יישומי מחשב - Matlab

מבוא

MATLAB הינה תוכנה מתמטית המיועדת לביצוע חישובים טכניים והנדסיים.

במקור נכתבה התוכנה, בסוף שנות השבעים, כדי לפתור בעיות באלגברה ליניארית, ומכאן גם משמעות שמה - MATrix-LABoratory. כיום כוללת התוכנה אופציות רבות נוספות כגון: כלים לטיפול במשוואות דיפרנציאליות, עיבוד נתונים, פולינומים, מחרוזות, ועוד. התוכנה מאפשרת עיבוד נתונים והצגת תוצאות ברמה ויזואלית גבוהה; יצירת גרפים דו-ממדים, תלת ממדים ואנימציות.

אופי התוכנה נומרי, והאפשרות לביצוע חישובים סימבוליים מוגבלת מאוד. ניתן להוסיף לתוכנה TOOL-BOXES המהווים הרחבה של אוצר הפקודות בנושאים שונים כגון בקרה, עיבוד אותות, משוואות דיפרנציאליות חלקיות ועוד.

עם הכניסה לתוכנה נפתח חלון הפקודות של ה- MATLAB. כל שורה שנכתוב בחלון זה תתחיל בסימן << (ה- MATLAB מסמן אוטומטית). כל הפעולות והמשתנים שנגדיר במסך זה ישמרו במרחב העבודה של ה-MATLAB (ה- WORKSPACE). ה- MATLAB עובד בשיטה של "מפרש" (INTERPRETER), לכן לחיצה על מקש RETURN תגרום ל- MATLAB לבצע את הפקודה או רצף הפקודות שנכתבו בשורה האחרונה. כאשר מעונינים לבצע סידרה ארוכה של פקודות (תוכנית) מומלץ לכתוב אותה בעורך (editor) חיצוני. ניתן להשתמש בכל עורך השומר בפורמט ASCII. גרסת MATLAB בה אנו עובדים הינה גרסת PC, הכוללת עורך פנימי.

את התוכניות יש לשמור עם סיומת filename.m . כאשר עובדים במחשב PC התוכניות צריכות להישמר בספריה בה נמצאת התוכנה MATLAB או להגדיר ל- MATLAB מסלול חיפוש אחר. ניתן להשתמש בפקודות cd ו-dir כמו במערכת ההפעלה DOS.

הרצת תוכנית ב-MATLAB מבוצעת ע"י כתיבת שמה בחלון הפקודות (ללא הסיומת m).

פקודות ניהול קבצים

File Management Functions	
cd	Show present working directory or folder
p=cd	Return present working directory in **p**
cd path	Change to directory or folder given by **path**
chdir	Same as **cd**
chdir path	Same as **cd path**
delet test	delete the M-file **test.m**
dir	List all files in the cornet directory or folder
ls	Same as dir
matlabroot	Return directory path to MATLAB executable program
path	Display or modify MATLAB's search path
pwd	same as **cd**
type test	Display the M-file **test.m** in the Command window
what	Return a listing of all M-file **test.m** and MAT-files in the current directory or folder
which test	Display the directory path to **test.m**

cd - שינוי ספרית עבודה.

dir או ls - רשימת קבצים בספריה נוכחית.

path - נטיב חיפוש של קבצי - m.

cd path - שינוי נטיב חיפוש.

delet f-name - מחיקת קובץ.

type f-name - הצגת קובץ על המסך.

פעולות בסיסיות

Operation	Symbol	Example
addition, $a + b$	+	5 + 3
subtraction, $a - b$	-	23 - 12
multiplication, $a \cdot b$	*	3.14 * 0.85
division, $a \div b$	/ or \	56 / 8 = 8 \ 56
power, a^b	^	5 ^ 2

שני סימני חילוק פעילים: חילוק ימני / וחילוק שמאלי \.

החישוב יבוצע משמאל לימין בסדר הבא: חזקות, כפל או חילוק, חיבור או חיסור. ניתן לשנות את הסדר ע"י שימוש בסוגריים. הסוגריים הפנימיים ביותר יחושבו ראשונים.

ה- MATLAB אינו מתחשב ברווחים לדוגמא:

```
» 2*4+2  *  5
ans =
   18
```

אם לא הוגדר משתנה עבור התוצאה ה- MATLAB ישתמש במשתנה ברירת המחדל ans.

אפשר לבצע חישוב דומה ע״י הגדרת משתנים:

```
» a=4,b=5;
a =
    4
» c=2*a+2*b
c =
   18
```

אחרי ביצוע השורה הראשונה נכתב ערכו של a על המסך, לעומת זאת ערכו של b לא נכתב. שימוש בסיומת **;** מונע הדפסה של התוצאה על המסך (בכל מקרה, גם ללא הדפסה התוצאה נשמרת בזיכרון).

נשנה את ערכו של a ונבדוק האם c ישתנה בהתאם.

```
» a=10;
» c
c =
   18
```

כדי לעדכן את ערכו של c צריך לבצע מחדש את פעולת החישוב.

```
» c=2*a+2*b
c =
   30
```

אפשר לשחזר שורה שכבר נכתבה בחלון הפקודות ע״י שימוש בחצים ↑ ↓. אם נקיש מספר אותיות ראשונות של פקודה שכבר נכתבה ו- ↑ נקבל מיידית את הפקודה.

פורמט הצגה של מספרים

מספרים שלמים יכתבו ללא נק` עשרונית. ברירת המחדל עבור מספרים ממשים היא 4 ספרות אחרי הנק` העשרונית.

הפקודות הבאות משמשות לקביעת צורת ההצגה.

MATLAB Command	Comment	Example: 215 / 6
format short	default display	35.8333
format long	16 digits	35.83333333333334
format short e	5 digits + exponent	35.833e+01
format long e	16 digits + exponent	35.83333333333334e+01
format hex	hexadecimal	4041eaaaaaaaaaab
format bank	2 decimal digits	35.83
format +	positive negative or zero	+
format rat	rational approximation	215 / 6

שימוש בפורמט מסוים אינו משנה את ערך המשתנים אלא רק את אופן הצגתם.

משתנים

שם של משתנה יורכב ממלה אחת עד 19 תווים, כאשר התו הראשון חייב להיות אות. ה- MATLAB מבחין בין אותיות גדולות וקטנות. פקודות חייבות להיכתב באותיות קטנות.

