecs.umich.edu/~aey/eecs216.html>
- הפקודה החשובה ביותר עליכם לזכור : help

מהו MATLAB?

- MATrix LABoratory – חזק מאוד בפעולות מתמטיות
- מבוסס שפת C ו-JAVA
- מרבית הפעולות המתמטיות ממומשות בקבצי dll
- Interpreter (אינו מצריך קומפילציה) - מקצר את שלב הפיתוח
- מצוין ל post processing, פחות מומלץ ל real time
- 3 סוגי קבצים:
 - קבצי קוד - .m
 - קבצי נתונים - .mat
 - קבצי גרפים - .fig

סביבת העבודה

MATLAB 7.10.0 (R2010a)

File Edit Debug Parallel Desktop Window Help

Current Folder: E:\11backup\desktop\Blasius

Current Folder

desktop ▸ Blasius

Name

- BL_baseflow.m
- Blasius.m
- cheb.m

Details

Workspace

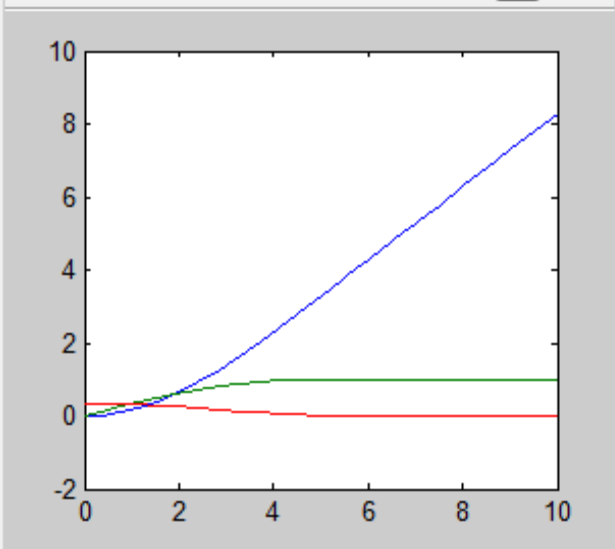
Select data to plot

Name	Value	Min
ans	<1x27 double>	0
eta	<1x27 double>	0
f	<3x27 double>	-1.293

Command History

- format long
- f(1,3)
- %-- 10/29/15 8:53 AM --%
- plot(eta,f)
- [eta,f]=Blasius
- plot(eta,f)

Figures - Figure1



Editor - E:\11backup\desktop\Blasius...

```
1 function [eta,f]=Blasius
2 % Solve Blasius boundary layer
3 eta_inf = 5;
4 eta_infmax = 10;
5
6 eta = linspace(0,eta_inf,100);
7 solinit = bvpinit(eta,[0 1 0]);
8 sol = bvp4c(@ode,@bc,solinit);
9 eta = sol.x;
10 f = sol.y;
11 % figure
12 % plot(eta,f)
13 % plot(eta,f')
```

main.m mappings.m BL_baseflow.m

Command Window

Columns 25 through 27

6.7792	7.2792	8.2792
1.0000	1.0000	1.0000
0.0000	0.0000	-0.0000

```
>> plot(eta,f)
fx >>
```

Start

OVR

פקודות בסיסיות

- (כמעט) כל קוד מתחיל בפקודות הבאות
 - clc, clear all, close all
 - dos הפקודות המוכרות של
 - cd, dir..
 - who, whos
 - format short, format long
 - why
- משתנים
- שליטה על הצגת מספרים
- פקודה לשעות הקשות

מטלב כמחשבון

- סדר פעולות מתמטית : ^ (חזקה), *, / , + -

```
>> -5/(4.8+5.32)^2  
ans = -0.0488
```

- כאשר אנחנו לא מאחסנים את התוצאה במשתנה התשובה מאוחסנת
אוטומטית במשתנה ans
- משתנים מרוכבים

```
>> x = 3+4i % x is stored in workspace and displayed  
>> abs(x) % Absolute value.  
ans = 5  
>> angle(x) % Phase angle (in radians).  
ans = 0.9273  
>> conj(x) % Complex conjugate.  
ans = 3.0000 - 4.0000i  
>> imag(x) % Complex imaginary part.  
ans = 4  
>> real(x) % Complex real part.  
ans = 3
```

אתחול מטריצות ידני

- כל העבודה בשלב זה תתבצע ב- command window (cw)
- אין צורך להגדיר את המשתנים וגודלם מראש
- נגדיר ידנית את מטריצת ה"קסם" הבאה:

```
>>A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

- שימו לב למשתנה שנוצר ב workspace

```
A =  
16 3 2 13  
5 10 11 8  
9 6 7 12  
4 15 14 1
```



אתחול מטריצות ידני

- סימן " ; " בתוך המטריצה מורה על סוף שורה
 - הוספת " ; " בסוף הפקודה מונע את הצגת התשובה לפעולה ב-cw
 - בין אברי השורה ניתן להשתמש בפסיק או ברווח
 - אתחלו את המטריצה הבאה :
- ```
>>A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```
- מטריצה ריקה [] - סוגריים ריקים נותנים מערך בגודל 0x0 :
- ```
>>B = [ ]; % empty matrix
```

עבודה עם איברי מערכים

- אינדקסים של איברי מטריצה

A =

		Columns (n)				
		1	2	3	4	5
Rows (m)	1	4 ¹	10 ⁶	1 ¹¹	6 ¹⁶	2 ²¹
	2	8 ²	1.2 ⁷	9 ¹²	4 ¹⁷	25 ²²
	3	7.2 ³	5 ⁸	7 ¹³	1 ¹⁸	11 ²³
	4	0 ⁴	0.5 ⁹	4 ¹⁴	5 ¹⁹	56 ²⁴
	5	23 ⁵	83 ¹⁰	13 ¹⁵	0 ²⁰	10 ²⁵

A (2,4)

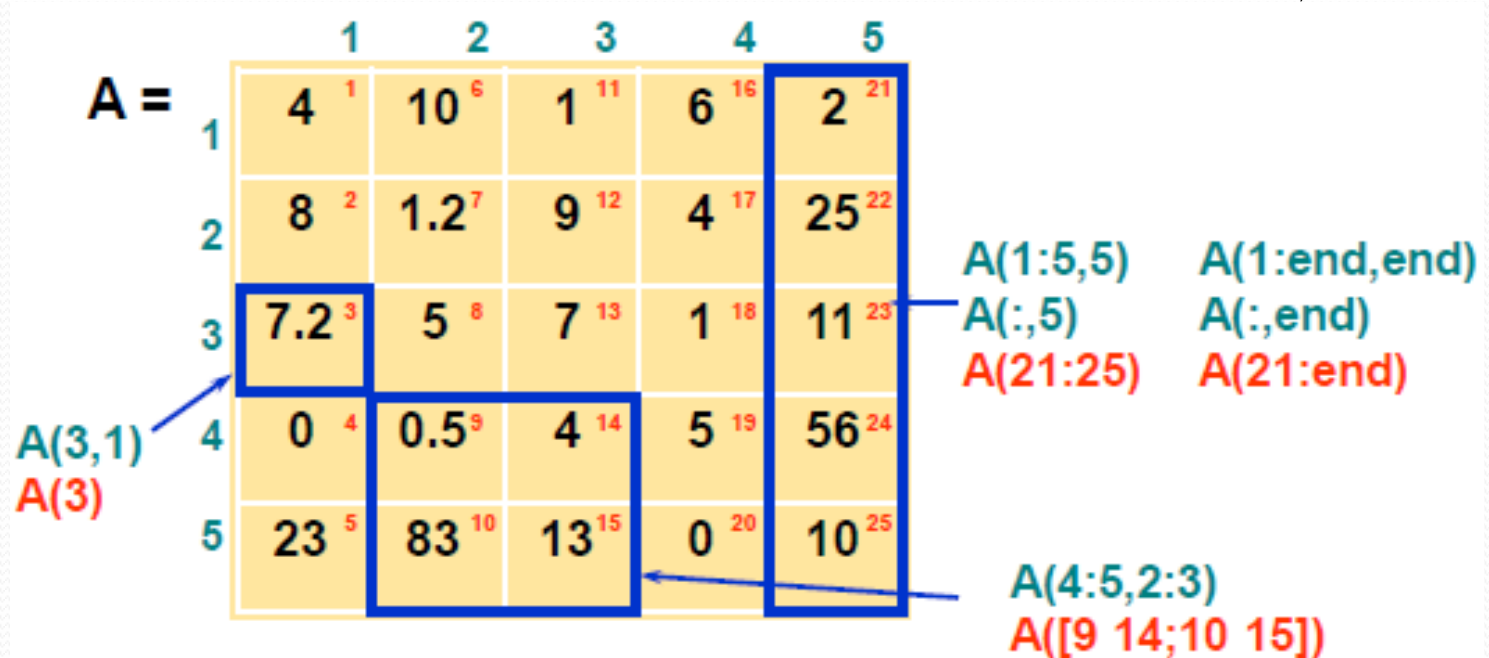
A (17)

Rectangular Matrix:
Scalar: 1-by-1 array
Vector: m-by-1 array
1-by-n array
Matrix: m-by-n array

- שימו לב, האינדקס הראשון הוא 1 ולא 0

עבודה עם איברי מערכים

- אינדקסים של איזורים במטריצה



- אופרטור " : " פורש את כל האינדקסים הקיימים במימד הנתון
- אופרטור end בתור אינדקס נותן את הערך המקסימלי במימד הנתון

עבודה עם איברי מערכים

שימושי האופרטור " : "

- קריאה לשורה/עמודה שלמה
`>>x = A(2,:); % 2nd row, all columns`
- פורש כל מערך על פי סדר המימדים שלו לוקטור עמודה
- מניעת שגיאות – כאשר אין וודאות בכיוון וקטורים הניתנים ככניסה לפונקציה. למשל, אם רוצים לצרף ווקטור בתור עמודה נוספת למטריצה קיימת באותו אורך:
`total_mat = [total_mat user_vector(:)];`
- נחזור למטריצה A :
`>>A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]`
- מה התוצאה של הביטויים?
`>>A(3,2) A(1,:) A(:,2) A([1 2],1) A(1,[1 end])`
- צרו וקטור עמודה C ווקטור שורה R, המכילים את העמודה והשורה השלישית של A בהתאמה

אתחול מטריצות ידני

- ניתן להגדיר מטריצות ע"י שמות משתנים קיימים או ביטויים:

```
>> a=2;  
>> A = [ a R exp(0) j^2 3+5 ]  
      A = 2 9 6 7 12 1 -1 8  
>> B=[A ; A]  
      B = 2 9 6 7 12 1 0-1i 8  
          2 9 6 7 12 1 0-1i 8
```

- איזה מהביטויים ייתן תוצאה?

```
>> D=[R; C] or >>D=[R transpose(C) ]
```

- מחקו את המתשנים שהוגדרו עד כה:

```
>> clear A B R C D  
>> clear all % shorter, but could be dangerous inside a script
```

אתחול מטריצות אוטומטי

- קבלת מימדי מטריצה (מקרה כללי)

```
>>A = [ 1 2 3 ; 4 5 6];  
>>[m,n] = size(A)  
m = 2 n = 3  
>> len = length(A) % length(A) = max(size(A))
```

- פונקציות להגדרת מטריצות

A = zeros(m,n)	• מטריצת אפסים – יעיל להקצאת מקום בזכרון
A = ones(m,n)	• מטריצת אחדות
A = eye(n)	• מטריצת יחידה
A = rand(m,n) A = randn(m,n)	• מטריצות של מספרים אקראיים
	• מטריצה אלכסונית

```
>> d = [4 1 9 2.5]'; A = diag(d)  
A =  
4 0 0 0  
0 1 0 0  
0 0 9 0  
0 0 0 2.5
```

אתחול מטריצות אוטומטי

- יצירת וקטור עוקב:

• וקטור עולה במרווחי יחידה: $x = s:d:f \rightarrow x=[s \ s+d \ s+2d \ \dots \ s+(n-1)d]$

```
>> x = 1:4 -> x= 1 2 3 4
```

```
>> x = 0:10:100
```

```
x = 0 10 20 30 40 50 60 70 80 90 100
```

```
>> x = 10:-1:1
```

- ניתן לקבוע את גודל הקפיצות:

- כדי לקבוע וקטור יורד יש להגדיר קפיצות שליליות

- נסו את הביטויים הבאים:

$$x = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 13 & 14 & 15 & 16 & 17 \\ 11 & 13 & 15 & 17 & 19 \\ 1 & 0.8 & 0.6 & 0.4 & 0.2 \\ 9 & 8 & 7 & 6 & 5 \end{bmatrix}$$

- מה יהיה האיבר האחרון בוקטור: $0:3:10$

- מה תהיה תוצאת הביטוי: $10:5$

- הזינו בצורה הקצרה ביותר את המטריצה:

- היעזרו בפקודה `round` ו-`rand` וצרו וקטור עמודה אשר מתחיל בערך 1 ונגמר בערך 10 והינו בעל אורך (מספר איברים) אקראי בין 1 ל-100

- `rand(m,n)` – מטריצה בגודל $m \times n$ של איברים בפילוג אחיד בתחום $[0 \ 1]$

- `randn(m,n)` – מטריצה בגודל $m \times n$ של איברים בפילוג נורמלי (גאוסייני) ממוצע 0 ושונות 1

עבודה עם איברי מערכים

- ניתן לשלוט במדויק בכמות האיברים ע"י הפקודה `linspace(s,f,N)`
- שימוש ב `linspace` מקשה על השליטה בגודל המדויק של מרווח הסדרה, פרמטר מאוד חשוב בתחום עיבוד האותות והנומריקה

```
>>x=0.1:0.1:1;
```

- דצימציה – לקיחת כל איבר שני שביצענו היא למעשה דגימה/דצימציה בפקטור 2

```
>>x_sampled = x(1:2:end)
```

- דילול – הפעולה ההפוכה לדצימציה הינה ריווח הסדרה הנתונה

```
>>x_dilul(1:2:2*length(x)) = x
```

- היפוך סדר מערך

```
>>x_rev = x(end:-1:1)
```


פעולות אריתמטיות על מטריצות

- ניתן לבצע את כל הפעולות האלגבריות תוך שמירה על מימדים נכונים :
- פעולת חיבור/חיסור

```
>> A = [1 2 ; 3 4]
>> B = A + 1
>> B = A + [1 1 ; 1 1]
```

- שתי הפעולות שקולות. חיבור מטריצות במימדים שונים מותר רק כשאחת מהן היא סקלר.

- משימה : הוסיפו את וקטור השורה $[2+i \ 1/3]$ לשורה הראשונה במטריצה A

- אופרטור שחלוף (conjugate transpose = '). שימושי כי נהוג להגדיר וקטור כעמודה

- $A' = \text{transpose}(A)$ ממשיית A

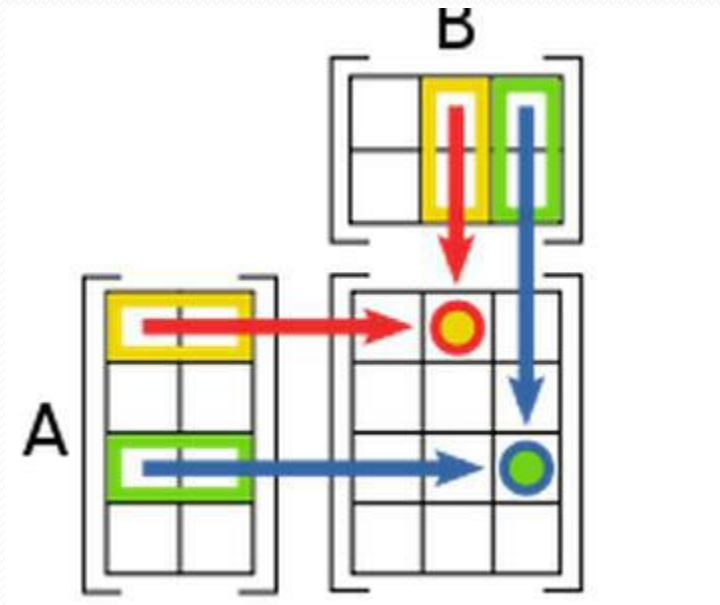
- $A.' = \text{transpose}(A)$ מרוכבת A

- מכפלת מטריצות בסקלר :

```
>> B = 2*A
```

$$B = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

כפל מטריצות



פעולות אריתמטיות על מטריצות

- כפל מטריצות

```
>> A = [1 2 ; 3 4 ; 5 6]
>> B = [7 8 ; 9 10 ; 11 12]
>> C = A * B'
```

$$C = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 7 & 9 & 11 \\ 8 & 10 & 12 \end{bmatrix} = \begin{bmatrix} 23 & 29 & 35 \\ 53 & 67 & 81 \\ 83 & 105 & 127 \end{bmatrix}$$

- כפל מטריצות/וקטורים

```
>> a = [1 2 3]'; b = [10 20 30]'
```

```
>> c = a' * b
```

$$C = [1 \ 2 \ 3] \cdot \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix} = 140$$

- מכפלה פנימית $\langle a, b \rangle$:

```
>> c = a * b'
```

$$C = \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix} \cdot [1 \ 2 \ 3] = \begin{bmatrix} 10 & 20 & 30 \\ 20 & 40 & 60 \\ 30 & 60 & 90 \end{bmatrix}$$

- מכפלה חיצונית:

תרגילים

- בעזרת הוקטור $x=1:10$ יצרו מטריצה kefel שהיא לוח הכפל
- השמו את האיברים האי-זוגיים של השורה החמישית של kefel בוקטור הנקרא r2
- אפסו ערכי kefel בעמודות הזוגיות
- מחקו את השורה והעמודה האחרונות של kefel, כלומר שנו אותה לגודל 9×9

פעולות אריתמטיות על מטריצות

- $\text{rank}()$ - מציאת דרגה של מטריצה

- $\text{det}()$ - מציאת דטרמיננטה

- היפוך מטריצה ריבועית

- פונקצית inv או העלאה בחזקה $\text{inv}(A)$ or $A^{(-1)}$ `>>A = [1 2; 3 4];`

- חלוקה שמאלית - \backslash

- חלוקה ימנית - $/$ (פחות שימושי)

- תרגילים

- חשבו את הביטוי $B^{-1}A$ בעזרת אופרטור החלוקה כאשר $B=[5 \ 1; 6 \ 4]$

- מצאו את פתרון מערכת המשוואות $Ax=[4; 2]$

- מצאו את פתרון מערכת המשוואות $xA=[5 \ 7; 6 \ 8]$

- חזרו על הסעיף הקודם בעזרת חלוקה שמאלית. רמז: $(xA)^T=A^Tx^T$

פעולות אריתמטיות על מטריצות

סינגולריות של מטריצות

- "מספר המצב" – באופן מעשי, מטריצות יכולות להיות קרובות לסינגולריות והפיכתן גוררת שגיאות נומריות גדולות. נהוג לאמוד את "חולי" המטריצות ע"י - `cond()`:

$$\text{מטריצה חולה} = \text{cond}(A) \gg 1$$

```
>> A = magic(4)
```

```
>> rank(A) -> 3
```

הדרגה קטנה מהמימד

```
>> det(A) -> 0
```

ואכן הדטרמיננטה מתאפסת

```
>> inv(A)
```

`inv` מחזיר תשובה עם אזהרה

- האם צריך לסמוך על התשובה שחזרה מהפונקציה?