משמעות מיוחדת ניתנת למשתנים הבאים :

Special variables	Value
ans	The default variable name used for results
pi	The ratio of circumference of a circle to its diameter
eps	The smallest number such that when added to one creates a number grater than one on the computer
flops	Count of floating point operations
inf	Which stands for infinity, e.g., 1/0
NaN	Which stands for Not-a-Number, e.g., 0/0
I (end) **j**	$\mathbf{I} = \mathbf{j} = \sqrt{-1}$
relmin	The smallest usable positive real number
relmax	The largest usable positive real number
nargin	Number of function input arguments used
nargout	Number of function output arguments used

inf = אינסוף , חלוקה ב0- תגרום לתוצאה inf ולא לעצירת המחשב.

eps = המספר הקטן ביותר המוכר למחשב.

NaN = לא קיים מספר, "ריק".

אינפורמציה על המשתנים המוגדרים ב- WORKSPACE נקבל ע״י שימוש בפקודות:

```
» who
Your variables are:
a      b      c
```

או יותר מפורט:

```
» whos
        Name      Size      Elements      Bytes   Density   Complex
         a        1 by 1       1           8       Full       No
         b        1 by 1       1           8       Full       No
         c        1 by 1       1           8       Full       No
Grand total is 3 elements using 24 bytes
```

הפקודות הבאות משמשות למחיקת משתנים מ ה- WORKSPACE:

```
» clear all
    או
» clear a b . . .
```

ניקוד פיסוק והערות

הסימנים ״;״ ו- ״,״ משמשים להפרדה בין פקודות. שימוש ב- ; מונע את הצגת התוצאה.

הסימן ״%״ משמש לכתיבת הערות, ה- MATLAB יתעלם מכל מה שייכתב מימין ל- %.

הסימן ״. . .״ אומר ל- MATLAB שהפקודה ממשיכה בשורה הבאה:

```
» 4*...
5
ans =
   20
```

לא ניתן לקטוע שם של משתנה על ידי רווח (במקרה ורוצים להפריד שם של משתנה לשתי שמות מומלץ לעשות זאת על ידי קו תחתון לדוגמא first_name).

מטריצות

מטריצה הינה יחידת ארגון הנתונים הבסיסית ב-MATLAB. כל נתון מספרי – ממשי או מורכב, מיוצג למעשה על-ידי מטריצה. מטריצה מיוצגת על-ידי מערך דו-ממדי. וקטור הנו מקרה פרטי של מטריצה חד-ממדית.
שימוש במטריצות מאפשר ביצוע פעולות מתמטיות על קבוצה גדולה של איברים, מבלי לחזור על הפעולה עבור כל איבר.
הגדרת מטריצה ב-MATLAB תעשה ע"י שימוש בסוגריים מרובעים [. . .]. מעבר בין שורות יסומן ב-"; ". לצורך הפרדה בין איברים בשורה, ניתן להשתמש ברווח או ב-", ".

☜ לדוגמה

רוצים לחשב את הביטוי y = sin(x) עבור הערכים: $0 \le X \le \pi$. מכיוון שלא ניתן לחשב עבור כל נקודה בתחום, דוגמים את הפונקציה במרווחים קבועים של π0.1:

```
» x=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1]*pi
x =
  Columns 1 through 7
  0   0.3142   0.6283   0.9425   1.2566   1.5708   1.8850
  Columns 8 through 11
  2.1991  2.5133  2.8274  3.1416

» y=sin(x)
y =
  Columns 1 through 7
  0   0.3090   0.5878   0.8090   0.9511   1.0000   0.9511
  Columns 8 through 11
  0.8090   0.5878   0.3090   0.0000
```

הפקודות הבאות משמשות ליצירת מערכים חד-ממדיים ליניאריים או לוגריתמיים:

יצירת וקטור שורה המכיל איברים שונים	x = [2 2* pi sqrt(2) 2-3j]
יצירת וקטור שורה, אשר איבריו מתחילים בערך first, מתקדמים בפסיעות של 1, ומסתיימים בערך last או בערך הקרוב אליו ביותר מלמטה.	x = first : last
יצירת וקטור שורה, אשר איבריו מתחילים בערך first, מתקדמים בפסיעות של increment, ומסתיימים בערך last או בערך הקרוב אליו ביותר מלמטה.	x = first : increment : last
יצירת וקטור שורה בעל n איברים, אשר איבריו מתחילים בערך first, ומסתיימים בערך last.	x = linspace (first ,last ,n)
יצירת וקטור שורה בעל n איברים, אשר איבריו מתחילים בערך 10^{first} ומסתיימים בערך 10^{last}, בעל ריווח לוגריתמי.	x = logspace (first ,last ,n)

☜ **לדוגמה**

את הוקטור **x** שהופיע בדוגמא הקודמת אפשר לבנות גם כך:

```
» x=[0:0.1:1]*pi;
```

או

```
» x=[linspace(0,1,11)]*pi;
```

פניה לאיבר מסוים בתוך מטריצה תעשה ע"י ציון שם המטריצה, כפי שהגדרנו אותו בזמן הגדרת המטריצה, והאינדקס של אותו איבר. המספר הראשון מציין את מספר השורה והאיבר השני – את מספר העמודה. פנייה למספר איברים רציפים, בטור או בשורה, תתבצע על-ידי הסימן "**:**". אם נכתוב את הסימן "**:**" ללא מספרים מצידיו, הפנייה מתייחסת לכל איברי השורה / עמודה.

☜ **לדוגמה**

אם נשתמש בוקטור, שהגדרנו קודם לכן:

```
» x(3)
ans =
    0.6283
» x(1:5)
ans =
    0    0.3142    0.6283    0.9425    1.2566
» x(3:-1:1)
ans =
    0.6283    0.3142         0
```

ניתן להרכיב מטריצה גדולה מתת-מטריצות קיימות.

☜ **לדוגמה**

```
» a=1:5,b=1:2:9
a =
     1     2     3     4     5
b =
     1     3     5     7     9

» c=[a b]
c =
     1     2     3     4     5     1     3     5     7
9
```

כאשר מגדירים מערך דו- ממדי יופרדו השורות , כפי שכבר צוין, ע"י הסימן "**;**":

```
» A=[1 2 3;4 5 6]

A =
     1     2     3
     4     5     6
```

הגדרת ווקטור עמודה:

```
» a=[1; 2 ;3 ];
a =
     1
     2
     3
```

או ע"י שימוש בסימן '''' (שמשמעותו tranpose):

```
» a=[1 2 3 4].'
a =
     1
     2
     3
     4
```

אם נשתמש בסימן transpose ללא נקודה לפני הגרש, מספרים מרוכבים יהפכו לצמוד.

☜ **לדוגמה**

```
» c=a+a*i
c =
   1.0000 + 1.0000i
   2.0000 + 2.0000i
   3.0000 + 3.0000i
   4.0000 + 4.0000i
» c'
ans =
     1.0000 -1.0000i   2.0000 -2.0000i   3.0000 -3.0000i
     4.0000 -4.0000i
```

באופן כללי הסימן ".." ב-MATLAB מאפיין פעולות בין מערכים, כאשר לא מתייחסים אל המערכים כאל מטריצות (בדרך ההתייחסות המוכרת לנו מאלגברה ליניארית), כלומר הפעולה החשבונית מבוצעת על זוגות איברים תואמים.

☜ **לדוגמה**

```
» a=[1 2];b=[3 4];
» c=a.*b
c =
     3     8
```

שימוש בסימן כפל ללא נקודה מייצג הכפלת מטריצות כמו באלגברה ליניארית ומחייב ממדי מטריצות תואמים:

```
» c=a*b'
c =
     11
```

פעולות בין סקלר למטריצה

כאשר מבצעים פעולה חשבונית בין סקלר ומטריצה תבוצע פעולה זהה על כל אחד מאברי המטריצה:

```
» a=[1 2;3 4]
a =
     1     2
     3     4
» a+2
ans =
     3     4
     5     6
» 2*a-1
ans =
     1     3
     5     7
```

פעולות בין מטריצות

פעולה בין מטריצות אפשרית אך ורק כאשר למטריצות ממדים זהים. בין מטריצות אפשר לבצע: חיבור חיסור כפל חילוק וחזקה. כבר ראינו שאם נסמן נקודה אחרי שם המטריצה , הפעולה תבוצע בין איברים בעלי אינדקס זהה.

☜ **לדוגמה**

```
» a=[1 2;3 4];
» b=[1 1;2 2];
» a+b
ans =
     2     3
     5     6
» a.*b
ans =
     1     2
     6     8
» a./b   %  ( = b.\a )
ans =
    1.0000    2.0000
    1.5000    2.0000
» a.^b
ans =
     1     2
     9    16
» a.^(-1)
ans =
    1.0000    0.5000
    0.3333    0.2500
```

הטבלה הבאה מסכמת פעולות בין מטריצות:

	a = [a1 a2 ... an] , b = [b1 b2 ... bn] c = <a scalar>
חיבור סקלר ומטריצה	a + c = [a1+c a2+c ... an+c]
הכפלת סקלר במטריצה	a∗c = [a1∗c a2∗c ... an∗c]
חיבור מטריצות	a + b = [a1+b1 a2+b2 ... an+bn]
הכפלת איברי מטריצות	a .∗ b = [a1∗b1 a2∗b2 ... an∗bn]
חילוק ימני	a . / b = [a1/b1 a2/b2 ... an/bn]
חילוק שמאלי	a . \ b = [a1\b1 a2\b2 ... an\bn]
פעולות בחזקה	a . ^c = [a1^c a2^c ... an^c] c . ^a = [c^a1 c^a2 ... c^an] a . ^b = [a1^b1 a2^b2 ... an^bn]

פעולות חכמות

החלפת איבר בתוך מערך:

```
» a=[1 2;3 4]
a =
     1     2
     3     4
» a(2,2)=5
a =
     1     2
     3     5
» a(3,3)=6
a =
     1     2     0
     3     5     0
     0     0     6
```

MATLAB לא דורש הצהרות לגבי גודל או סוג המערך לכן במקרה זה, כשדרשנו למקם איבר בעל אינדקס (3,3) במערך 2*2 המערך הוגדל ונוספו בו אפסים.

מטריצות ותת-מטריצות

הגדרת מטריצה מתוך מטריצה אחרת:

```
» a=[1 2 3;4 5 6;7 8 9]
a =
     1     2     3
     4     5     6
     7     8     9
» b=a(1:2,2:3)
b =
     2     3
     5     6
```

הרכבת מטריצה על-ידי שימוש במטריצות אחרות:

```
» a=[1 2;4 5]
a =
     1     2
     4     5
» b=[3 6]'
b =
     3
     6
» c=[a b;7 8 9]
c =
     1     2     3
     4     5     6
     7     8     9
```

מחיקת שורה או עמודה - שימוש במטריצה ריקה:

```
» c(:,2)=[]
c =
     1     3
     4     6
     7     9
```

שימוש בסימן **(:)** במקום אינדקס מיצג את כל האינדקסים הקיימים:

```
» c(:)'
ans =
     1 4 7 3 6 9
```

שכפול וקטור ליצירת מטריצה:

```
» c
c =
     1     3
     4     6
     7     9
» d=c(:,[2 2 2 2])
d =
     3     3     3     3
     6     6     6     6
     9     9     9     9
```

האיבר הראשון בסוגריים העגולים - ":", מתייחס לכל האיברים בעלי אינדקס עמודה שווה, בכל השורות – למעשה וקטור עמודה. האיבר השני מגדיר את אינדקס העמודה. בדוגמה הנ"ל, האיבר השני הוא וקטור. לפיכך יתבצע תהליך בניית וקטור עמודה עבור כל איבר בוקטור.

אפשר לפנות אל איבר השייך למערך דו-ממדי ע"י שימוש באינדקס אחד. ה-MATLAB סופר את איברי המטריצה, לפי סדר העמודות, ממעלה למטה.

```
» a=[1 2 3 4;5 6 7 8]
a =
```

```
     1     2     3     4
     5     6     7     8
» a(5)
ans =
     3
```

מערכים המכילים את האיברים אפס ואחד נקראים מערכים לוגיים. אפס מייצג שקר (false) ואחד מיצג אמת (true).

☜ **לדוגמה**

```
» x=-3:3
x =
    -3    -2    -1     0     1     2     3
» abs(x)>1
ans =
     1     1     0     0     0     1     1
» y=x(abs(x)>1)
y =
    -3    -2     2     3
```

כאשר הפניה לאיברים במטריצה מיוצגת על-ידי תנאי בוליאני, ההתייחסות היא לאיברים המקיימים את התנאי.

הטבלה הבאה מסכמת את השימוש באינדקסים

A(r,c)	פנייה לאיבר השייך למטריצה A הנמצא בשורה r ובעמודה c.
A(r,:)	פנייה לכל האיברים השייכים למטריצה A אשר נמצאים בכל העמודות שבשורה r.
A(:,c)	פנייה לכל האיברים השייכים למטריצה A אשר נמצאים בכל השורות שבעמודה c.
A(:)	פונה לכל האיברים השייכים למטריצה A כאשר האיברים מסודרים כווקטור עמודה. האיברים נלקחים לווקטור זה עמודה עמודה לפי הסדר.
A(i)	פנייה לאיבר i של מטריצה A המסודרת כווקטור עמודה כפי שמוסבר לעיל.