- אלגוריתם `inv` לא אמור לפעול על מטריצות סינגולריות. אם נבדוק

```
>> cond(A) -> ~1e17
```

פעולות אריתמטיות על מטריצות

- וקטורים עצמיים וערכים עצמיים

$$A\vec{v} = \lambda\vec{v}$$

- למטריצה ריבועית A קיימים ו"ע וע"ע כדרגת המטריצה –

$$\Lambda = V^{-1}AV$$

- לכסון מטריצה מתבצע בעזרת הוקטורים העצמיים

- ישנן מספר פונקציות שמבצעות פירוק ע"ע, הנפוצה הינה – $\text{eig}()$

```
>>A = [1 2 3 ; 0 4 5 ; 0 0 6];
```

```
>>[V,D] = eig(A)
```

V =	1.0000	0.5547	0.5108	D =	1	0	0
	0	0.8321	0.7982		0	4	0
	0	0	0.3193		0	0	6

- קבלו חזרה את A מהמטריצות V ו- D

פעולות אריתמטיות על מטריצות

פונקציות מטריציות נוספות:

- קבלת מטריצה מדורגת `rref()`
- עכבת מטריצה `trace()`
- פירוק מטריצה למכפלת משולשת עליונה ומשולשת תחתונה `lu()` – ($A=L*U$)
- היפוך מטריצה לא ריבועית `pinv()`

פעולות חד ממדיות נפוצות:

- סכום `sum(X,dim)`, אם `dim` לא נתון, הפנק' תפעל על המימד הראשון שאינו באורך 1
- סכום מצטבר `cumsum()`
- ממוצע `mean()`
- כפל `prod()`
- מה יהיה מימד התוצאה של `prod([1 2 ; 3 4])`?

1	2	3	4	5	6	7
1	3	6	10	15	21	28

תרגיל

- יישום – הפחתת רעש לבן ממדידות חוזרות
- נדגים את יעילות השימוש במטריצה לשם טיפול במדידות חוזרות בעזרת אות א.ק.ג רועש
- השפעת רעש לבן עם ממוצע אפס יורדת במיצוע מדידות חוזרות בעלות תבנית קבועה
- הורידו את הקובץ sig.mat והעלו אותו ע"י `load sig.mat`
- הקובץ משתנה x המכיל 1000 מדידות של סיגנל באורך 300, ציירו `plot(x)`
- בצעו ממוצע על מנת "לבטל" את הרעש, הכניסו את התוצאה ל-x0
- ציירו את x0 ע"י `hold on; plot(x0, 'k--', 'linewidth', 3)`

תרגיל

- נחזור לאות הא.ק.ג.
- מצאתם את הממוצע של 1000 מדידות ע"י

```
load sig;  
plot(x)  
x0 = mean(x,2);  
hold on;  
plot(x0,'k--','linewidth',3)
```

- בונוס : מהו מספר המדידות הנדרש כדי לקבל שגיאה מכסימלית של 0.01 מ- x_0

```
i=1;  
while max(abs(mean(x(:,1:i),2)-x0))>0.01  
    i=i+1;  
end;
```

פעולות על מערכים – אופרטור הנקודה

- Hadamar's product - הפעולה תבוצע איבר איבר

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{bmatrix}$$

$$C = A * B = \begin{bmatrix} a_{11} \cdot b_{11} & a_{12} \cdot b_{12} & a_{13} \cdot b_{13} & a_{14} \cdot b_{14} \\ a_{21} \cdot b_{21} & a_{22} \cdot b_{22} & a_{23} \cdot b_{23} & a_{24} \cdot b_{24} \\ a_{31} \cdot b_{31} & a_{32} \cdot b_{32} & a_{33} \cdot b_{33} & a_{34} \cdot b_{34} \end{bmatrix}$$

$$C = A / B = \begin{bmatrix} a_{11}/b_{11} & a_{12}/b_{12} & a_{13}/b_{13} & a_{14}/b_{14} \\ a_{21}/b_{21} & a_{22}/b_{22} & a_{23}/b_{23} & a_{24}/b_{24} \\ a_{31}/b_{31} & a_{32}/b_{32} & a_{33}/b_{33} & a_{34}/b_{34} \end{bmatrix}$$

$$C = A.^B = \begin{bmatrix} a_{11}^{b_{11}} & a_{12}^{b_{12}} & a_{13}^{b_{13}} & a_{14}^{b_{14}} \\ a_{21}^{b_{21}} & a_{22}^{b_{22}} & a_{23}^{b_{23}} & a_{24}^{b_{24}} \\ a_{31}^{b_{31}} & a_{32}^{b_{32}} & a_{33}^{b_{33}} & a_{34}^{b_{34}} \end{bmatrix}$$

פעולות על מערכים – אופרטור הנקודה

- מיישם פעולה אריתמטית על כל איבר במערך בנפרד

- ממוקם לפני אופרטור הפעולה המתמטית

```
>>A = [1 2 ; 3 4]
```

```
>>A^2
```

```
ans = 7 10
```

```
15 22
```

```
>>A.^2
```

```
ans = 1 4
```

```
9 16
```

- כאשר משתמשים בפעולה זו, חשוב להקפיד על גודל מערכים זהה

```
>>t = [0:1e-3:10]; % time base
```

```
>>a = 5; % constant scalar
```

```
>>x = [a*t^2]; % won't work. Where shall we place the dot?
```

```
>>x = [a*t.^2];
```

- צרו את סדרת מספרים שמבטאת את $x = \sin(t)/t$

- ציירו: `plot(t,x)`

אופרטורים לוגיים

(or) ||, |, (and) &&, &, (not) ~, <=, >=, <, >, = •

```
>> r=1:5;  
>> ab = logical([1 0 0 0 1])  
>> r(ab)
```

• קריאה לאיברי מערך :

• דרך תנאים על איברי המערך

```
>> r = rand(100,1);  
>> my_cond = [r<0.75 & r>0.25];  
>> r_top = r(my_cond);
```

• דרך תנאים על האינדקסים של המערך

```
>> ind = 1:length(r);  
>> my_cond = [ind/4==round(ind/4) | ind/3==round(ind/3) ]  
>> r( my_cond )
```

• דרך תנאים על מערך מקביל אחר

```
>> t = [0:0.1:50]';  
>> cos_t = sin(2*pi/50*t) .* (t<=max(t)/2);
```

אופרטורים לוגיים

- `any()` , `all()` , `(isreal, isnan) is***()`

- `find()` - מחזירה את האינדקסים עצמם במקום מערך `logicals`

```
>> isreal(1+i);
```

```
>> isnan(0/0)
```

```
>> find(r>0.5)
```

- משימה – צרו את המטריצה :

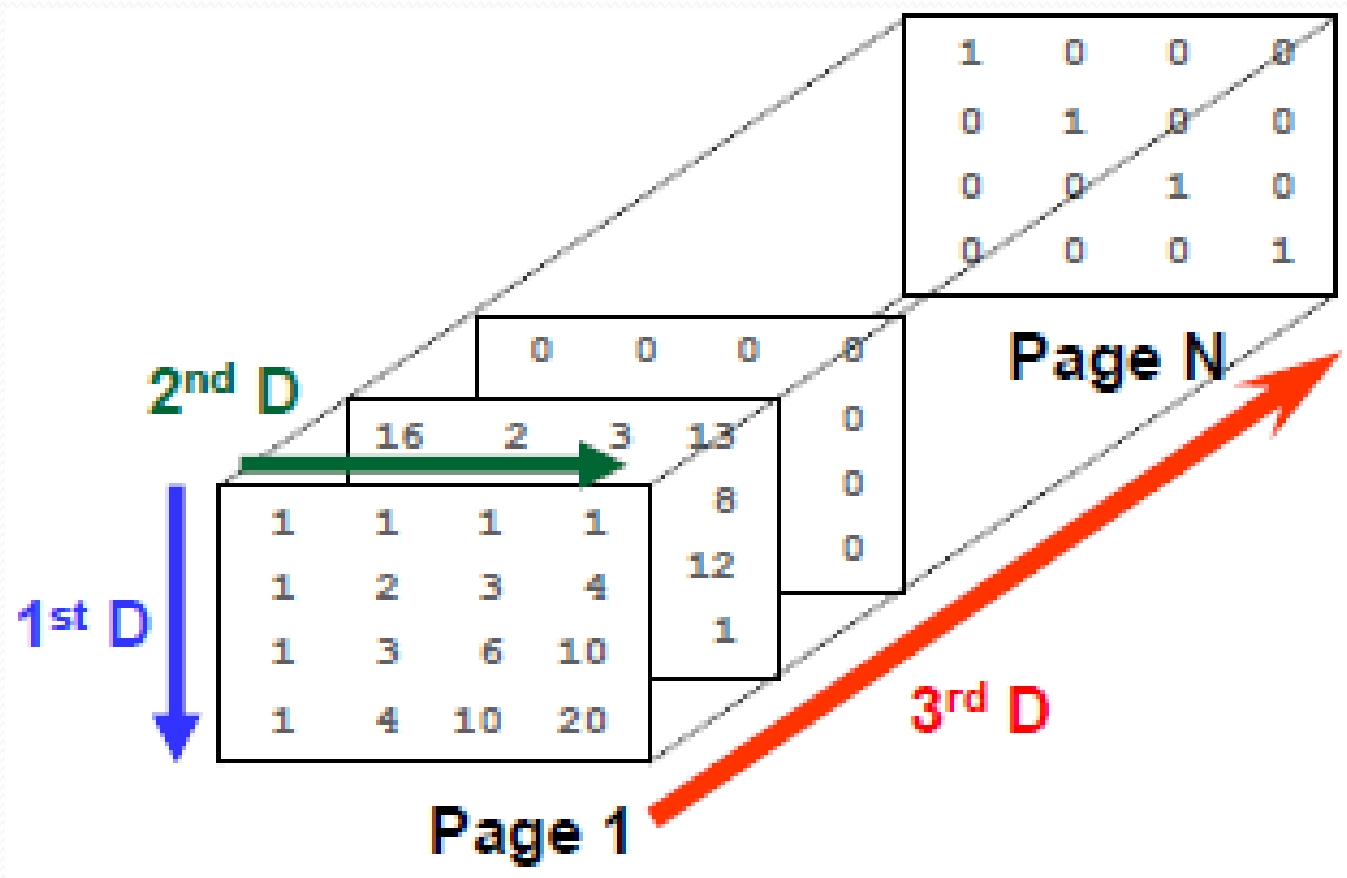
```
>> A = [5 0 2 3 ; 0 5 8 5; 5 3 5 0 ; 9 5 1 1]
```

- שנו את המטריצה כך שכל איבר שאינו שווה ל 5 יקבל את הערך NaN

- שנו את איברי האלכסון של A לערך Inf, ללא שימוש בפקודה `diag` העזרו במטריצות האינדקסים M ו-N אותן תייצרו ע"י :

```
>> [N,M] = meshgrid(1:4)
```

מטריצות מרובות מימדים



מטריצות מרובות מימדים

- אתחול:

- הפניית אינדקסים –

```
>>A(:, :, 1) = pascal(4);  
>>A(:, :, 2) = magic(4);  
>>A(:, :, 4) = diag(ones(1, 4));
```

- שרשור – cat()

```
>>A = cat(dim, A, B);
```

- אתחול רגיל באמצעות פונקציות יצירת מטריצות

```
>> ones(2, 2, 2, 2) % 4D matrix
```

- reshape, meshgrid, repmat - נלמד אותן בהמשך

- התמרת אינדקסים:

- כזכור, אינדקסים מיוצגים באופן רציף, עפ"י סדר המימדים הקיים, או באופן פרטני עבור כל מימד. ניתן להיעזר בפונקציית sub2ind כדי לבצע את ההתמרה (על אף פשטותה)

מטריצות מרובות מימדים

```
>> A1 = pascal(4)
```

```
A1 =
```

1	1	1	1
1	2	3	4
1	3	6	10
1	4	10	20

```
>> A2 = magic(4)
```

```
A2 =
```

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

```
>> W = cat(3, A1, A2)
```

```
W(:, :, 1) =
```

1	1	1	1
1	2	3	4
1	3	6	10
1	4	10	20

```
W(:, :, 2) =
```

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

מטריצות מרובות מימדים

- פנקי sub2ind

שימו לב שהכיוון המוביל

הוא בכיוון העמודה

```
>> A = magic(4)
```

```
A =
```

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

```
>> sub2ind(size(A), 2 ,2)
```

```
ans =
```

```
6
```

```
>> [a,b] = ind2sub(size(A), [6])
```

```
a =
```

```
2
```

```
b =
```

```
2
```

מטריצות מרובות מימדים

- פנקי `repmat(A,m,n)`

שכפול מטריצות

m פעמים בכיוון השורות

n פעמים בכיוון העומדות

```
>> A = [1 2 3]
```

```
A =
```

```
1     2     3
```

```
>> repmat(A, 3 , 1)
```

```
ans =
```

```
1     2     3
1     2     3
1     2     3
```

מטריצות מרובות מימדים

- פנקי `reshape(A,m,n)`

ארגון האיברים מחדש

מה עושה הפקודה

? `reshape(A,length(A),1)`

```
>> A = [ 1 2 3 ; 4 5 6]
```

```
A =
```

1	2	3
4	5	6

```
>> reshape(A, 3 , 2)
```

```
ans =
```

1	5
4	3
2	6

מטריצות מרובות מימדים

- פנק' permute

מארגנת את המטריצה מחדש

עפ"י הכיוונים הרצויים

```
>> A = [ 1 2 3 ; 4 5 6]
```

A =

1	2	3
4	5	6

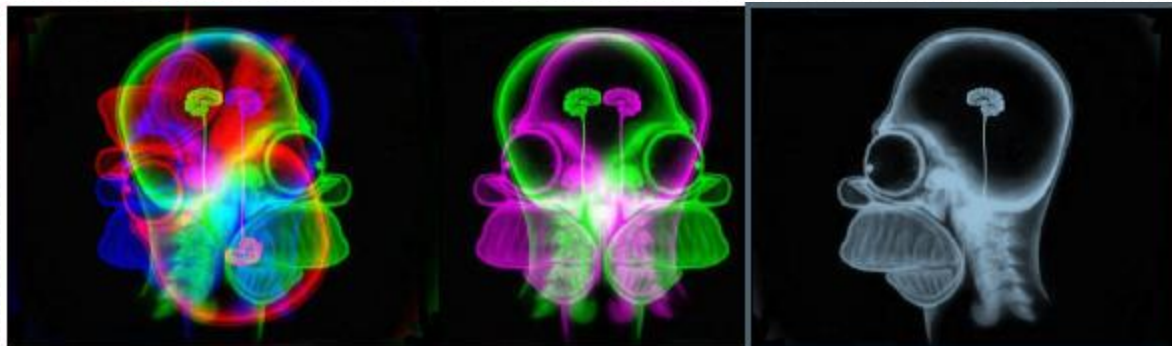
```
>> permute(A , [2 1])
```

ans =

1	4
2	5
3	6

מטרירות מרובות מימד

- משימה : מניפולציה על תמונת RGB
- העלו את תמונת patient.jpg, ניתן לבצע זאת ע"י גרירת הקובץ ואישור.
- וודאו כי המטריצה patient מופיעה ב workspace. בחנו את מימדיה ומאפייניה
- בחנו את תמונת המטופל ע"י image(patient)
- באמצעות שימוש באינדקסים, הפכו את כיוון פני המטופל מצד שמאל לצד ימין
- וודאו את התוצאה בשימוש image
- באופן דומה, שנו את הכיוון האופקי של התמונה רק של תמונת הצבע הירוק (ה-page השני מתוך השלושה). בחנו את התוצאה



תרגילים

תרגיל 1

- צרו באופן הקצר ביותר סדרה הנדסית על בסיס 2, כלומר:

$$a(n+1) = 2 \cdot a(n), n=1, \dots, 10$$

$$a(0) = 1$$

תרגיל 2

- צרו וקטור t באורך 250 מ-0 עד 1, השתמשו ב `linspace`. בעזרת t צרו וקטור סינוס בשם `sin6` שיכיל 6 מחזורים (אל תשכחן להכפיל ב- 2π). הציגו את `sin6` בעזרת `plot(t, sin6)` כאשר ציר ה- x נע בין 0 ל-1.
- כעת צרו וקטור `sin7` בעל 7 מחזורים. צרו מטריצה בגודל 250×2 ש-`sin6` הינו וקטור עמודה אחד שלה ו-`sin7` הינו וקטור עמודה שני. הציגו `plot(t, sins)`

תרגיל 3

- צרו מטריצה שהיא לוח הכפל אך שכל הערכים הקטנים מ-20 והגדולים מ-70 מתאפסים. מצאו את כל המיקומים במטריצה `kefel` בהם הערך גדול שווה ל-81 (שימו לב שהתוצאה מתייחסת ל-`kefel` בתור וקטור ולא בתור מטריצה). להתמרת האינדקסים המתקבלים השתמשו ב-`ind2sub` או כתבו לבד את כלל ההתמרה

תרגילים

תרגיל 4

- צרו מטריצה נורמלית אקראית בגודל $[10 \times 5 \times 3]$ בעזרת הפקודה `randn`. מצאו את המקסימום בערך המוחלט ואת מיקומו בעזרת הפקודות `max`, `abs` ו-`int2sub`.
- בצעו את אותה הפעולה תוך הפעלת הפונקציה `max` פעם אחת בלבד.

תרגיל 5

- צרו סדרת מספרים אקראית אחידה בתחום $[-1, 1]$ באורך 99 בעזרת `A=rand(99,1)`.
- חשבו ממוצע (`mean`) וסטיית תקן (`std`) של כל תת-סדרה המורכבת מכל איבר שלישי. נסו לבצע זאת בפעולה אחת כשהוקטורים מסודרים במטריצה אחת.
- השתמשו בפקודה `sort` כדי למיין את `A` מהמספר הקטן אל הגדול.
- הפכו את כיוון `A` כך שתהיה מהגדול אל הקטן וקחו את חמשת המספרים החיוביים הקטנים ביותר. ראינו קודם כיצד לבצע זאת בעזרת מערכי אינדקסים לוגיים. ניתן ומומלץ, לצורך התרגול, להשתמש בפונקציית `find` כדי למצוא את האיבר השלילי הראשון ולחתוך את הסדרה החל ממנו:

```
first_neg_index = find(A<0,1)
```

תרגילים

תרגיל 6

- צרו מטריצת אחדות A בגודל $[10 \times 10]$ בעזרת `eye`. צרו וקטור עמודה $v=1:10$. נסו להוסיף את הוקטור לכל עמודה של A .
- בעזרת הפקודה `vMat = repmat(v,_,_)` צרו מטריצת שכפולים (בשם `vMat`) של v במימדי מטריצת A . בדקו כי שתי המטריצות באותו הגודל וחברו ביניהן.