<u>פקודות חיפוש</u>

i=find(x)	Return indices of the **x** where its elements are nonzero
[r,c]=find(x)	Return row and column indices of the array **x** where its elements are nonzero

☜ **לדוגמה** כמו בדוגמא הקודמת נבנה מטריצה המורכבת מכל המספרים שערכם המוחלט גדול מ- 1 בתחום : (-3) עד 3. נשתמש בפקודת חיפוש :

```
» x=-3:3
x =
    -3    -2    -1     0     1     2     3
» k=find(abs(x)>1)
k =
     1     2     6     7
```

```
» y=x(k)
y =
    -3    -2     2     3
```

פקודות מטריצה נוספות

flipud(A) - סיבוב מטריצה למעלה או למטה.
fliplr(A) - סיבוב מטריצה ימינה או שמאלה.
reshape(A,m,n) - שינוי הממדים של המטריצה A (A חייבת להכיל m*n אברים).
rot90(A) - סיבוב מטריצה בכפולות של 90 מעלות ימינה או שמאלה.

☜ **לדוגמה**

```
» a=[1 2 3;4 5 6]
a =
     1     2     3
     4     5     6
» b=rot90(a)
b =
     3     6
     2     5
     1     4
```

diag(A) - בונה ווקטור עמודה מהאלכסון הראשי של המטריצה **A**.
diag(v) - בונה מטריצה אלכסונית, הווקטור **V** ירכיב את האלכסון הראשי.

☜ **לדוגמה**

נעשה שימוש בפקודה diag כדי לבנות מטריצת יחידה:

```
» v=[1 1];
» I=diag(v)
I =
     1     0
     0     1
```

מערכים רב-ממדיים

גרסאות קודמות של - MATLAB אפשרו שימוש במערכים חד ממדיים ודו ממדיים בלבד. הגרסה הנוכחית, MATLAB - 5, מאפשרת שימוש במערכים רב ממדיים (מערכים בעלי מס' ממדים גדול מ - 2).

שימוש במערכים רב-ממדים בפקודות אלגברה ליניארית המיועדות לעבוד עם מטריצות אסור, במקרה זה אפשר לפרק את המערך הרב-ממדי לתת-מערכים דו-ממדים.

אם תופעל פעולה בין סקלר ומערך רב-ממדי תבוצע פעולה זהה בין הסקלר ובין כל אחד מאברי המערך.

יצירת מערך רב-ממדי אפשרית בשלוש דרכים :

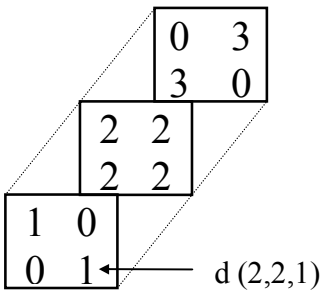
- ע"י שימוש באינדקסים.
- שימוש בפקודות MATLAB (מערכים מיוחדים).
- שימוש בפקודה cat (הרכבת מערך רב-ממדי)

דוגמא - יצירת מערך תלת ממדי בעל שלוש שכבות משלושה מערכים דו ממדים.

```
» a=[1 0;0 1];
» b=[2 2;2 2];
» c=[0 3;3 0];
```

שלושה ממדים

```
» d=cat(3,a,b,c)
d(:,:,1) =
    1    0
    0    1
d(:,:,2) =
    2    2
    2    2
d(:,:,3) =
    0    3
    3    0
» size(d)
ans =
    2    2    3
```



שימוש באינדקסים:

```
» A=[1 0 4;3 5 6;9 8 7];
» A(:,:,2)=[1 0 4;3 5 6;9 8 7]
A(:,:,1) =
    1    0    4
    3    5    6
    9    8    7
A(:,:,2) =
    1    0    4
    3    5    6
    9    8    7
```

פקודות הפועלות על ווקטורים

```
» max(A)
ans(:,:,1) =
    9    8    7
ans(:,:,2) =
    9    8    7
```

פקודות היוצרות מערכים מיוחדים:

```
» 5*ones(2,2,2)
ans(:,:,1) =
    5    5
    5    5
ans(:,:,2) =
    5    5
    5    5
```

קבלת אינפורמציה על מערכים רב-ממדיים:

```
» size(A)
ans =
    3    3    2
» ndims(A)
ans =
    3
» whos
  Name      Size          Bytes  Class
  A         3x3x2           144  double array
```

ממד

עזרה – help

הדרך הנוחה והמפורטת ביותר לקבלת עזרה היא על ידי התחברות לרשת דרך התפריט:

Help Desk (HTML)

כמו כן ניתן לקבל הסבר על כל פקודה ופקודה ע״י הקשת help ושם הפקודה, לדוגמא:

```
» help sqrt

 SQRT Square root.
        SQRT(X) is the square root of the elements of X. Complex
        results are produced if X is not positive.

        See also SQRTM.
```

הפקודה **help** כשלעצמה תגרום להצגת רשימת נושאים - help topics.

help topic - הצגת הפקודות הקשורות באותו נושא.

lookfor keyword - רשימת פקודות לפי הקשר.

```
» lookfor complex
CONJ  Complex conjugate.
IMAG  Complex imaginary part.
REAL  Complex real part.
. . . . .
. . . . .
```

מספרים מרוכבים

ב- MATLAB אין צורך לקבוע את סוג המשתנה, באופן כללי משתנים ב- MATLAB הם מטריצות המכילות איברים מרוכבים. מקרה פרטי יכול להיות סקלר ממשי.

הערך המדומה של מספר מרוכב יסתיים באותיות **i** או **j** .

```
» c1=1+2i
c1 =
  1.0000 + 2.0000i
» c2=1+sin(pi/2)*i
c2 =
  1.0000 + 1.0000i
```

הצגת מספרים מרוכבים אפשרית בצורה פולארית או קרטזית:

$$M\angle\theta = M \cdot e^{j\theta} = a + bi$$

$$M = \sqrt{a^2 + b^2} \qquad \theta = \tan^{-1}(b/a)$$

$$a = M\cos\theta \qquad b = M\sin\theta$$

ב- MATLAB יבוצע המעבר ע"י שימוש בפקודות הבאות:

```
» M=abs(c1)
M =
   2.2361
» teta=angle(c1)
teta =
   1.1071
» a=real(c1)
a =
    1
» b=imag(c1)
b =
    2
```

זוויות ב- MATLAB מיוצגות ברדיאנים.

הפקודה abs משמשת גם לחישוב ערך מוחלט.

פקודות נוספות

conj(z)	הצמוד של מספר מרוכב.
[teta,r]=cart2pol(x,y)	מציג בצורה פולרית מספר מרוכב המתואר בצורה קרטזית.
[x,y]=pol2cart(theta,r)	מציג בצורה קרטזית מספר מרוכב המתואר בצורה פולרית.

פונקציות מתמטיות

Mathematical Functions	
abs(x)	Absolute value or magnitude of complex number
acos(x)	Inverse cosine
acosh(x)	Inverse hyperbolic cosine
angle(x)	Four quadrant angle of complex
asin(x)	Inverse sine
asinh(x)	Inverse hyperbolic sine
atan(x)	Inverse tangent
atan2(x,y)	Four quadrant inverse tangent
atanh(x)	Inverse hyperbolic tangent
ceil(x)	Round toward plus infinity
conj(x)	Complex conjugate
cos(x)	Cosine
cosh(x)	Hyperbolic cosine
exp(x)	Exponential: e^x
fix(x)	round toward zero
floor(x)	round toward minus infinity
gcd(x,y)	Greatest common divisor of integer **x** end **y**
imag(x)	Complex imaginary part
lcm(x,y)	Least common multiple of integer **x** end **y**
log(x)	Natural logarithm
log10(x)	Common logarithm
real(x)	Complex real part
rem(x,y)	Remainder after division **rem(x,y)** gives the remainder of **x / y**
round(x)	Round toward nearest integer
sign(x)	Signum function: return sign of argument e.g., **sign(1.2)=1, sign(-23.4)=-1, sign(0)=0**
sin(x)	Sine
sinh(x)	Hyperbolic sine
sqrt(x)	Square root
tan(x)	Tangent
tanh(x)	Hyperbolic tangent

דוגמאות:

```
» y=sqrt(3^2+4^2)
y =
    5
» y=rem(23,4)
y =
    3
```

23/4=5 (3)

rem(x,y) - שארית אחרי חילוק **x / y** .

```
» x=-2.6, y1=round(x), y2=fix(x)
x =
   -2.6000
y1 =
    -3
y2 =
    -2
```

round(x) - מעגל כלפי המספר השלם הקרוב ביותר.

fix(x) - מעגל כלפי אפס.

פעולות על מטריצות ואלגברה ליניארית

במקור נכתבה MATLAB כדי לפשט חישובי מטריצות ואלגברה ליניארית. אחת מהבעיות הנפוצות באלגברה ליניארית היא פתרון של מערכת משוואות ליניארית. נראה כיצד פותרים בעיה זו ב- MATLAB.

נתונה מערכת משוואות:

$$\begin{aligned} x_1 + 2\cdot x_2 + 3\cdot x_3 &= 366 \\ 4\cdot x_1 + 5\cdot x_2 + 6\cdot x_3 &= 804 \\ 7\cdot x_1 + 8\cdot x_2 &= 351 \end{aligned}$$

ובצורתה המטריציונית:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 366 \\ 804 \\ 351 \end{bmatrix}$$

$$A * x = b$$

פעולת הכפל במקרה זה מייצגת מכפלת מטריצות, כפי שמכירים מאלגברה ליניארית (בניגוד למכפלת מערכים המסומנת ב- MATLAB בסימן "*.").

הפתרון למערכת המשוואות נתון ע"י הביטוי $x = A^{-1} * b$. קיים פתרון יחיד אם המטריצה A היא מטריצה הפיכה.

נבדוק בעזרת ה- MATLAB האם המטריצה A הפיכה (מטריצה לא סינגולרית).

det(A) - דטרמיננטה של מטריצה A

מטריצה ריבועית בעלת דטרמיננטה שונה מאפס היא מטריצה הפיכה.

```
» A=[1 2 3;4 5 6;7 8 0]
A =
     1     2     3
     4     5     6
     7     8     0
» det(A)
ans =
27
```

rank(A) – דרגת המטריצה A

מטריצה ריבועית בעלת דרגה מלאה היא מטריצה הפיכה.

```
» rank(A)
ans =
3
```

את הפתרון למערכת המשוואות אפשר לחשב ב- MATLAB בשתי דרכים:

1. inv(A) – המטריצה ההופכית למטריצה A

```
» A=[1 2 3;4 5 6;7 8 0];
» b=[366;804;351];

» x=inv(A)*b
x =
   25.0000
   22.0000
   99.0000
```

2. הדרך השניה כוללת חלוקת מטריצות, פעולה כזו אינה קיימת באלגברה ליניארית ומשמעותה ב- MATLAB היא כפל בהופכי.

יש לשים לב להבדל הקיים בחלוקת מטריצות בין חלוקה שמאלית לחלוקה ימנית:

A\b=inv(A)*b חלוקה שמאלית

b/A=b*inv(A) חלוקה ימנית

התוצאה אינה זהה

לדוגמה:

```
» x=A\b
x =
   25.0000
   22.0000
   99.0000
```

הפתרון המומלץ ב- MATLAB הוא הפתרון השני, הוא מבוסס על פחות פעולות מתמטיות ולכן יותר מהיר.

כאשר מספר המשוואות קטן ממספר הנעלמים קיימים אינסוף פתרונות. במקרה זה החלוקה **A\b** תיתן את הפתרון שבו מספר מקסימלי של אפסים.

מטריצות מיוחדות

מטריצה ריקה

```
» a=[1 2 3;4 5 6];
» b=find(a > 10)
b =
     []
```

מטריצה ריקה מוגדרת ב- MATLAB כמטריצה בעלת ממדים אפס.

השם של המטריצה (במקרה זה b) הוא משתנה וקיים ב- WORKSPACE .