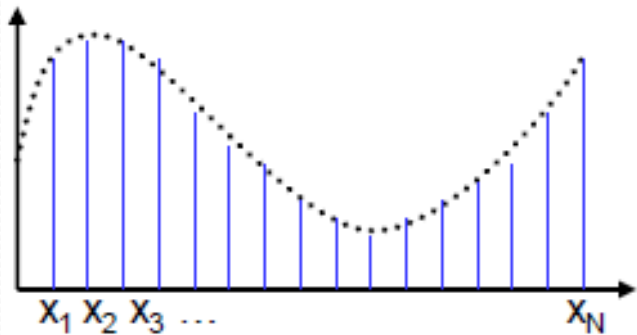
תרגיל 7

- הגדירו מטריצה A במימדים 2×4 שאיבריה נתונים ע"י $A(i,j)=i+j$
- חלצו 2 מטריצות $A1, A2$ בגודל 2×2 ממטריצה A . כאשר $A1$ מכילה את 2 העמודות הראשונות של A ו- $A2$ מכילה את 2 העמודות האחרונות של A
- חשבו את מטריצה B שהיא הסכום של $A1$ ו- $A2$
- חשבו את הערכים העצמיים והוקטורים העצמיים של B
- פתרו את המערכת הליניארית $Bx=b$, כאשר b הינו וקטור שכל אבריו הם 1
- חשבו את הדטרמיננט של B
- חשבו את ההופכי של B
- חשבו את ה-condition number של B

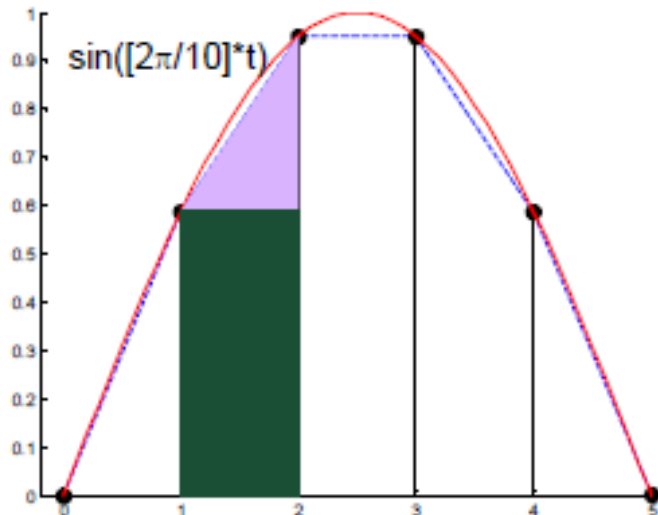
חלק ב'

אינטגרציה נומרית

- אות דיסקרטי מיוצג ע"י סדרת מספרים במיקומים נתונים אשר מהווים את כל האינפורמציה.



- אינטגרציה נומרית מקרבת את הפתרון האנליטי ע"י חישוב סכומי שטחים מקורבים הכלואים בין הנקודות

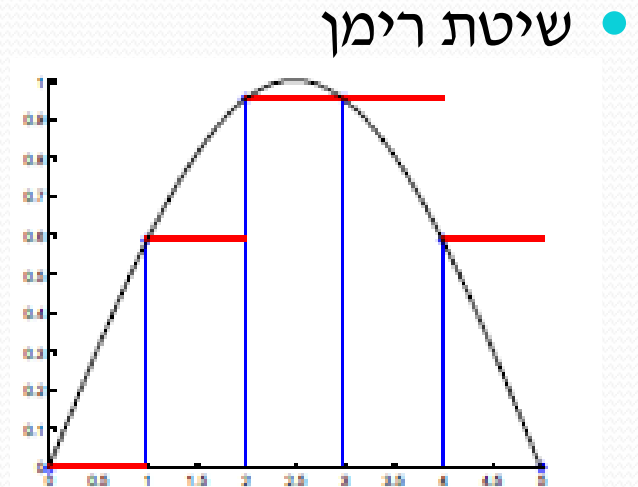


- אינטגרל רימן
- שיטת הטרפז

אינטגרציה נומרית

$$I = \sum_{i=1}^N f(x_i)(x_{i+1} - x_i)$$

```
>> t=0:1:5;  
>> y=sin(2*pi/10*t);  
>> I = sum(y)/pi % dt = 1, const.
```



שיטת הטרפז

ניתן לעשות שימוש בפונקציית הספריה trapz

```
>> I_trapz = trapz(t,y)/pi
```

ב-2 המקרים מתקבל $I=0.9797\pi$

מדוע התוצאה זהה ב-2 המקרים?

גזירה נומרית

- הנגזרת הדיפרנציאלית, אותה מקבלים בהשאפת מרווח הדגימה ל-0 נשארת, עבור סדרות דיסקרטיות, במובן הפרמיטיבי שלה:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \rightarrow \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \quad \text{פרשים קדמיים}$$

- מובן כי בחישוב הפרשים מקטינים את אורך הסדרה. ניתן להימנע מיצירת פאזה בין סדרת המקור לסדרת הפרשים באמצעות הנגזרת המרכזית

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{2h} \rightarrow \frac{f(x_{i+1}) - f(x_{i-1}))}{x_{i+1} - x_{i-1}} \quad \text{פרשים מרכזיים}$$

- מכיוון שלרוב עובדים עם מדידות מעשיות, נגזרת מבליטה רעש נומרי.
- בשל ייצוג מוגבל של תחום הערכים האפשריים, הדיוק הכי טוב שניתן לקבל הוא גודל מרווח הדגימה.
- הכרת פונקציות נומריות מובנות: gradient, diff.

גזירה נומרית

- משימה : עבור האות הבא

```
>> dt=0.05; t=(0:dt:2)';  
>> y=sin(2*pi*t);
```

- הציגו את האות ע"י הפקודה

```
>>plot(t,y); % t,y- same length
```

- חשבו את הנגזרת הראשונה באופן נומרי (כמשתנה dydt) והציגו אותה בעזרת

```
>>hold on; plot(t,dydt,'r') % in red. t,dydt - same length
```

- מהאות dydt חשבו את האינטגרל המצטבר בעזרת cumtrapz או cumsum. הציגו את התוצאה.

- האם הגעתם בחזרה לפונקציה המקורית?

גזירה ואינטגרציה נומרית – אומדן שגיאה

- ההפרש בין סדרת המקור לשחזור: $e = f - f_{app}$

- נחפש פרמטר בודד שיתאר את השגיאה

- מתקבלת שגיאה ממוצעת אפס בשל סימטריות $e_{tot} = mean(e)$

- אומדן שגיאה אבסולוטית, הביטוי אינו גזיר $e_{tot} = mean(|e|)$

- שגיאה ריבועית ממוצעת $e_{tot} = mean(e^2)$

- נרצה לנרמל את השגיאה (השגיאה תלויה באמפליטודה)

- חלוקה איבר איבר תגרום לחלוקה באפס: $e = \frac{f_i - f_{i,app}}{f_i}$ `>> en = (y-y_int)./y`

- חלוקה באנרגית אות המקור מונעת חלוקה באפס (חלוקת mse)

- $$e = \frac{\sqrt{\sum (f_i - f_{i,app})^2}}{\sqrt{\sum f_i^2}}$$
 `>> en = norm(y-y_int)/norm(y)`

טרנספורמציות ליניאריות ב-2D

- הזזה (x_0, y_0) , סיבוב (α) , scaling (s)

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = s \begin{bmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

- נפשט את הביטוי

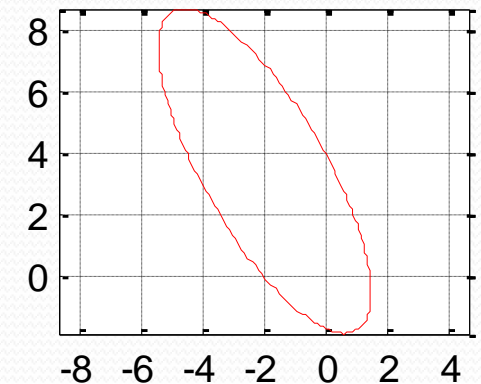
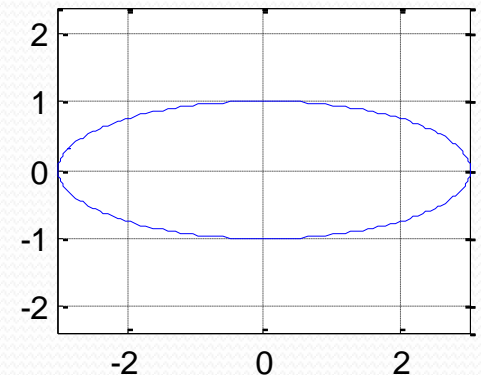
$$v = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = Au + q$$

$$a = s \cos \alpha \quad b = s \sin \alpha \quad q = [x_0; y_0]$$

טרנספורמציות ליניאריות ב-2D

- מימוש נומרי:

```
% define x,y:
N = 200;
t = linspace(0,2*pi,N);
x = 3*cos(t)+1;
y = sin(t)+2;
plot(x,y); % plot x,y:
grid on; hold on; axis equal;
% transformation parameters:
x0 = 2;
y0 = 0;
s = 2;
alpha = pi/3;
% 1. define A, q.
% 2. calculate v with v = A*u+q;
plot(v(1,:),v(2,:))
```



- השלימו את הפקודות בשורות הירוקות

- השתמשו ב repmat על מנת לשכפל את q לאורך המתאים

פונקציות נוספות

- פונקציות שימושיות (בדקו אופן שימוש ע"י help)
- min, max, sort, abs, sign, ceil, floor, fix
- מציאת שורשי פולינום – roots

Example: $13x^3 + 25x^2 + 3x + 4$

```
>> C = [13 25 3 4];
```

```
>> r = roots(C)
```

- פירוק שברים חלקיים – residue

```
>> [R,P,K] = residue([5,3],[1 3 0 -4])
```

$$\frac{5s+3}{s^3+3s^2-4} \rightarrow \frac{-\frac{8}{9}}{s+2} + \frac{\frac{7}{3}}{(s+2)^2} + \frac{\frac{8}{9}}{s-1}$$

מחרוזות

- מחרוזת הינה מערך של תווים מסוג char

- הגדרת פשוטה של מחרוזת: `>>myString = 'Hello World'`

- mystring הוא מערך בגודל [1x11]

- ייצוג ascii

```
>> as = double(myString)
```

```
As = 72 101 108 108 111 32 87 111 114 108 100
```

- המרת קוד ascii : `>> x=char(65) -> x='A'`

- המרה של מספר למחרוזת `num2str(109)` וחזרה `str2num('109')`

- שימושי בעיקר בהקשר של גרפים (title,xaxis...)

- היפוך ושרשור מחרוזות (בדיוק כמו מערכים):

```
>>yourString = fliplr(myString) → 'dlroW olleH'
```

```
>>matString = [myString ; yourString]
```

מחרוזות

- אם נרצה להוסיף כעת שורה שלישית באורך שונה, נשתמש ב- `strvcat` :

```
>>matString = strvcat(matString, 'Hi World')
```

- הפונקציה מרפדת את המחרוזות ברווחים לאורך של המחרוזת הארוכה ביותר.

- ניתן להשתמש באופן דומה בפונקציה `str2mat` (איננה מתעלמת ממחרוזות ריקות) :

```
>> x=str2mat('first','','second','third')
```

- השוואת מחרוזות: `C1 = 'Hello' ; C2 = 'hello' ; C3 = 'hell'`

- אופרטור שוויון לוגי – משווה איבר איבר במערכים :

```
>> C1 == C2      →   ans = 0 1 1 1 1   איך ניתן לנצל תוצאה זו?
```

```
>> C1 == C3      →   error, Matrix dimensions must agree.
```

- פונקציות להשוואת מחרוזות:

```
>>strcmp(C1,C2) → 0    , strcmpi(C1,C2) → 1
```

- חיפוש תת-מחרוזת במחרוזת אחרת:

```
>>findstr(C3,C2) → 1
```

- פונקציה מתקדמת להשוואה וחיפוש במחרוזות: `regexp`

מחרוזות

- עבור המשתנה `temp=25` הציגו מחרוזת (באמצעות הפקודה `disp`) אשר תציג את ערך המשתנה `temp` בתוכה ותיראה כך :

```
the room temperature is 25 degrees
```

```
>>disp(['the room temperature is ',num2str(temp),' degrees'])
```

- הרצת מחרוזת כביטוי (`eval`)

```
>> strx = 'x = 2^3 + exp(0)'
```

```
>> eval(strx) → x = 9
```

מערכי Cell

- מערכים שכל תא בהן יכול להיות מסוג משתנה אחר

מערך בגודל
 2×3 .

כל איבר
במערך נקרא
תא

cell 1,1 <div><table><tr><td>3</td><td>4</td><td>2</td></tr><tr><td>9</td><td>7</td><td>6</td></tr><tr><td>8</td><td>5</td><td>1</td></tr></table></div>	3	4	2	9	7	6	8	5	1	cell 1,2 <div><table><tr><td>'Anne Smith'</td></tr><tr><td>'9/12/94'</td></tr><tr><td>'Class II'</td></tr><tr><td>'Obs. 1'</td></tr><tr><td>'Obs. 2'</td></tr></table></div>	'Anne Smith'	'9/12/94'	'Class II'	'Obs. 1'	'Obs. 2'	cell 1,3 <div><table><tr><td>.25+3i</td><td>8-16i</td></tr><tr><td>34+5i</td><td>7+.92i</td></tr></table></div>	.25+3i	8-16i	34+5i	7+.92i
3	4	2																		
9	7	6																		
8	5	1																		
'Anne Smith'																				
'9/12/94'																				
'Class II'																				
'Obs. 1'																				
'Obs. 2'																				
.25+3i	8-16i																			
34+5i	7+.92i																			
cell 2,1 <div><table><tr><td>[1.43 2.98 5.67]</td></tr></table></div>	[1.43 2.98 5.67]	cell 2,2 <div><table><tr><td>7</td><td>2</td><td>14</td></tr><tr><td>8</td><td>3</td><td>45</td></tr><tr><td>52</td><td>16</td><td>3</td></tr></table></div>	7	2	14	8	3	45	52	16	3	cell 2,3 <div><table><tr><td>'text'</td><td><table><tr><td>4</td><td>2</td></tr><tr><td>1</td><td>5</td></tr></table></td></tr><tr><td>[4 2 7]</td><td>.02 + 8i</td></tr></table></div>	'text'	<table><tr><td>4</td><td>2</td></tr><tr><td>1</td><td>5</td></tr></table>	4	2	1	5	[4 2 7]	.02 + 8i
[1.43 2.98 5.67]																				
7	2	14																		
8	3	45																		
52	16	3																		
'text'	<table><tr><td>4</td><td>2</td></tr><tr><td>1</td><td>5</td></tr></table>	4	2	1	5															
4	2																			
1	5																			
[4 2 7]	.02 + 8i																			

מערכי Cell

- מטריצות שכל תא בהן יכול להיות מסוג משתנה אחר

```
>> A = {'a',1;[20 21; 22 23],true}
```

```
A =
```

```
    'a'                [1]
```

```
 [2x2 double]         [1]
```

- A(1,1) תחזיר cell בגודל 1x1

- A{1,1} תחזיר את הערך של התא (1,1) - במקרה זה string

- פנייה אל תא במערך ואל איבר בתוך התא עצמו ע"י אינדקסים שונים:

```
>>A{2,1}(2,:)
```

- איזו מהפקודות הבאות חוקית?

```
>> A{2,1} = 2      or      A(2,1) = 2
```

- ניתן לאחסן מחרוזות באורך שונה:

```
>> A={'first','second','third'}
```

מערכי Cell

- ניתן להפעיל פונקציות מסוימות על התאים עצמם במקום על מערך התאים:

```
>> length(A) → 3          על מערך התאים  
>> cellfun('length',A) → 5 6 5   על כל תא בנפרד
```

- פונקציות להמרת תאים

```
>>C = {[1] [2 3 4]; [5; 9] [6 7 8; 10 11 12]}
```

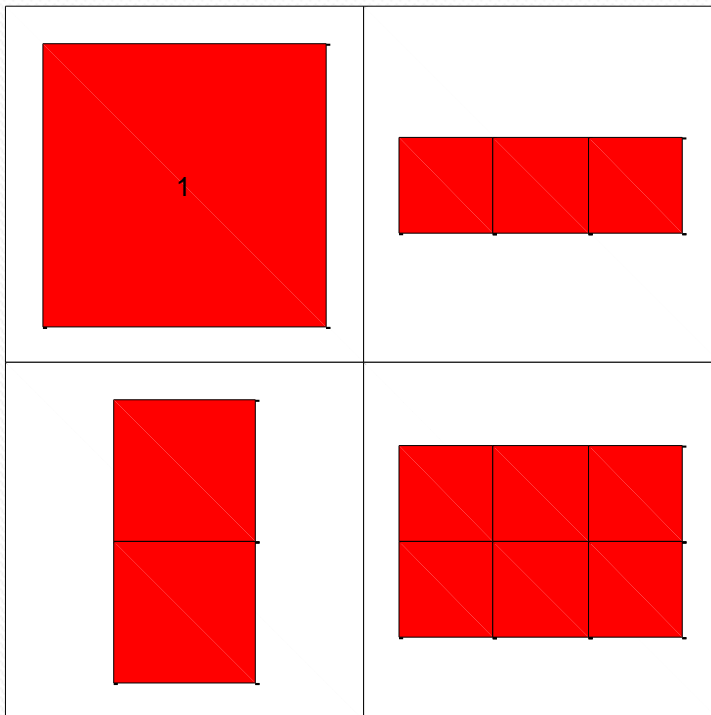
- מה הגודל של C?