מטריצה של אפסים

```
» zeros(3)
ans =
     0     0     0
     0     0     0
     0     0     0
```

מטריצה של אחדים

```
» ones(2,4)
ans =
     1     1     1     1
     1     1     1     1

» ones(3)*pi
ans =
    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416
    3.1416    3.1416    3.1416
```

מטריצת יחידה

```
» eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

מטריצה אקראית (רנדומלית), איברים בין אפס לאחד

```
» rand(3,1)
ans =
    0.2190
    0.0470
    0.6789
```

כדי ליצור מטריצה בעלת ממדים זהים למטריצה אחרת לדוגמא- **size** אפשר להשתמש בפונקציה -

```
a =
     1     2     3
     4     5     6
» ones(size(a))
ans =
     1     1     1
     1     1     1
```

פקודות העוסקות בקבצי- m

כאשר מעונינים לבצע סדרת פעולות ארוכה לא "נוח" לעבוד בחלון הפקודות של ה- MATLAB.

במקרה זה מומלץ לכתוב את תוכנית ולשמור אותה כקובץ עם סיומת *.m .

M-File Functions	
disp(ans)	Display result without identifying variable name
echo	Control the Command window echoing of scrip file commands
input	Prompt user for input
keyboard	Giev control to keyboard temporarily (Type return to quit)
pause	pause until user presses any keyboard key
pause (n)	pause for n seconds
waitforbuttonpress	pause until user presses mouse key or keyboard key

כאשר ה- MATLAB מבצע פקודות מתוך m-file הוא אינו מציג את הפקודות על המסך, הצגה כזאת אפשרית אם משתמשים בפקודות:

```
» echo on
» echo off   - לביטול
```

קלט-פלט

disp - הדפסת מחרוזת או ערך של משתנה ללא שמו.

input - הדפסת מחרוזת והמתנה לקלט מהמקלדת.

```
» x=input('input x : ');
input x : 3

» disp(x)
   3
```

בקרת תוכנית

הפקודות הבאות מאפשרות עצירה זמנית של התוכנית לצורך debugging או שינוי פרמטרים.

keyboard - בקרה זמנית ללוח המקשים. מצב **keyboard** מאפשר הפסקה זמנית של תוכנית והפעלת פקודות דרך ה- WORSPACE . פקודה return משמשת לביטול.

pause - הפסקה זמנית בתוכנית, כל מקש יגרום לביטול.

pause(n) - הפסקה זמנית, n שניות.

waitforbuttonpress - הפסקה זמנית, ביטול ע"י לחיצה על העכבר.

לחיצה על **ctrl-c** מפסיקה את ה- MATLAB באמצע ביצוע פעולה.

נדגים שימוש בפקודות : **keyboard**. נכתוב m-file הנקרא **k_board**.

תוכנית

```
A=5;
B=5;
keyboard
C=A+B
```

פלט

```
» k_board
K» B=10;
K» return
C =
   15
```

הפעלת התוכנית k_board.

הפקודה **keyboard** עצרה את ביצוע התוכנית ואיפשרה לשנות את ערכו של B מ- 5 ל- 10. הסימן «**K** מופיע כאשר נמצאים במצב **keyboard**.

תכנות

קבצים

תכנות ב-MATLAB מבוסס על M-FILES. קיימים שני סוגים של קבצי MATLAB:

- קבצי תוכנית (script).
- קבצי פונקציה (function).

ההבדלים העיקריים בין שני הסוגים הם:

א. קבצי פונקציה מקבלים ארגומנטים כקלט ומחזירים ארגומנטים כפלט.

ב. קבצי M-FILES פועלים על המשתנים הקיימים במרחב העבודה של MATLAB ואילו משתנים פנימיים בקבצי פונקציה (המוגדרים בפונקציה עצמה), הינם משתנים מקומיים, אשר לא מוגדרים מחוץ לפונקציה.

מבנה כללי של קובץ:

function [*out1,out2,...,outm*]=*func_name*(*in1,in2,...,inn*)

% FUNC_NAME Simple function that does nothing

% FUNC_NAME(IN1,IN2,..,) blha, blha, blha …

% … explanation of function …

… program body …

שורה ראשונה של פונקציה תכלול את המילית function, אחריה בסוגריים מרובעים הארגומנטים המוחזרים כפלט מהפונקציה. במידה והפלט כולל ארגומנט אחד בלבד, אין צורך בסוגריים המרובעים. במידה ואין כלל פלט מהפונקציה, ניתן להשאיר סוגריים מרובעים ריקים או לא לכתוב דבר. לאחר סימן השוויון יופיע שם הפונקציה ואחריו בסוגריים עגולים הארגומנטים שמקבלת הפונקציה כקלט. שורה זו תופיע בפונקציות בלבד.

שורה שניה נקראת שורת help 1 (H1). MATLAB יציג שורה זו, כאשר נעשה שימוש בפונקציה *lookfor* או כאשר מתבקשת עזרה (*help*) למחיצה שלמה. שורה זו יכולה להופיע בקבצים משני הסוגים.

שורות ההערות הבאות יוצגו יחד עם שורת H1 כשתתבקש עזרה על הפקודה המסויימת הזו (**help** *function_name*).

לאחר שורות ההערות יופיע גוף התוכנית / פונקציה.

global – הגדרת משתנים כגלובליים

משתנים אשר יופיעו לאחר הפקודה *global* יהיו מוכרים כמשתנים גלובליים ולא מקומיים. רצוי להגדיר את המשתנים הגלובליים בתחילת התוכנית ולמען בהירות התוכנית – יש להשתמש באותיות גדולות.

דוגמה נכתוב פונקציה המחשבת את מקדמי פולינום ממעלה שניה, כאשר המקדם a קבוע ומוגדר כמשתנה גלובלי (mycube.m):

```
function P=mycube(x1,y1,x2,y2)
% MyCube calculate second order polynom coeficients
% mycube(x1,y1,x2,y2) returns coeficiets b and c (P(1) and
P(2))
% of the polynom:  Y=aX^2+bX+c, from data of two points on
the curve.
% A is a global variable
  global a;
  d1=y1-a*x1^2;
  d2=y2-a*x2^2;
  A(1,1)=x1;
  A(1,2)=1;
  A(2,1)=x2;
  A(2,2)=1;
  d=[d1 d2]';
  P=inv(A)*d;
```

הרצת התוכנית:

```
» help mycube
MyCube calculate second order polynom coeficients
mycube(x1,y1,x2,y2) returns coeficiets b and c (P(1) and
P(2))
of the polynom:  Y=aX^2+bX+c, from data of two points on
the curve.
A is a global variable
» global a;
» a = 2;
» P=mycube(1,1,2,1)
p=
     -6
     5
```

טיפוסי משתנים

קיימים שישה טיפוסי מידע בסיסיים ב-MATLAB:

- *double* – הייצוג הסטנדרטי של מספרים ב-MATLAB – 8 בתים.
- *char* – ייצוג סטנדרטי של תווים ומחרוזות. כל תו מיוצג על-ידי 2 בתים.
- *sparse* – צורה לאחסון מטריצות, אשר רוב איבריהן אפסים.
- *storage* – צורת אחסון קומפקטית של MATLAB. לא ניתן לבצע חישובים אריתמטיים על מערכים מטיפוס זה.
- *cell* – מבנה אחסון המאגד מספר משתנים מטיפוסים שונים, הקשורים במבנה לוגי. דומה לטיפוס *struct* בשפות תכנות עיליות.
- *struct* – דומה ל-*cell*.

כל אחד מטיפוסים אלו מיוצג על-ידי מערך רב-מימדי.

אופרטורים

קיימים שלושה סוגים של אופרטורים :

- **אופרטורים אריתמטיים**

לביצוע פעולות מתמטיות על מערכים ומטריצות (+, -, *, *., : ,^. וכו').

- **אופרטורים יחסיים**

מחזירים TRUE או FALSE (<, <=, >, >=, == ו- =~).

הפעלת אופרטור לוגי על שני מערכים מפעילה את האופרטור הלוגי על כל אחד מאיברי המערך ומחזירה מערך בעל מימדים זהים, המכיל 0 ו - 1 בהתאם לתוצאת הפעולה הלוגית. מערך זה נקרא **מערך לוגי**.

- **אופרטורים לוגיים** :

האופרטור	תאור
&	AND לוגי.
|	OR לוגי.
~	NOT לוגי.
xor	EXCLUSIVE OR לוגי.
all	מחזיר 1 אם כל איברי הוקטור שונים מאפס. במטריצה האופרטור פועל על העמודות.
any	מחזיר 1 אם קיים איבר בוקטור השונה מאפס.

דוגמה

```
»A=[1 2 3;4 5 6;7 8 9];
»B=[3 3 2;4 7 6;0 8 1];
»A==B
ans=
     0     0     0
     1     0     1
     0     1     0
»u=[1 0 6 0 9 2];
»v=[4 0 0 1 1 0];
»u & v
```

```
ans=
     1     0     0     0     1     0
»u | v
ans=
     1     0     1     1     1     1
»~u
ans=
     0     1     0     1     0     0
»all(B)
ans=
     0     1     1
»all(u)
ans=
     0
»any(v)
ans=
     1
```

בקרות זרימה

בקרות זרימה (control flow) הן פקודות המשמשות לניתוב זרם הפקודות בתוכנית.

לולאת for

מבנה כללי:

for *index* = *start* : *increment* : *end*

...statements...

end

המשתנה *index* יתקדם מהערך *start* לערך *end* בפסיעות של *increment*. ערך ברירת המחדל של הפסיעה הוא 1.

דוגמה

התוכנית הבאה (mloop.m) מחשבת את ממוצעי העמודות של המערך M.

```
A=[13 5 10 7;5 6 7 8;15 10 13 12];
M=zeros(1,4);
i=0;
for x=A
     i=i+1;
     M(1,i)=sum(x)/length(x);
end
```

הרצת התוכנית:

```
» mloop
» A
A =
     13        5        10        7
     5         6        7         8
     15        10       13        12
» M
M =
     11        7        10        9
```

- נהוג להתחיל שורות השייכות למבנה סגור (כגון לולאה או תנאי) מעט שמאלה מקו השורות של התוכנית הראשית, צורה זו מדגישה את המבנה ההיררכי של התוכנית (כלומר מבנה ראשי הכולל בתוכו תת מבנים).
- מומלץ להימנע משימוש בלולאה כאשר ניתן לקבל תוצאה זהה ללא לולאה.

את הדוגמא הקודמת ניתן (ב- MATLAB) לבצע בשתי דרכים נוספות:

דוגמה

```
» [m,n]=size(A);     % Method number 1
» M=sum(A)/m
M =
    11     7    10     9

» M=mean(A)          % Method number 2
M =
    11     7    10     9
```

צורות אלה מומלצות משום שהן מנצלות את היכולת של ה- MATLAB לבצע פעולות על מערכים, ולכן חוסכות את זמן הריצה הארוך יחסית של הלולאה.
פעולה נוספת המומלצת לצורך חיסכון בזמן היא הגדרה מראש של מערכים, הנבנים תוך כדי ריצת הלולאה כמערכי אפסים בגודלם הסופי (מקסימלי).

לולאתwhile (לולאה מותנת)

מבנה כללי:

while *expression*

...statements...

end

הלולאה מבצעת כל עוד *expression* שווה לביטוי אמת (1).

דוגמה

נכתוב את התוכנית הבא (eps1.m), לחישוב הערך *eps* (המספר הקטן ביותר המוכר ל- MATLAB):

```
num=0;
eps=1;
while (1+eps)>1
     eps=eps/2;
     num=num+1;
end
```

הרצת התוכנית:

```
» eps1
num =
    53
» eps=2*eps
eps =
  2.2204e-016
```

התוכנית מחפשת בצורה איטרטיבית את המספר הקטן ביותר שאפשר להוסיף למספר 1, כך שסכומם (ב- MATLAB) יהיה גדול מ- 1.

משפטי תנאי if, else, elseif

מבנה כללי:

if *expression1*

...statements...

elseif *expression2*

...statements...

else

...statements...

end

במקרה זה יתכנו המקרים הבאים:

- אם ביטוי 1 מתקיים תבוצע סדרת הפקודות הראשונה.
- אם ביטוי 1 לא מתקיים תתבצע בדיקה של ביטוי 2. במידה וביטוי 2 מתקיים - תבוצע סדרת הפקודות השניה.
- אם אף אחד מהביטויים אינו מתקיים, תבוצע סדרת הפקודות השייכת ל- else.

ניתן להשתמש במספר ביטוים מסוג elseif. השימוש ב- else ו/או elseif אינו חובה.

דוגמה

נבדוק האם מספר אקראי נמצא בתחום [0 - 0.1) , [0 - 0.9) או [0.9 - 1] (arand.m):

```
a=rand(1)
if a < 0.1
     disp('range = [0-0.1)')
elseif (a < 0.9) & (a >= 0.1)
     disp('range = [0.1-0.9)')
else
      disp('range = [0.9-1]')
end
```

פלט התוכנית:

```
» arand
a =
    0.8913
range = [0.1-0.9)
```

דוגמה

שימוש במשפט תנאי בתוך לולאת for, לחישוב הערך eps (eps2.m):

```
eps2=1;
for num=1:1000;
eps=eps/2;
     if (1+eps)<=1
                eps=eps*2;
                break
     end   % {if}
end   % {for}
```

פלט התוכנית:

```
» eps2
eps =
  2.2204e-016
```

switch-case-otherwise

מבנה כללי:

switch *expression* (scalar or string)

case *value1*

...statements...

case *value2*

...statements...

:

otherwise

...statements...

end

פקודת switch משמשת למקרים בהם מעוניינים לבצע פעולות שונות, בהתאם לערכו של משתנה או ביטוי מסוג סקלר או מחרוזת.

דוגמה (swit1.m):