- נסו את הפונקציות הבאות:

cell2mat(C)

celldisp(C)

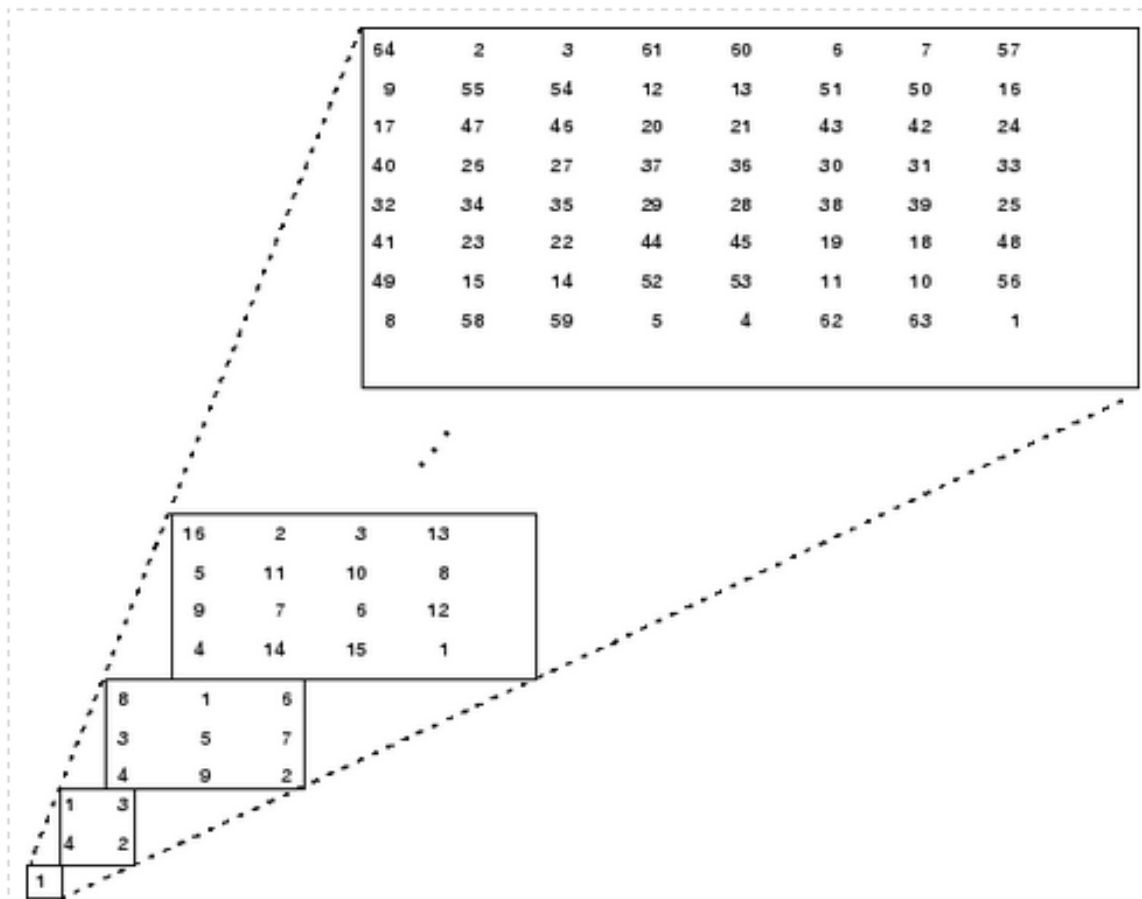
cellplot(C)



מערכי Cell

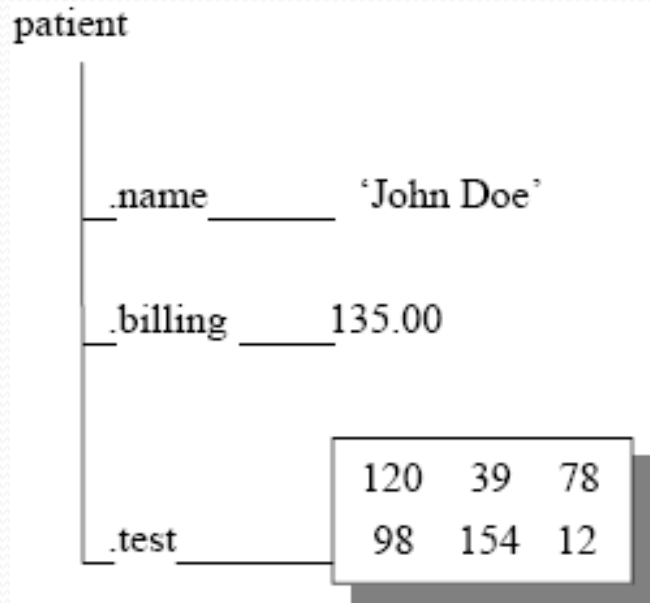
- דוגמא ל-cell בו כל תא הוא ריבוע קסם במימד אחר

```
M = cell(1,8);  
for n=1:8  
    M{n}=magic(n);  
end;  
cellplot(M)
```



מבנים - Structures

- מבנה הוא מערך המכיל טיפוסים משתנים שונים בשדות נפרדים, בעלי שמות מאפיינים.



- שם השדה מופרד בנקודה משם המבנה:

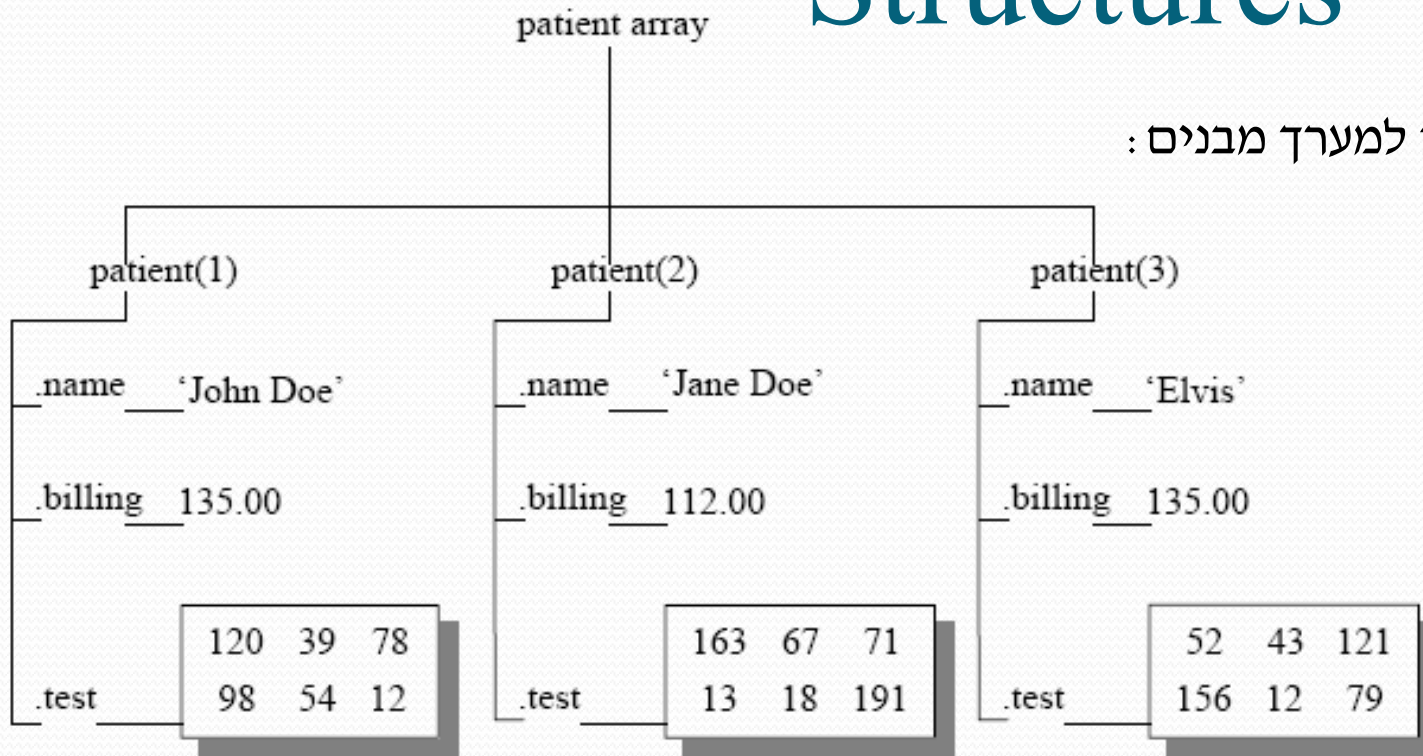
```
» patient.name='John Doe';  
» patient.billing = 135.00;  
» patient.test= [120 39 78;  
                 98 54 12];
```

- יתרונות שימוש:

- טיפול מסודר בנתונים בעלי היררכיה.
- ביצוע חיתוכים בין תכונות (שדות) של אלמנטים שונים במערך.
- השמת כמות אינפורמציה רבה בתוך משתנה ראשי אחד.
- המבנה הוא מערך ומכיל יותר מאלמנט אחד בכל שדה (אם רוצים...)

מבנים - Structures

- הרחבת מבנה למערך מבנים:



- באופן כללי ניתן ליצור מערך מבנים באמצעות:

```
>>StrArray = struct('field1',{values1},'field2',{values2},...)
```

- ערכי השדות ניתנים כתאים ולא מוגבלים לאותו הטיפוס. מימד התאים קובע את מימד המערך.

מבנים - Structures

- שם שדה יכול להיות דינמי, עבור קלט משתמש או תוצאת תוכנית:
- הזינו שם שדה (מחרוזת) ל- `userField` וערך כלשהו ל- `userVal`
- נסו את הביטוי (שימו לב לסוגריים):

```
>>s.(userField) = userVal
```

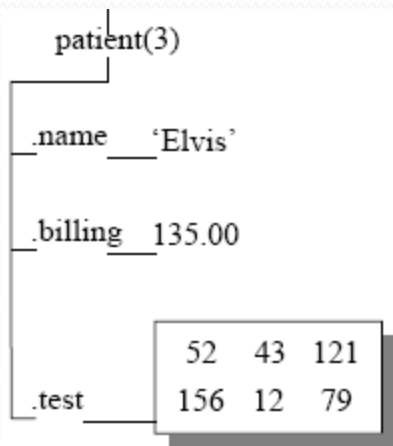
- קבלת פירוט השדות של המבנה

```
>>fieldnames(structname)
```

- מחיקת שדה ממבנה:

```
patient = rmfield(patient,'billing')
```

- גישה לאיבר במערך:



```
>> patient(3)
ans = name: 'Elvis'
      billing: 135
      test: [2x3] double
>> patient(3).test(1,2)
ans = 43
```

מבנים - תרגיל

בתרגיל זה נסרוך ונערוך רשימת פונקציות יעודיות למטריצות בספריית Matlab. המטרה: לאתר את קובץ הפונקציה הקטן ביותר בנפחו. עקבו אחר הפעולות הבאות. ממשו ובחנו כל שורה:

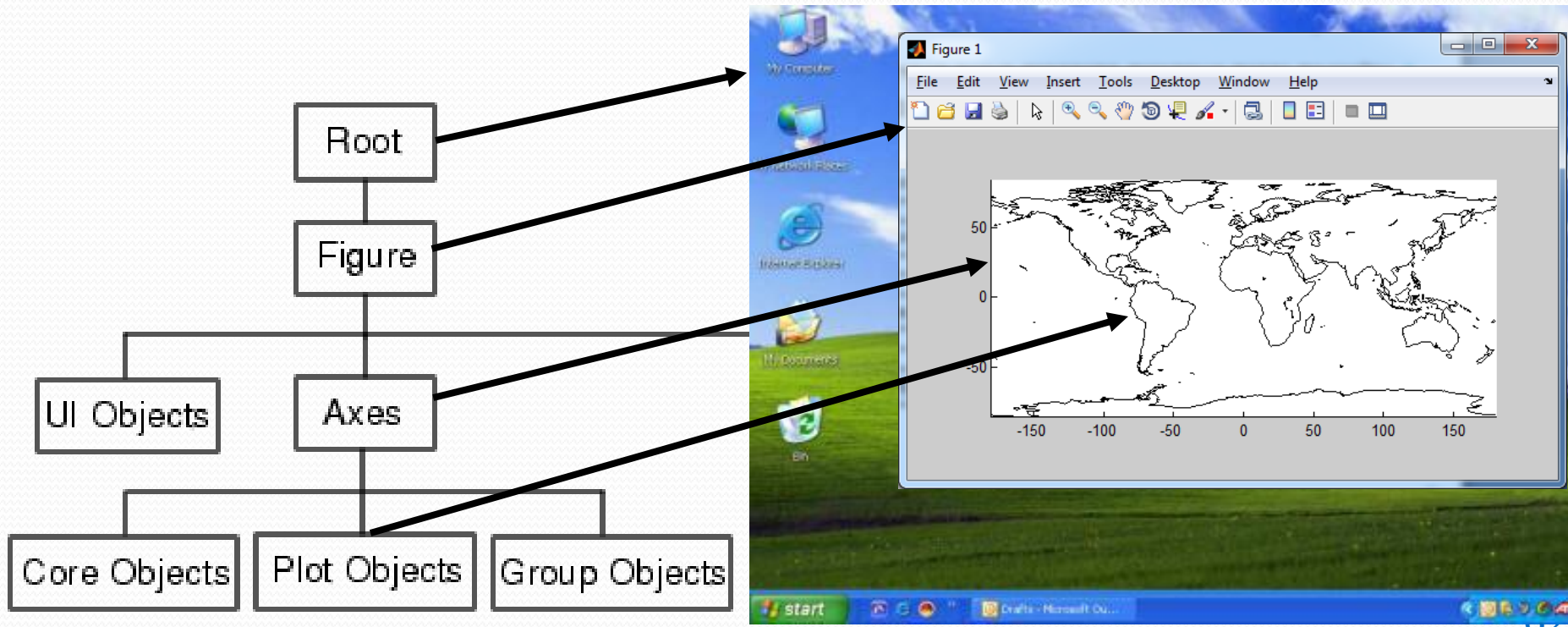
1. הספרייה הנבחרת `path=fullfile(matlabroot,'toolbox\matlab\matfun')`
2. פעולת `dir` מחזירה תשובה לתוך מבנה. בחנו את השדות ותחולתם.
3. `list.isdir` באילו שדות נמצאת האינפורמציה הנחוצה?
4. `IsDir = cell2mat({list.isdir})'` חלק מהאלמנטים הם תיקיות וחלק קבצים.
5. `flist = _____` הפקודה מחזירה אוסף של מספרים
6. `bytes = cell2mat({flist.bytes})'` נהפוך את אוסף המספרים לוקטור
7. `[mVal,mInd] = min(bytes)` אנחנו מעוניינים רק בקבצים.
8. `minFile = flist(____)____` צרו מערך מבנים חדש שלא מכיל תיקיות
9. `fnames = _____({flist.name})'` הגדרת וקטור המכיל את גדלי הקבצים
10. `minFile_mat =` חיפוש הקובץ בעל הגודל הקטן ביותר
- % lets see why it is so small:
- `open(minFile)` השתמשו בתוצאת 7 והשלימו את הפקודה כדי לקבל את שם הקובץ המבוקש
9. צרו מטריצת מחרוזות המכילה את שמות הקבצים (שלב זה לא הכרחי, אך טוב לאימון)
10. פתחו את הקובץ בעל הגודל המינימלי

מבנים - פתרון

```
path=fullfile(matlabroot,'toolbox\matlab\matfun')
list = dir(path)
list.isdir
IsDir = cell2mat({list.isdir})'
flist = list(~IsDir)
bytes = cell2mat({flist.bytes})'
[mVal,mInd] = min(bytes)
minFile = flist(mInd).name
fnames = strvcat({flist.name})'
minfile_mat = fnames(mInd,:)
% lets see why it is so small:
open(minFile)
```

גרפיקה בסיסית

- אלמנטים גרפיים מאורגנים בהיררכיה כאשר היחידה המשורטטת, בעלת האינפורמציה, היא לרוב האובייקט הנמוך ביותר בשרשרת



עריכת אובייקטים

	get properties list	object's handle
Root	get(0)	0
Figure	get(hf)	hf = gcf, hf = figure(1)
Axes	get(ha)	ha = gca ha = axes('Parent',hf) ha = subplot(m,n,k)
Object (line,patch,surface)	get(ho)	ho = gco (if selected) ho = plot(...,'Parent',ha)

get: `get(handle,'PropertyName')` → returns property value

set: `set(handle,'PropertyName',PropertyVal)`

`set(handle,'PropertyName')` → returns property options

inspect: `inspect(handle)` → opens graphic interface

דו-מימד

פונקציית plot

- השימושית והאוטומטית ביותר עבור שרטוט גרפים קווים דו-מימדיים
- מייצרת את כל האובייקטים הדרושים לשרטוט הגרף
- פועלת במספר וריאציות ועל תבניות קלט שונות. הצורה הכללית ביותר:

```
plot(Xdata,Ydata,...,'properties',values,...)
```

- דוגמא

```
>>x = 0:0.1:1; y = x.^.5;
```

```
>>plot(x,y)
```

- הסדרות x,y מאותו האורך.
- נוצר figure ובתוכו axes, אשר תחתיו מצויר אובייקט מסוג line
- מכיוון שלא היה figure קיים, החדש קיבל את המספור 1
- ברירת המחדל היא שרטוט הנקודות בקו רציף (לינארי בין הנקודות) ובצבע כחול
- הצירים הותאמו אוטומטית לתחום ערכי הסדרות המספריות

דו-מימד

- `plot(y)` - כאשר הקלט לפונקציה כולל סדרה אחת בלבד, `plot` מציירת את הסדרה אל מול האינדקסים התואמים.
- הזנת מאפייני אובייקט – אופציה 1, בצורה מקוצרת:

`plot(x, y, 'symbols')`

Symbol	Color
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

Symbol	Line style
—	Solid
:	Dotted
-.	Dash-dot
—	Dashed

דוגמא

`plot(x, y, 'm:d')`

Symbol	Marker
.	.
o	o
x	x
+	+
*	*
s	□
d	◇

דו-מימד

- הזנת מאפייני אובייקט – אופציה 2, קביעת תכונות מפורטות:

```
>>plot(x,y,'property1',val1,...)
```

- כדי לקבל את רשימת התכונות האפשריות:

```
>>h = plot(x,y);
```

```
>>get(h)
```

- דוגמא:

```
>>plot(x,y,'Color','m','LineStyle',':','LineWidth',6,'Marker','d','MarkerEdgeColor','k','MarkerSize',16)
```

- שרטוט מספר מרוכב

```
>>t=0:.01:1; z = exp(j*2*pi*t)
```

```
>>plot(z) % equals: plot(real(z),imag(z))
```

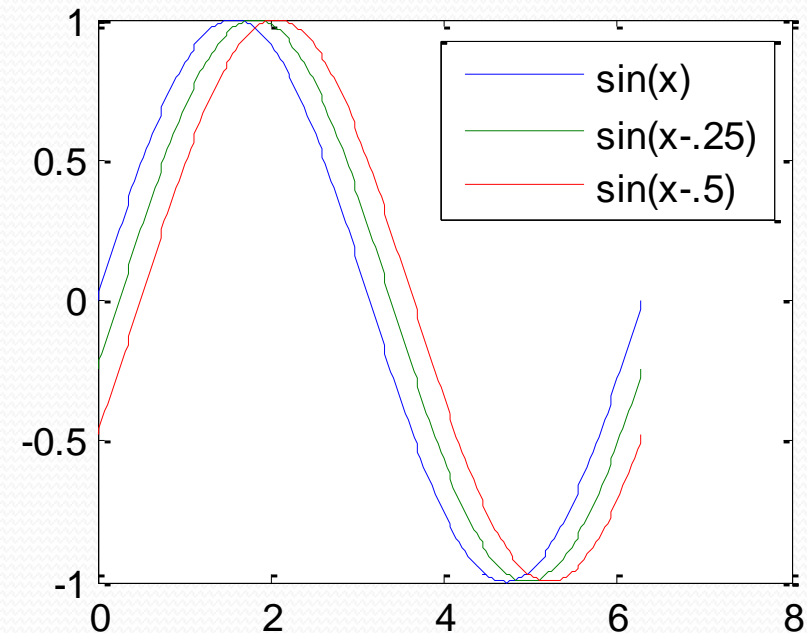
- שרטוט מספר אובייקטים יחדיו

```
>>plot(x1,y1,x2,y2)
```

דו-מימד

- שרטוט מספר אובייקטים יחדיו – אופציה 1 (באותה הפקודה)

```
>>x = 0:pi/100:2*pi;  
>>y = sin(x);  
>>y2 = sin(x-.25);  
>>y3 = sin(x-.5);  
>>plot(x,y,x,y2,x,y3)  
>>legend('sin(x)',...  
'sin(x-.25)', 'sin(x-.5)')
```

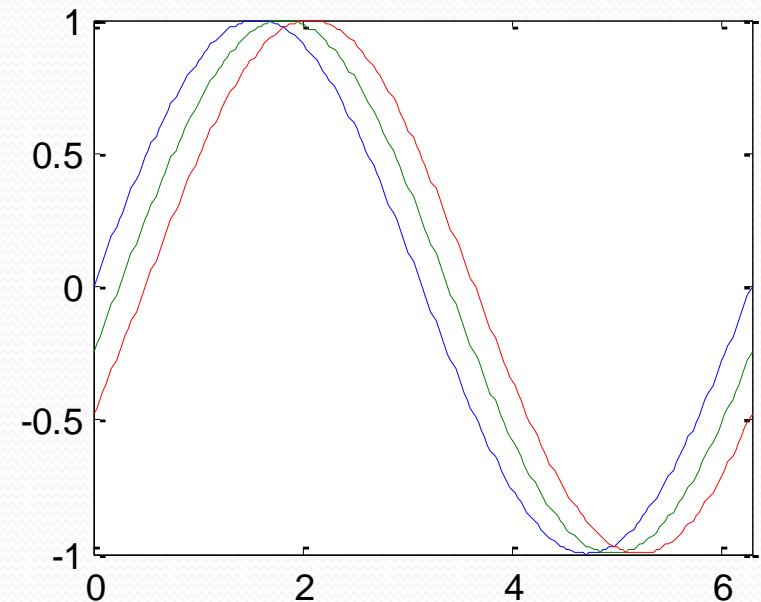


- legend – מייצרת מקרא לגרפים לפי סדר הגרפים
- משימה: שרטטו את $x=1:10$ ו- $y=0:0.5:8$ על גבי גרף אחד

דו-מימד

- שרטוט מספר אובייקטים יחדיו – אופציה 2 (שימוש בhold)

```
>>x = 0: pi/100: 2*pi;  
>>plot(x, sin(x),'b')  
>>hold on  
>>plot(x,sin(x-0.25),'r')  
>>plot(x,sin(x-0.5),'g')  
>>axis tight  
>>hold off
```



- hold – מאפשרת להוסיף אובייקט גרפי נוסף במקום החלפת הישן בחדשו
- axis tight – מצמצם את הצירים למינימום המכיל את הגרפים
- משימה: שרטוט את $x=1:10$ ו- $y=0:0.5:8$ על גבי גרף אחד תוך שימוש בhold

דו-מימד

- שרטוט מספר אובייקטים יחדיו – אופציה 3 (שרטוט עמודות מטריצה)

```
>>x = 0: pi/100: 2*pi;  
>>Mat=[sin(x)' sin(x-0.25)' sin(x-0.5)'];  
>>plot(x,Mat,'LineWidth',3)
```

- הצבעים מחולקים באופן אוטומטי
- LineWidth – שולט על עובי הקו

השוואה בין 3 השיטות

שיטה	אורכים זהים	אורכים שונים	סוגי גרפים שונים
צמדי וקטורים	שליטה בצבעי וסוגי הקווים בלבד	רישום קומפקטי	לא אפשרי
hold	שליטה מלאה בכל אובייקט	אפשרי, אך רישום מסורבל	אפשרי
וקטור ומטריצה	רישום קומפקטי, אך לא ניתן לשלוט בכל מאפייני אובייקט נפרד בגרף	לא אפשרי	לא אפשרי

דו-מימד

- שרטוט שתי סדרות בעלות יחידות שונות:

```
>>[Ax,h1,h2] = plotyy(X1,Y1,X2,Y2)
```

Ax - handle (pointer) to axes

hi - handle to Xi,Yi plot

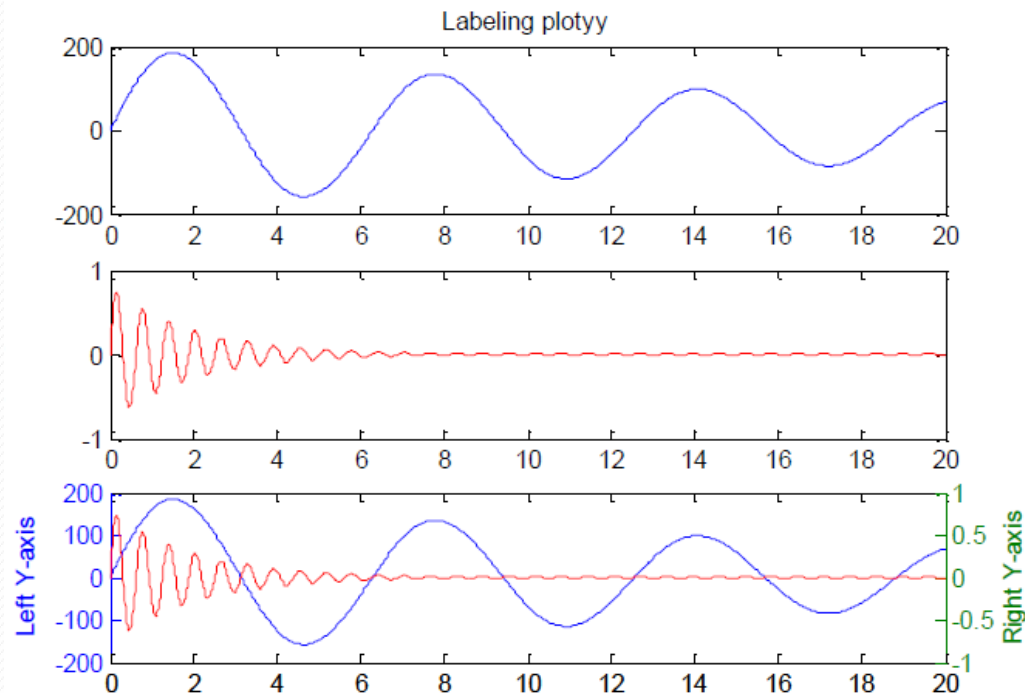
handle לאובייקט ylabel

```
get (AX(2), 'YLabel')
```

מתן כותרות לציר y

```
>>set (get (AX(1), 'YLabel')  
, 'String', 'Left Y-axis')
```

```
>>set (get (AX(2), 'YLabel')  
, 'String', 'Right Y-axis')
```

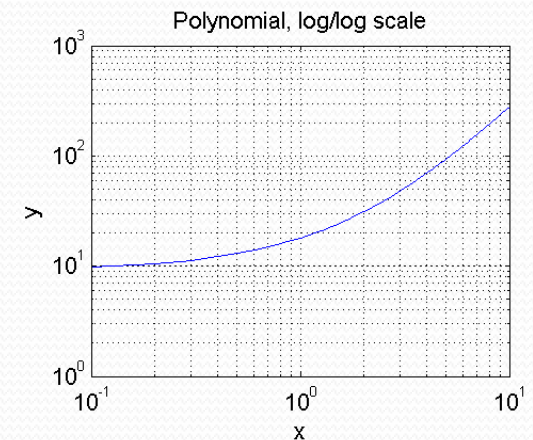
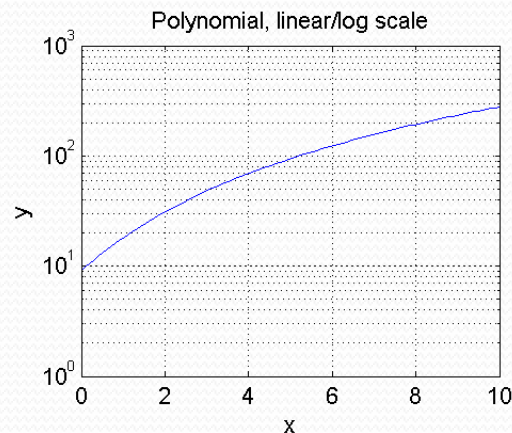
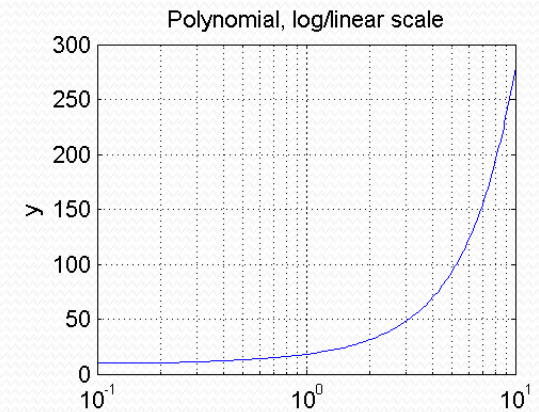
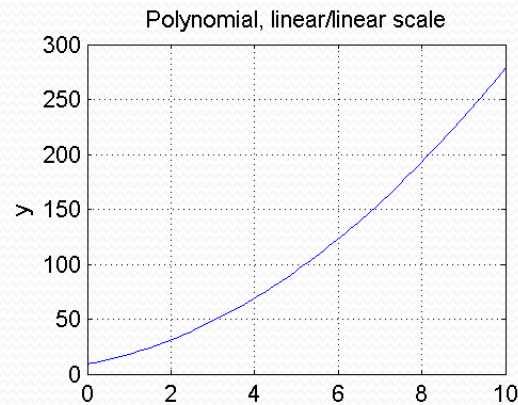


דו-מימד

• שרטוט בסקאלות לוגריתמיות

```
x = 0:10:100;  
y = 2*x.^2 + 7*x + 9;
```

```
plot (x,y)  
semilogx (x,y)  
semilogy (x,y)  
loglog (x,y)
```



פקודות עזר

- שליטה בתחום התצוגה - axis

command	description
axis ([xmin xmax ymin ymax])	Define minimum and maximum values of the axes
axis square	Produce a square plot
axis equal	equal scaling factors for both axes
axis tight	Scale axis to fit plot tightly
axis normal	turn off axis square, equal
axis (auto)	return the axis to defaults

- בחנו את הפקודה הבאה :

```
>>plot(2,2,'c*',1,2.4,'ks',0.2,0.5,'r^','MarkerSize',10)
```

- נסו לשנות את תחום הצגת הצירים : האם התמונה ברורה יותר?

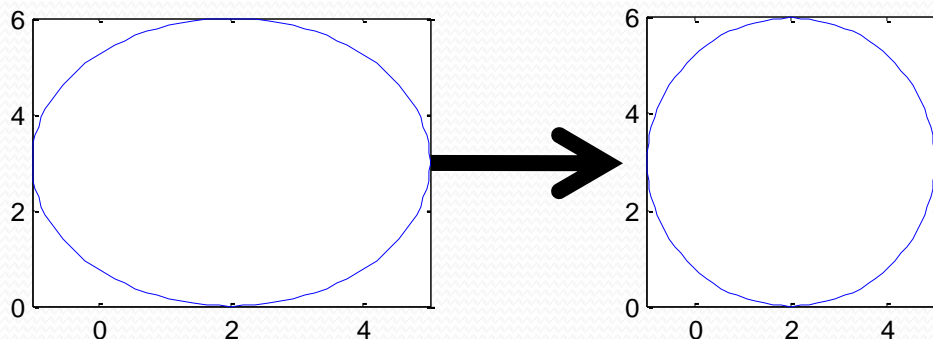
```
>>axis([0 2.5 0 3])
```


פקודות עזר

- תרגיל: ציירו מעגל שמרכזו בנקודה (2,3) ורדיוסו שווה 3

• תזכורת: $x = x0 + R*\cos(\theta); \quad y = y0 + R*\sin(\theta)$

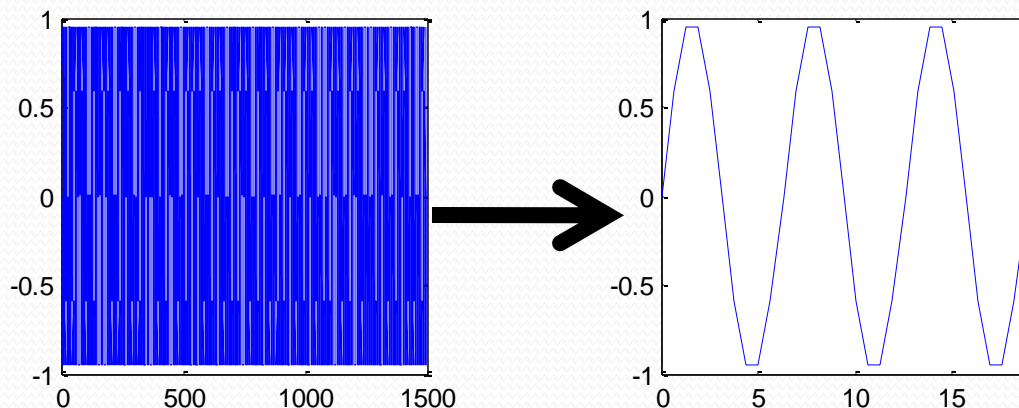
- הצורה המתקבלת אינה מזכירה עיגול במבט ראשון. נסו את הפקודות `axis equal` ו-`axis square` לשיפור התצוגה



- `xlim,ylim` - שולטים בגבולות הצירים בנפרד.

- שימו לב – הפעולה אינה חותכת את האות

```
>>t=0:2*pi/10:1500;  
>>plot(t,sin(t))  
>>xlim([0 6*pi])
```

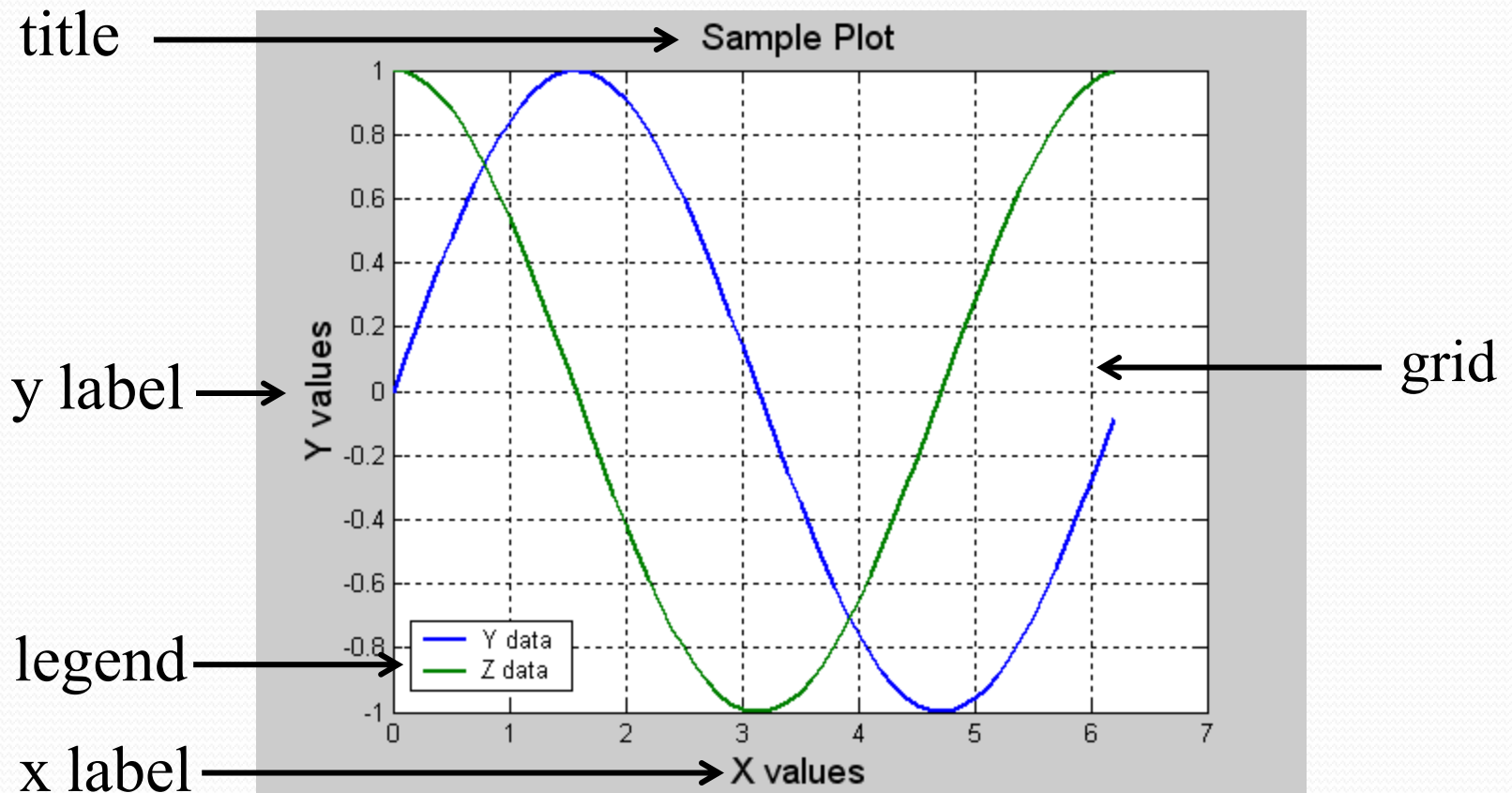


פקודות עזר

- תוספות ל axes
 - axis labels = xlabel, ylabel
 - title
 - grid on, off, minor
- מקרא – legend - יוצר אובייקט חדש מסוג axes
- ניתן להציב בטקסטים סימנים מיוחדים המפוענחים לפי קוד
 - לדוגמא 10^5 , α , \rightarrow , \leftarrow
- כל הטקסטים הם אובייקטים מסוג text והינם "ילדים" של ה-axes.

פקודות עזר

```
>>x=[0:0.1:2*pi];  
>>y=sin(x); z=cos(x);  
>>plot(x,y,x,z,'linewidth',2)  
>>title('Sample Plot','fontsize',14)  
>>xlabel('X values','fontsize',14)  
>>ylabel('Y values','fontsize',14)  
>>legend('Y data','Z data','fontsize',3)  
>>grid on
```

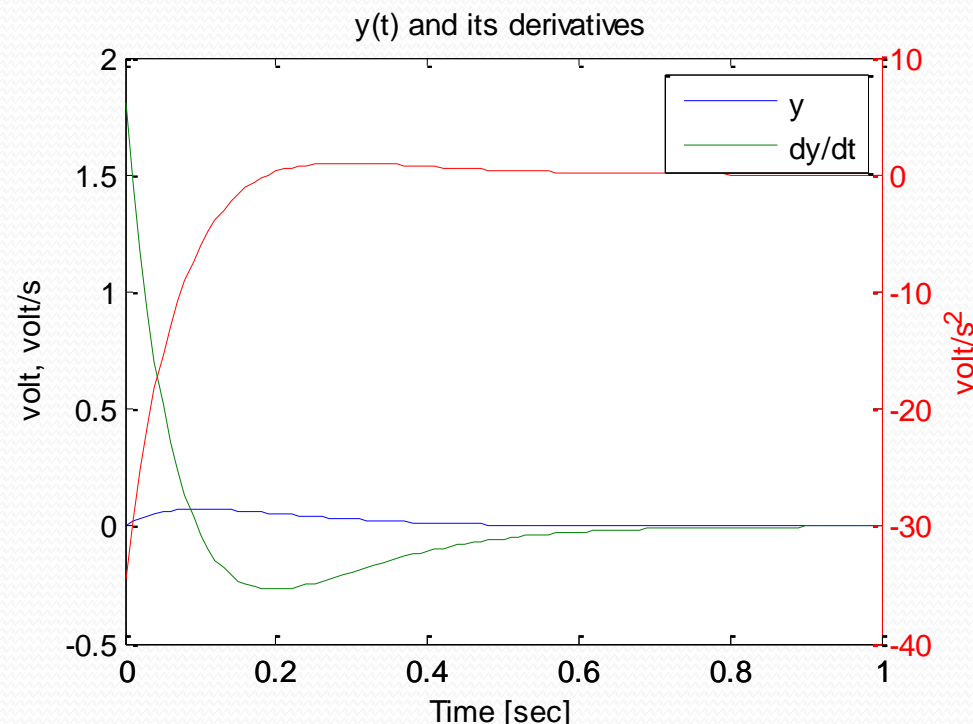


משימה

- שרטטו על אותו הגרף את הפונקציה הבאה ושתי נגזרותיה (ראשונה ושניה)
- אין דרישה לנגזרות מרכזיות
- הקפידו על צבעים שונים
- תנו כותרות מתאימות
- הוסיפו מקרא (legend)

$$t = 0:0.01:1[\text{sec}]$$

$$y(t) = 2t \cdot e^{-10t} [\text{volt}]$$



החזרת מצביע לאובייקט הציור:
`h = plot(...)`
`[hx,h1,h2] = plotyy(...)`
קבלת כל התכונות של אובייקט:
`get(h)`
הפיכת ציר $hx(1)$ לפעיל:
`axes(hx(1))`
הפיכת ציר $hx(2)$ לפעיל:
`axes(hx(2))`
שינוי תכונה של אובייקט:
`set(h,'property',propval)`

פתרון

```
dt = 0.01;
t = 0:dt:1;
y = 2*t.*exp(-10*t);
dydt = diff(y)/dt;
dydt(end+1)=dydt(end); %add last value
d2ydt2 = diff(y,2)/dt^2;
d2ydt2(end+1:end+2)=d2ydt2(end); %add last value

[hx,h1,h2]=plotyy(t,[y;dydt],t,d2ydt2);
xlabel('Time [sec]')
axes(hx(1));
ylabel('volt, volt/s')
title('y(t) and its derivatives')
legend('y','dy/dt')
axes(hx(2));
ylabel('volt/s^2')
```

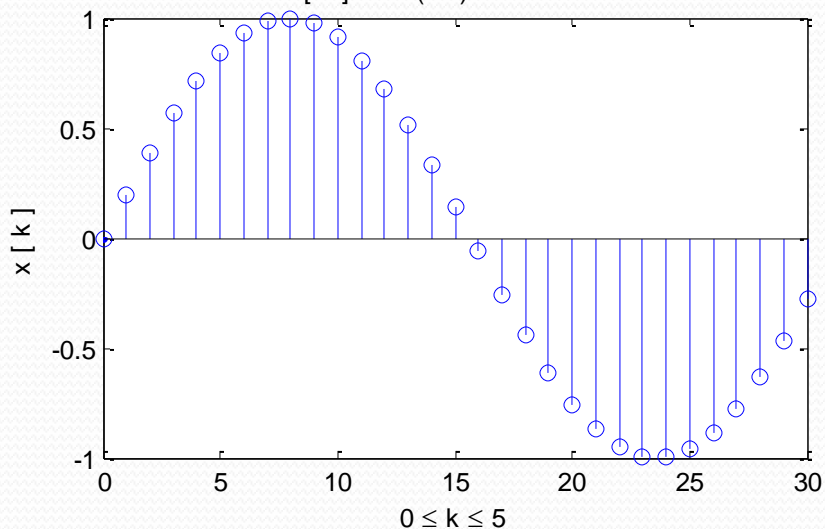
פונקציות נוספות

• שרטוט אות בדיד (stem, stairs):

```
>>k=[0:30];  
>>x=sin(k/5);  
>>stem(k,x)      or      stairs(k,x)  
>> xlabel('0 \leq k \leq 5');  
>> ylabel('x [ k ]');  
>> title('x[ k ] = sin(k/5) for 0 \leq k \leq 5');
```

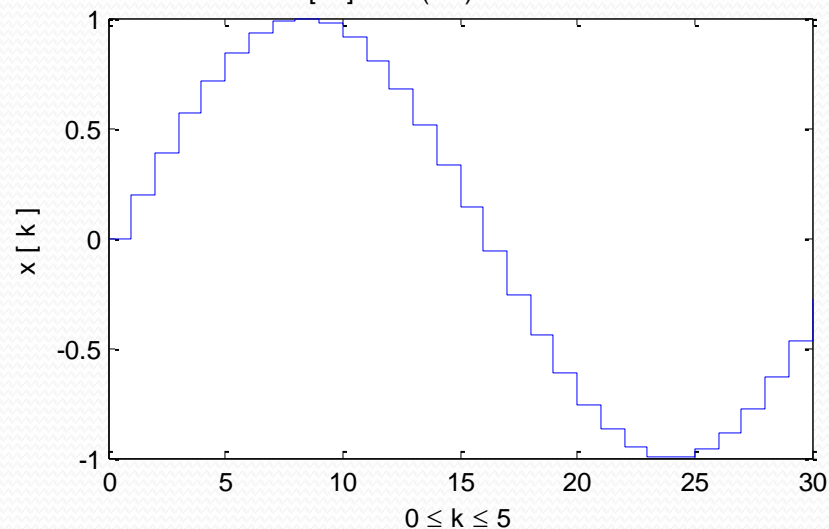
Stem

$x[k] = \sin(k/5)$ for $0 \leq k \leq 5$



Stairs:

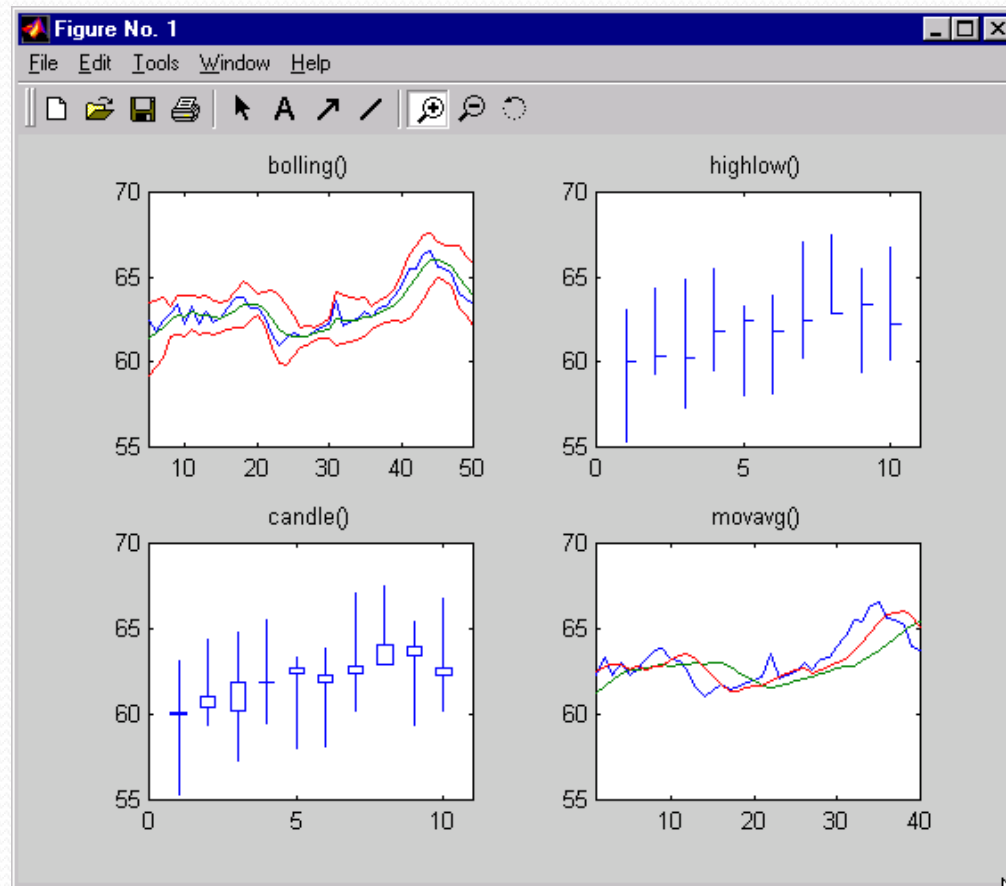
$x[k] = \sin(k/5)$ for $0 \leq k \leq 5$



ארגון גרפים

- ניתן לצייר מספר axes ב-figure אחד באמצעות הפקודה subplot

`subplot(rows, cols, index)`



Subplot(2,2,1)

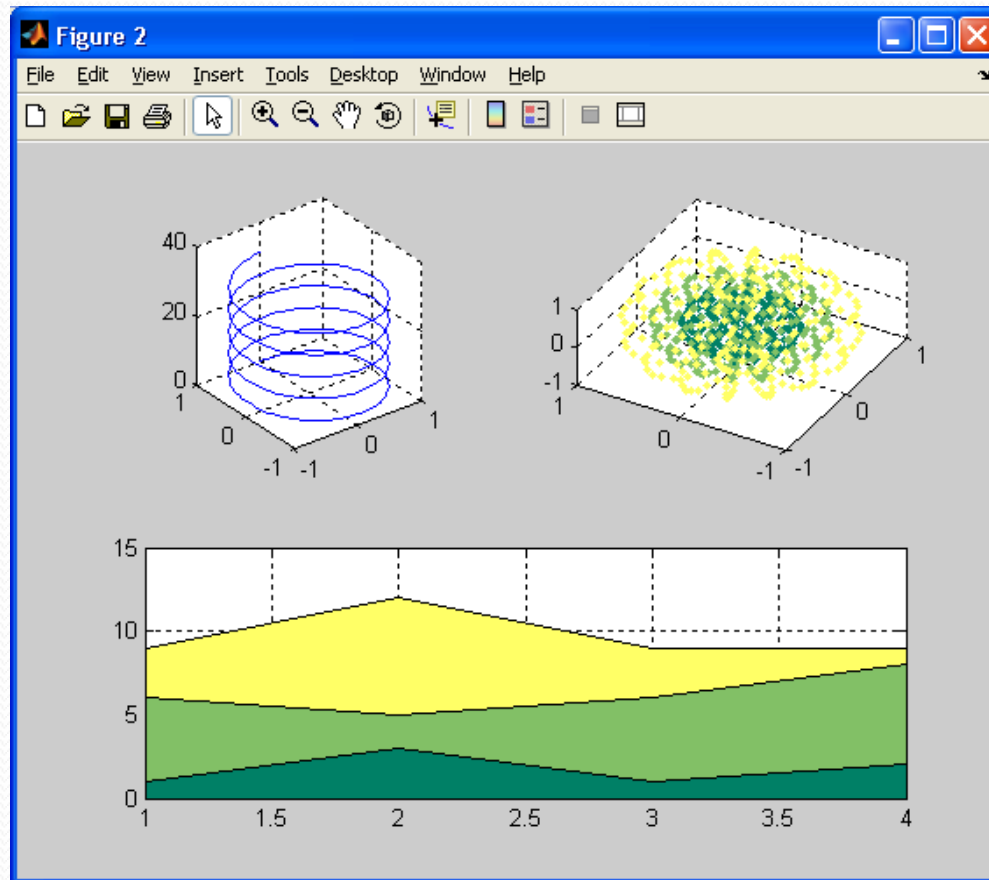
Subplot(2,2,2)

Subplot(2,2,3)

Subplot(2,2,4)

ארגון גרפים

• דוגמא נוספת



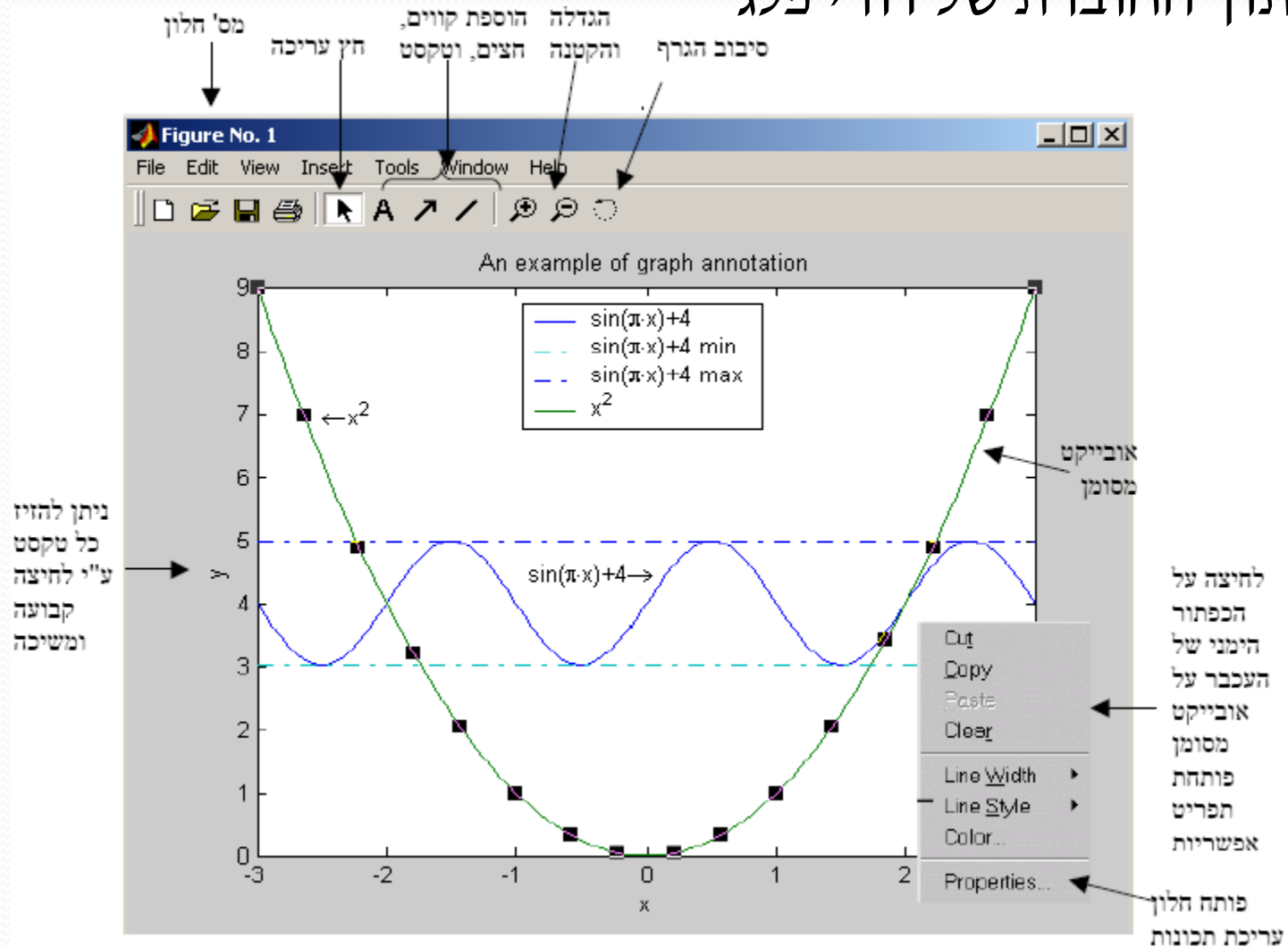
Subplot(2,2,1)

Subplot(2,2,2)

Subplot(2,2,3:4)

עריכה ידנית

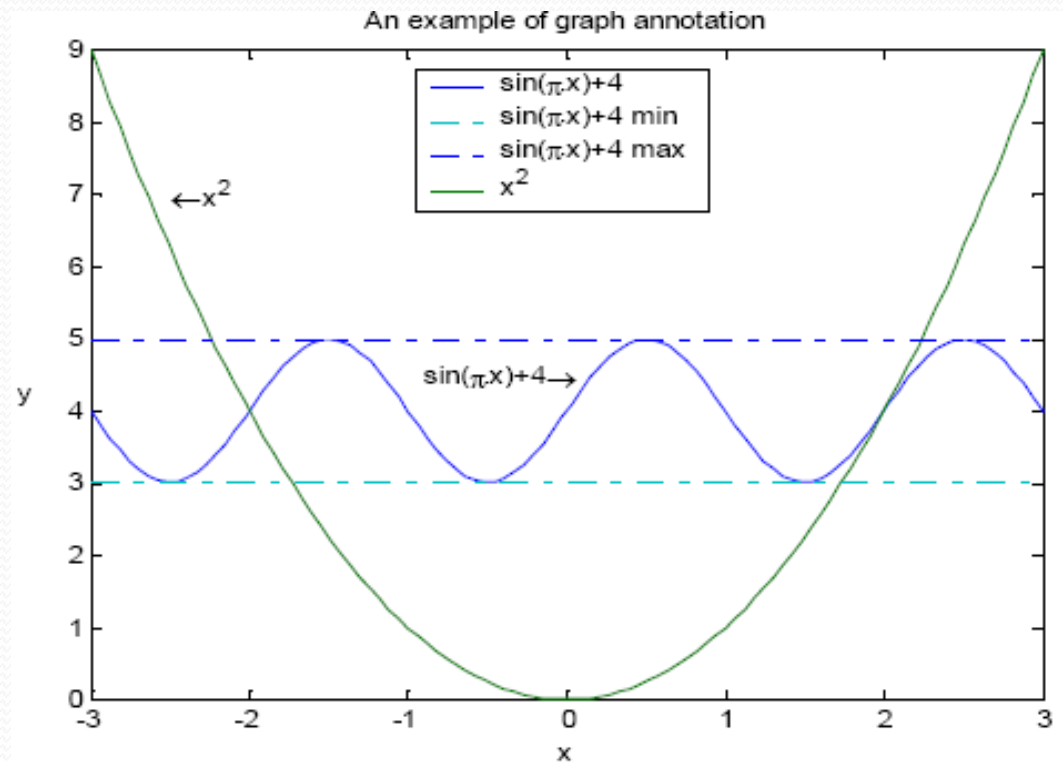
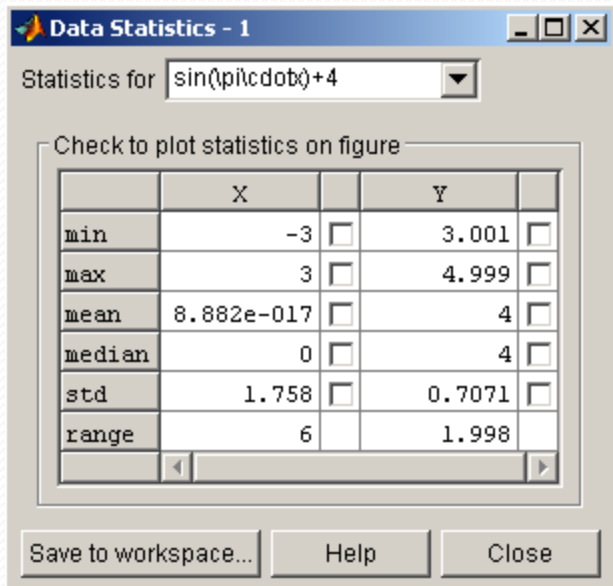
• מתוך החוברת של דורי פלג



עריכה ידנית

- מתוך החוברת של דורי פלג

Tools→Data Statistics:



תרגיל

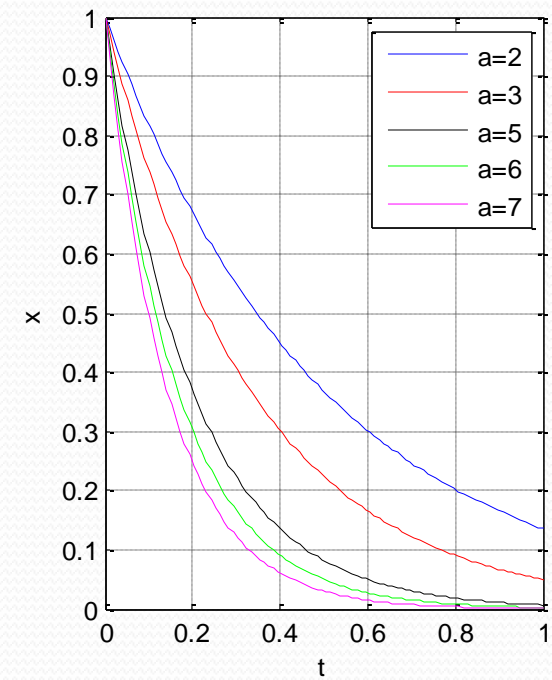
$$x = \exp(-at)$$

- יש לצייר על אותו הגרף אכספוננטים דועכים

כאשר $a=2,3,5,6,7$, $t=[0,1]$

- יש להשלים את השורה בירוקה :

```
t = linspace(0,1);  
a = [2 3 5 6 7];  
ch = 'brkgmcy';  
for k=1:length(a)  
    % calculate and plot x with color ch(k)  
    leg_str{k} = ['a=', num2str(a(k))];  
end;  
legend(leg_str)  
grid on;  
xlabel('t')  
ylabel('x')  
text(0.41,0.46, '\leftarrow slowest')
```



- נסו להוסיף את הפקודה :

תלת מימד

- פקודת meshgrid יוצרת את כל הצירופים האפשריים בין ערכי x וערכי y:

```
>> x = [1 2 3]; y = [4 5];
```

```
>> [x,y] = meshgrid(x,y)
```

```
x =
```

```
1     2     3
1     2     3
```

```
y =
```

```
4     4     4
5     5     5
```

```
-----
x = 0:0.1:2;
```

```
y = 0:0.1:2;
```

```
[xx, yy] = meshgrid(x,y);
```

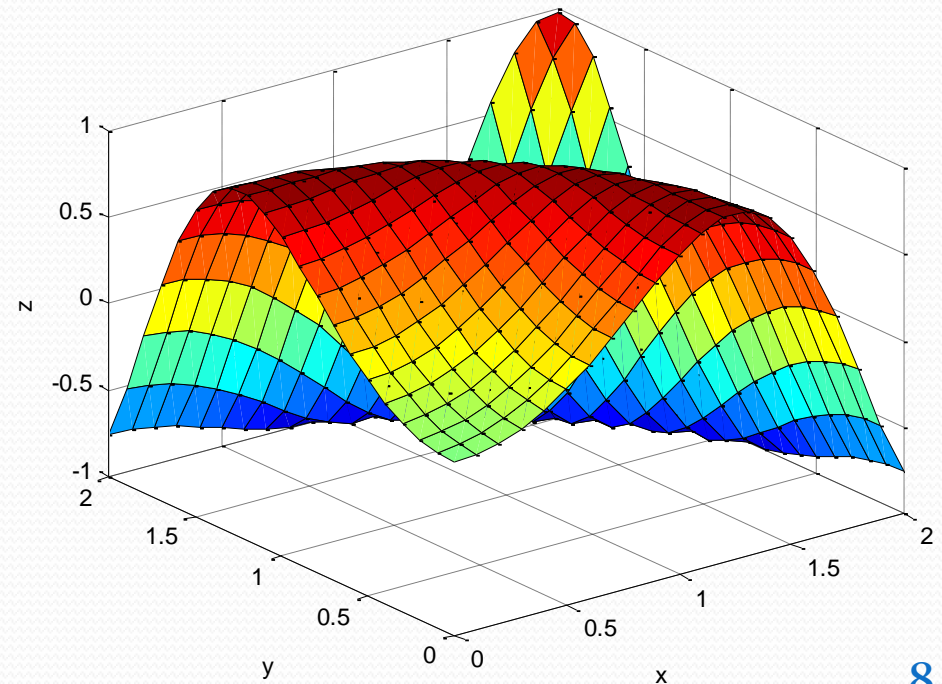
```
zz=sin(xx.^2+yy.^2);
```

```
surf(xx,yy,zz)
```

```
xlabel('x')
```

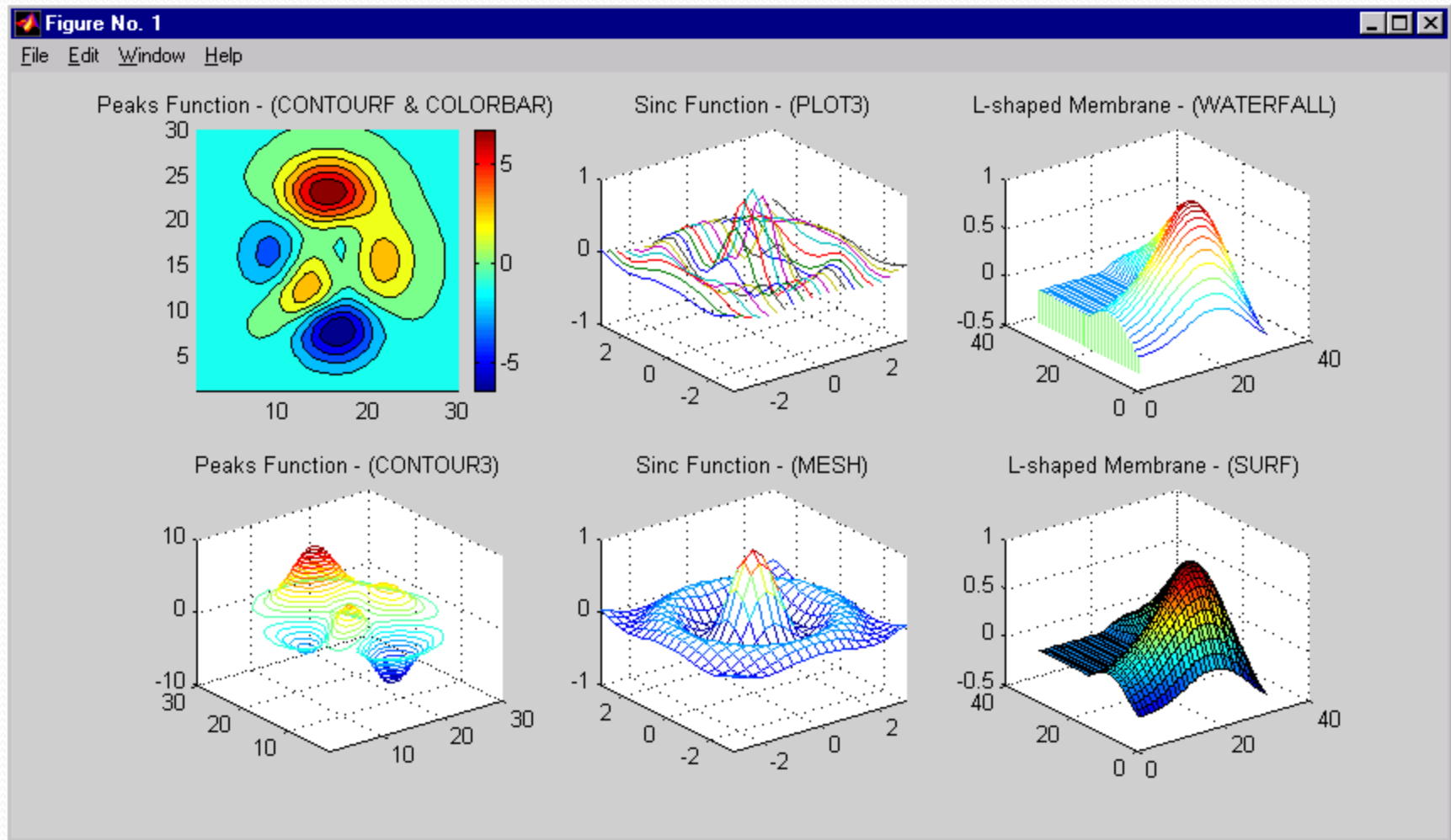
```
ylabel('y')
```

```
zlabel('z')
```



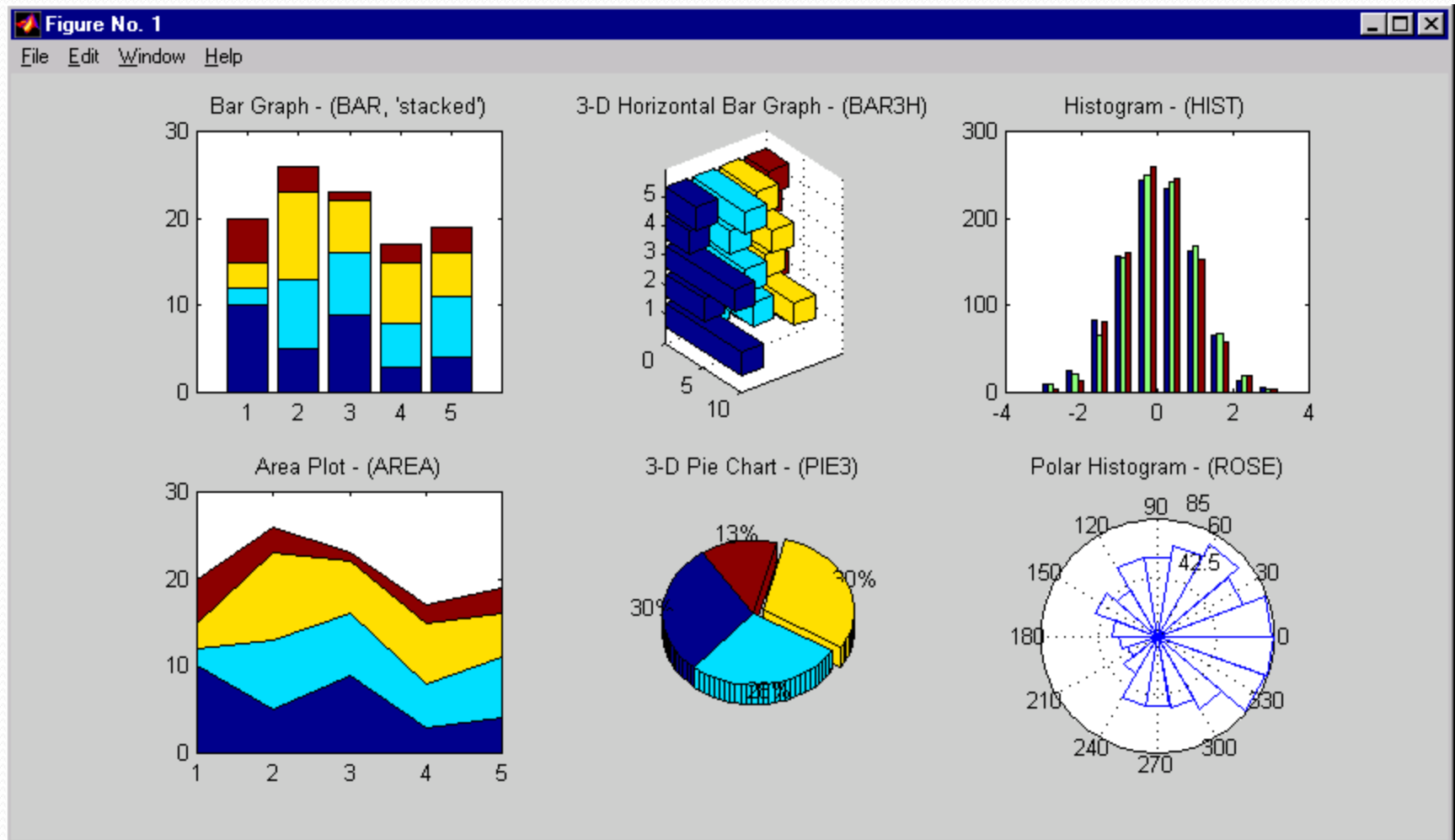
תלת מימד

- פונקציות נוספות: `contour`, `plot3`, `waterfall`, `contour3`, `mesh`, `surf`



גרפים נוספים

• פונקציות נוספות: bar, bar3, hist, area, pie3, rose



בקרת זרימה

- if statements •
- switch statements •
- for loops •
- while loops •
- break statements •

תנאים (If)

צורה כללית

```
>> if expression  
>> ...  
>> elseif expression  
>> ...  
>> else  
>> ...  
>> end
```

```
>> if i == j  
>>     a(i,j) = 2;  
>> elseif i >= j  
>>     a(i,j) = 1;  
>> else  
>>     a(i,j) = 0;  
>> end
```

• דוגמא 1

• דוגמא 2

```
>> if (attn>0.9) & (grade>60)  
>>     pass = 1;  
>> end
```


Switch

- דוגמא

צורה כללית

```
>> switch switch_expr
>> case case_expr1
>> ...
>> case case_expr2
>> ...
>> otherwise
>> ...
>> end
```

```
>> x = 2, y = 3;
>> switch x
>> case x==y
>>     disp('x and y are equal');
>> case x>y
>>     disp('x is greater than y');
>> otherwise
>>     disp('x is less than y');
>> end
x is less than y
```

- בשונה משפת C, לא נדרש שימוש ב-break

לולאות (for)

צורה כללית

```
>> for variable=expression  
>> ...  
>> ...  
>> end
```

• דוגמא 1

```
>> for x = 0:0.05:1  
>>     printf('%d\n',x);  
>> end
```

• דוגמא 2

```
>> a = zeros(n,m);  
>> for i = 1:n  
>>     for j = 1:m  
>>         a(i,j) = 1/(i+j);  
>>     end  
>> end
```

לולאות (while)

צורה כללית

```
>> while expression  
>> ...  
>> ...  
>> end
```

• דוגמא 1

```
>> n = 1;  
>> y = zeros(1,10);  
>> while n <= 10  
>>     y(n) = 2*n/(n+1);  
>>     n = n+1;  
>> end
```

• דוגמא 2

```
>> x = 1;  
>> while x  
>> %execute statements  
>> end
```

• 'true' ו-'false' שווה ערך ל-'true' ו-'false'

שימוש ב-break

- פקודת break עוצרת את הלולאה (for\while)

```
>> y = 3;
>> for x = 1:10
>>     printf('%5d',x);
>>     if (x>y)
>>         break;
>>     end
>> end
1 2 3 4
```

משתנים סימבוליים

- מאפשרים לקבל תוצאות סימבוליות, בדומה לתוכנת מתמטיקה:

```
syms x;  
f = cos(x)  
diff(f)  taylor(f)  int(f)  ezplot(x,f)  laplace(f)  fourier(f)
```

- פונקציות עזר

```
simple(f)  simplify(f)  factor(f)  expand(f)  collect(f)  
pretty(f)          subs(f,x,1)          solve(f)
```

- פנק' solve מחזירה תוצאה סימבולית, על מנת לקבל תוצאה מספרית צריך להשתמש בפנק' eval. למשל:

```
g=solve(x^2+2)  
eval(g)
```

פונקציות מוגדרות אישית

- שם הפונקציה צריך להיות תואם לשם הקובץ

- דוגמא, stat.m :

```
function [mean,stdev] = stat(x)
%STAT Interesting statistics.
n = length(x);
mean = sum(x) / n;
stdev = sqrt(sum((x - mean).^2)/n);
```

פונקציות inline

- לפעמים נח להגדיר פונקציה באופן סיבולי ללא קובץ

```
myfunc=@(x) sin(x)+cos(x)
```

- הביטוי יעבוד בדיוק כאילו היינו מגדירים את הפונקציה בקובץ:

```
function out=myfunc(x)
```

```
out= sin(x)+cos(x)
```

- ניתן לעבוד עם מספר כניסות/יציאות, דוגמא:

```
f=@(x,y,z) [x+y; x*z; z-y];
```

```
f(1,1,1)=[2;1;0]
```

```
f=@(t,x) [x(1)*t; x(2)*t^2; sin(x(3))];
```

```
f(1,[1;1;pi])=[1;1;0]
```

סימולציה של מערכת דינמית

- הסימולציה נעשית ע"י פונקציות מובנות ב- matlab (ode23,ode45...)

- השלב הראשון הוא הבאת המערכת למערכת מסדר ראשון $\dot{\vec{x}} = f(t, \vec{x})$

- לאחר מכן יש לממש את אגף ימין כפונקצית inline (ניתן גם כפונקציה נפרדת)

- לדוגמא עבור :
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \sin t + x_1 x_2 \\ x_1 t + x_2 \end{pmatrix}$$

- נגדיר :

```
odefun = @(t,x) [sin(t)+x(1)*x(2); x(1)*t+x(2)];
```

- ברוב המקרים נשתמש ב ode45 (Runge-Kutta מסדר 4 עם שגיאה מסדר 5) :

```
[tout,xout] = ode45(odefun, [t0 tf] , y0)
```

- $[t0 \quad tf]$ - הוא וקטור הזמנים ו- $x0$ מכיל את תנאי ההתחלה עבור משתני המצב

- $[tout, xout]$ – פלט הסימולציה, לא בהכרח במרווחי זמן קבועים

סימולציה של מערכת דינמית

- מטרת התרגיל היא לבצע סימולציה של תגובת המערכת הלא לינארית הבאה, עם תנאי ההתחלה הבאים:

$$\ddot{x} + (1 - \varepsilon x) \dot{x} + x = 0 \quad \varepsilon = 0.5$$

$$x(0) = 0, \dot{x}(0) = 1$$

- תחילה נכתוב את המערכת כמערכת מסדר ראשון ($\dot{\vec{x}} = f(t, \vec{x})$) ע"י הגדרת $x_1 = x, x_2 = \dot{x}$

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -(1 - \varepsilon x_1) x_2 - x_1 \end{pmatrix}$$

- ממשו את $\dot{\vec{x}}$ כפונקציית inline המקבלת כקלט זמן (t) ו-וקטור בעל 2 איברים (x).

הנחיה: יש להשלים את סימני השאלה בביטוי `xdot = @(t,x) [?; ?]`.

- בהגדרת `x0=[0;1]` מה מתקבל ע"י הרצת `xdot(0,x0)`

- בצעו סימולציה של המערכת בשימוש `ode45` ע"י `[t,x]=ode45(xdot,[0 tf],x0);`

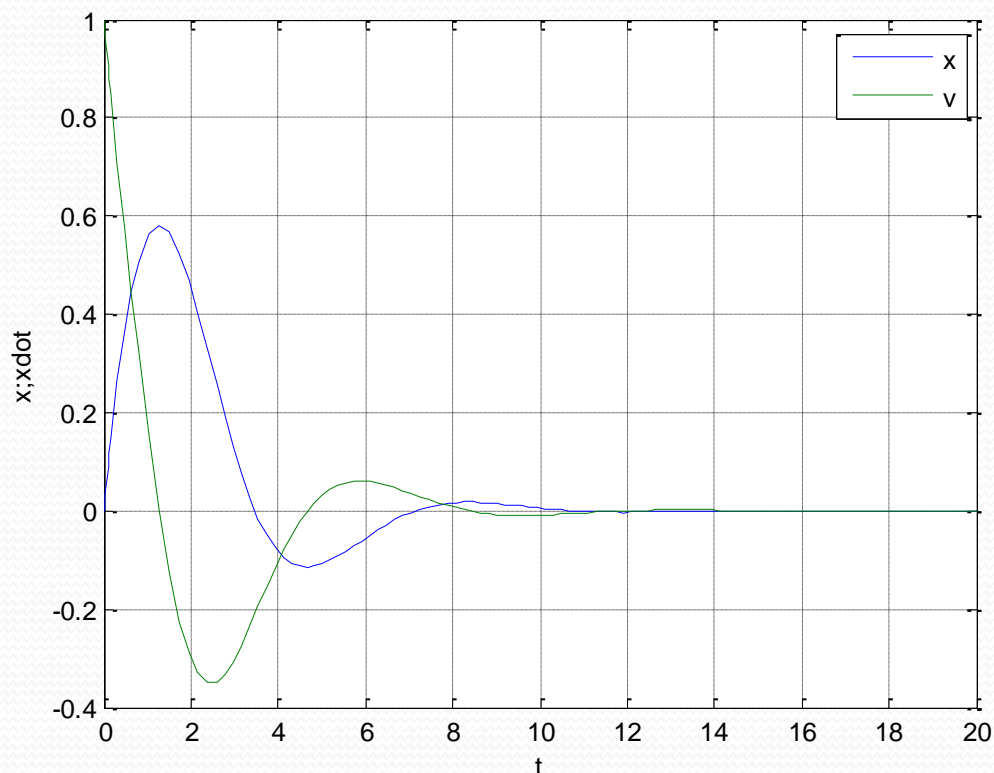
- הקלט הינו: `xdot` - ייצוג המערכת, `x0` - תנאי ההתחלה, `tf` - זמן הסיום (הגדירו אותו כ-20).

- הפלט הינו: `t` - וקטור זמנים (לאו דווקא צעד זמן קבוע), `x` - וקטור בעל 2 עמודות

סימולציה של מערכת דינמית

הערות:


- ניתן לאלץ את השיטה לעבור עם צעד זמן קבוע.
- שיטה זו תעבוד לכל מערכת לא לינארית. אם המערכת היא מסדר N נדרש להגדיר N משתני מצב ו- \dot{x} יכול N איברים.



בקרה – ניתוח תגובת תדר

- ניקח מערכת המתוארת ע"י פונקצית התמסורת הבאה: $G(s) = \frac{1}{s^3 + 2s^2 + 3}$

- נגדיר אותה ב-matlab ע"י הפקודה: $G = tf([1], [1 \ 2 \ 0 \ 3])$

- ניתן לנתח את המערכת במישור התדר ע"י הפונקציות bode, nyquist, nichols.
- מומלץ להשתמש ב-Data Cursor  על מנת לדגום נקודות ספציפיות בגרפים.

- על מנת לנתח את מיקום הקטבים בחוג סגור עבור משוב עם הגבר פרופורציוני נשתמש ב-rlocus. גם כאן מומלץ להיעזר ב-data cursor.

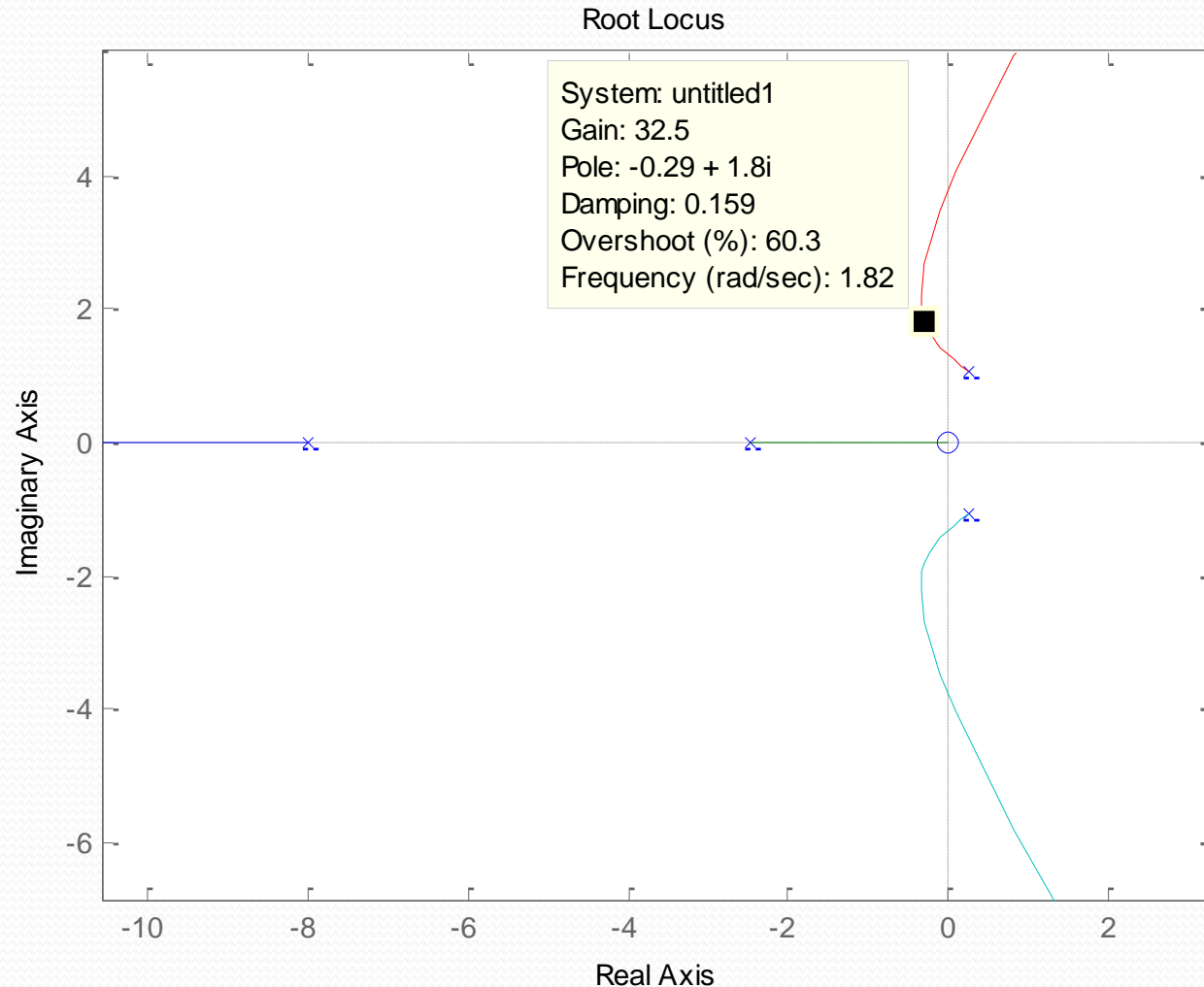
- הדרך הנוחה ביותר לתכנן חוג בקרה למערכת הזו היא ע"י sisotool. הכלי מאפשר להוסיף קטבים ואפסים על פני ה-root locus ולבחון את תגובת המערכת. ל-sisotool ישנו help המכיל הסברים על אופן השימוש.

- נניח את מערכת הדוגמא ע"י washout: $H(s) = \frac{s}{s+8}$

ב-matlab: $H = tf([1 \ 0], [1 \ 8])$

תכנון מערכת בקרה

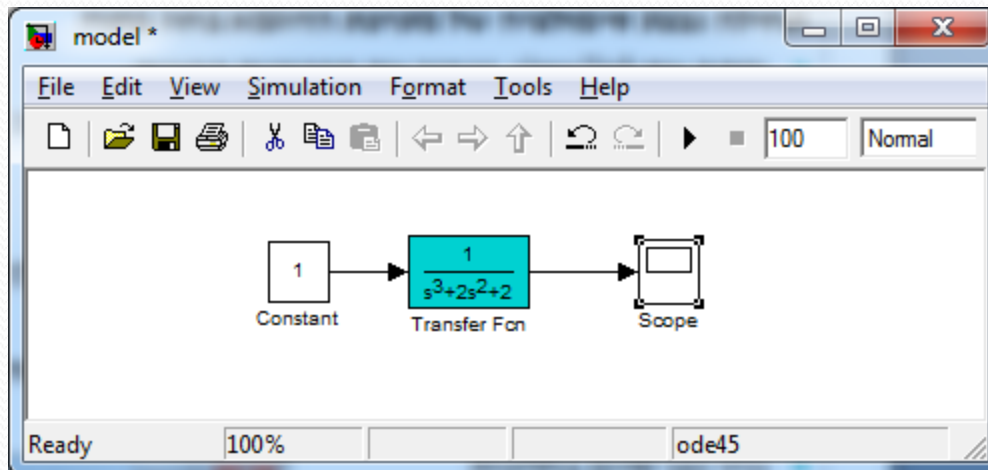
- נבחר בהגבר $K=33$ עבורו הקטבים בחוג סגור יציבים (ב-matlab: `rlocus (G*H)`):



סימולציה ב simulink

תחילה נבצע סימולציה של מערכת הדוגמא בחוג פתוח

- נפתח את simulink ונגרור את הרכיבים הבאים:
 - Transfer Fcn מתוך continuous (בהמשך יסומן בקיצור כ-TF)
 - constant מתוך sources
 - Scope מתוך sinks
- נחבר ביניהם ע"י בחירה ב-constant, החזקת ctrl והקלקה על TF, החזקת ctrl והקלקה על scope
- נגדיר את פנק התמסורת ע"י לחיצה כפולה על TF והכנסת מקדמי הפולינומים במונה ובמכנה
- נזין זמן סיום בחלונית המתאימה בצד ימין למעלה ונלחץ על play



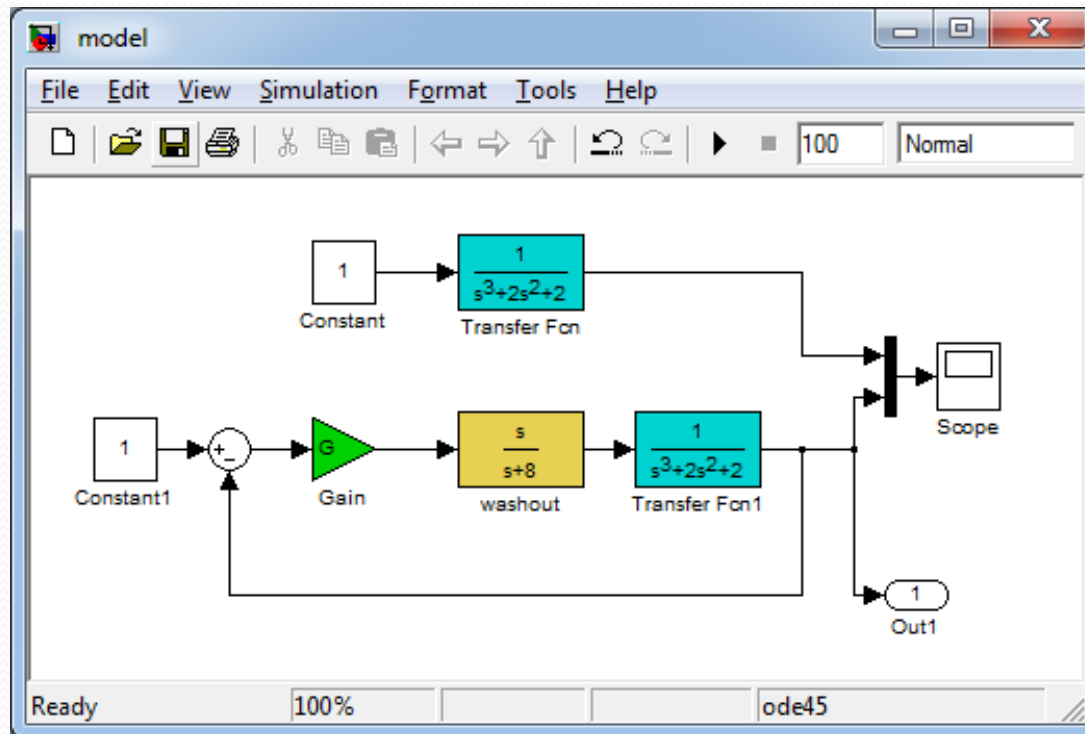
סימולציה ב simulink

נמשיך לסימולציה של המערכת בחוג סגור

- תחילה נשכפל את המערכת בחוג פתוח ע"י סימון כלל הבלוקים וגרירה שלהם עם לחיצה ימנית על העכבר. נוסיף את הבלוקים הבאים:
 - Transfer Fcn מתוך continuous (בשביל ה-washout)
 - Gain ו-sum מתוך math operations
 - Mux מתוך Signal Routing
- נגדיר את ה-washout (בדומה למתואר עבור TF בשקף הקודם)
- נשנה את הסימנים ב-sum ע"י לחיצה כפולה ושינוי של '+' ל '-'
- נשנה את ההגבר ע"י לחיצה כפולה על Gain והזנת 33 (ניתן להזין G בתנאי שהוגדר משתנה $G=33$)
- נחבר ה-mux ל-scope ונחבר בין הבלוקים כמתואר בשקף הבא ליצירת חוג סגור

סימולציה ב simulink

- בעת סימולציה של המערכת נקבל שבחוג פתוח המערכת לא יציבה ואילו בחוג סגור המערכת מתייצבת. ניתן להריץ את המודל גם ע"י קוד matlab וניתן להגדיר את ההגבר G מחוץ למודל. דוגמא לכך ניתן למצוא בקובץ model_sim.m



שונות

- בהקשר של בקרה לפעמים נדרשים לבצע אכספוננט של מטריצה – לצורך כך קיימת הפונקציה `expm`.
- שימו לב שיש הבדל בין `exp(A)` שפועל איבר-איבר לבין `expm(A)`

```
>> A=[1 2; -1 1];
```

```
>> exp(A)
```

```
2.7183    7.3891
```

```
0.3679    2.7183
```

```
>> expm(A)
```

```
0.4239    3.7972
```

```
-1.8986    0.4239
```

```
>> A = [1 0;0 2];
```

```
>> exp(A)
```

```
2.7183    1.0000
```

```
1.0000    7.3891
```

```
>> expm(A)
```

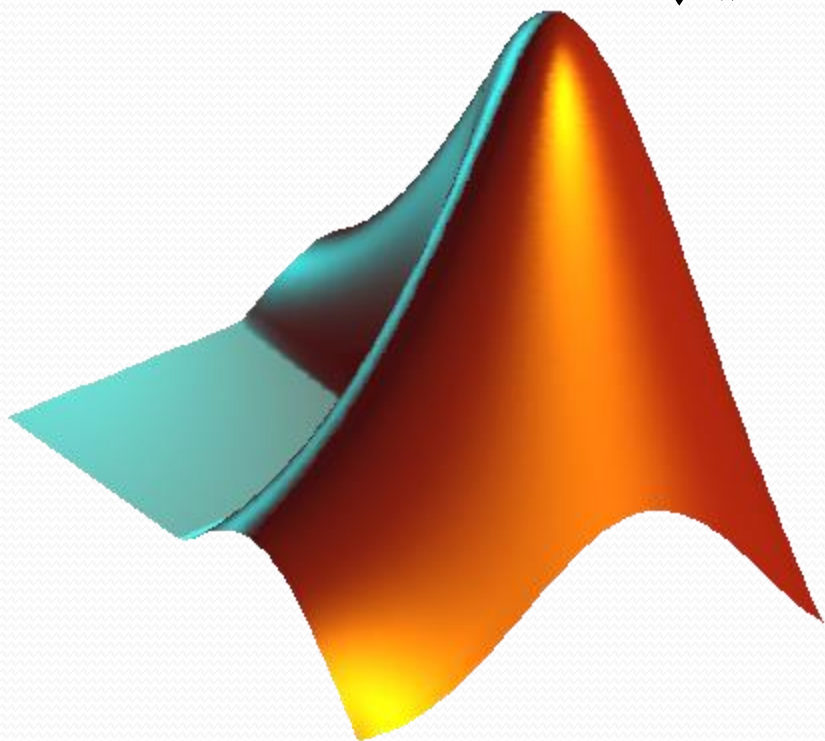
```
2.7183    0
```

```
0    7.3891
```


סיכום

- טעימה קטנה מהיכולות של תוכנת Matlab
- הפונקציה השימושית ביותר : help
- לרוב, החלק הקשה ביותר הוא למצוא את השם של הפונקציה שאנחנו רוצים (סיכוי גבוה שהיא קיימת)

בהצלחה!



שאלות?