```
go=1;
while go
     qu=input('input 1,2,3,4,5 or another number to quit');
     switch qu
          case {2,4}
               disp(' 2 or 4')
          case {1,2,3}
               disp(' 1 or 2 or 3')
          case 5
               disp(' 5')
          otherwise
               go=0;
          disp(' end');
     end
end
```

הרצת הקובץ:

```
» swit1
input 1,2,3,4,5 or another number to quit 1
 1 or 2 or 3
input 1,2,3,4,5 or another number to quit 2
 2 or 4
input 1,2,3,4,5 or another number to quit 3
 1 or 2 or 3
input 1,2,3,4,5 or another number to quit 4
 2 or 4
input 1,2,3,4,5 or another number to quit 5
5
input 1,2,3,4,5 or another number to quit 6
end
```

בשונה מ- switch בשפות כגון C, בהן לאחר פקודת ה-switch ממשיכה התוכנית מה-case המתאים עד לפקודת break או סוף הלולאה, ב-MATLAB לאחר ה-case המתאים התוכנית יוצאת מהלולאה.

פונקציות פנימיות

בקובץ MATLAB ניתן להגדיר מספר פונקציות בקובץ אחד. הפונקציה הראשית תופיע בראש הקובץ ותישא את שם הקובץ. הפונקציות האחרות תופענה בהמשך הקובץ. פונקציות המוגדרות באותו קובץ אינן יכולות לפנות למשתנים פנימיים של פונקציות אחרות (אלא אם המשתנים הוגדרו כמשתנים גלובליים).

בפניה לפונקציה כלשהיא, מחפש MATLAB פונקציה זו בסדר הבא:

- פונקציה פנימית.
- פונקציה פרטית (יוסבר בהמשך).
- פונקציה כללית.

פונקציה פרטית

פונקציות השוכנות תחת מחיצה מיוחדת בשם private. ניתן לגשת אליהן רק מתוכניות במחיצת האב (parent directory). מכיוון שפונקציות פרטיות אינן נגישות מחוץ למחיצת האב, ניתן להגדיר מחדש פונקציות בעלות שם זהה לפונקציות ממחיצות אחרות.

המרת מחרוזות

eval('string') – הרצת ביטוי המופיע במחרוזת.

feval('function') – הרצת פונקציה המופיעה כמחרוזת.

דוגמה

```
» t='i+j';
» i=2;
» j=3;
» eval(t)
ans=
     5
»f='sin';
» k=pi/2;
» feval(f,k)
ans=
     1
```

☝ עדיף להשתמש כשניתן בפונקציה ***feval***, שכן היא יעילה יותר מהפונקציה ***eval*** וניתנת להידור.

אופטימיזציה

MATLAB הינה תוכנית שתוכננה לביצוע פעולות על מטריצות ווקטורים. וקטוריזציה של לולאות תאיץ את ביצוע התוכנית.

דוגמה יצירת וקטור המכיל ערכי סינוס מ0- עד 10*pi (sin1.m):

```
tic
i=0;
for t=0:0.01:10
     i=i+1;
y(i)=sin(t);
end
toc
```

תוכנית זו לאחר וקטוריזציה (sin2.m):

```
tic
t=0:0.01:10;
y=sin(t);
toc
```

הרצה של שתי התוכנית תראה כי התוכנית השניה מהירה מהתוכנית הראשונה.

גרפיקה דו- ממדית

plot(x,y) – שרטוט עקום דו-מימדי.

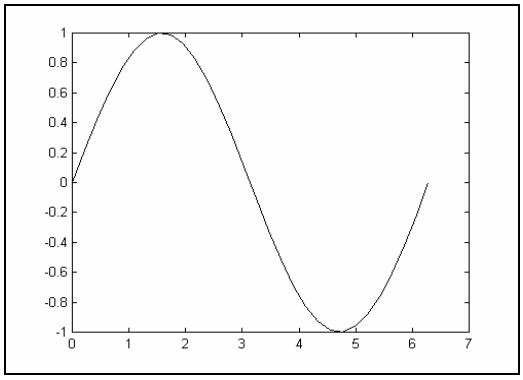
פקודה זו מציירת גרף של וקטור **y** כנגד וקטור **x**., כאשר כל נקודה בגרף מיוצגת על-ידי איברים מתאימים בוקטורים (הוקטורים חייבים להיות בממד שווה – עם אותו מספר איברים). הנקודות המוגדרות בוקטורים **x** ו-**y** מחוברות בקווים ישרים. במקרה בו **y** הנו מערך בגודל $n \times m$ ו-**x** הנו וקטור באורך m יתקבלו n עקומות בצבעים שונים.

דוגמה

שרטוט גרף של sin(x) בתחום $0-2\pi$:

```
» x=linspace(0,2*pi,30);
» y=sin(x);
» plot(x,y);
```

הפלט המתקבל:



התקבל עקום לפי 30 נקודות (גודל הוקטורים **x** ו-**y**), המחוברות בקווים ישרים.

אם יצוין רק וקטור נתונים אחד, לדוגמא plot(y) יתקבל גרף של ערכי y(i) כנגד i , כלומר y כנגד האינדקסים של y (i=1,2,....,n , n=גודל הוקטור **y**).

הפקודה **plot** מאפשרת לקבוע את סוג וצבע הקו על-ידי הוספת מחרוזת s:

```
» plot(x,y,s)
```

ניתן לבחור צבע וסוג מתוך האפשרויות הבאות:

Symbol	Color	Symbol	Marker	Symbol	Line Style
b	כחול	.	נקודה	-	קו רציף
g	ירוק	o	עיגול	:	נקודות
r	אדום	x	איקס	-.	קו – נקודה
c	כחול – ירוק	+	פלוס	--	מקווקו
m	אדום – ארגמן	*	כוכב		
y	צהוב	s	ריבוע		
k	שחור	d	יהלום		
w	לבן				

הפקודה plot יכולה לשמש גם לשרטוט מספר עקומות בגרף אחד.

<u>דוגמה</u>

```
» plot(x1,y1,'r',x2,y2,'b',x3,y3,'y')
```

הפקודה הנ"ל תשרטט שלוש עקומות: y1 כנגד x1 באדום, y2 כנגד x2 בכחול ו y3 כנגד x3 בצהוב.

<u>עיצוב תרשים</u>

xlabel('string') – הגדרת כותרת ציר x

ylabel('string') – הגדרת כותרת ציר y

legend('string1','string2',…) – הגדרת מקרא ומיקומו באמצעות עכבר

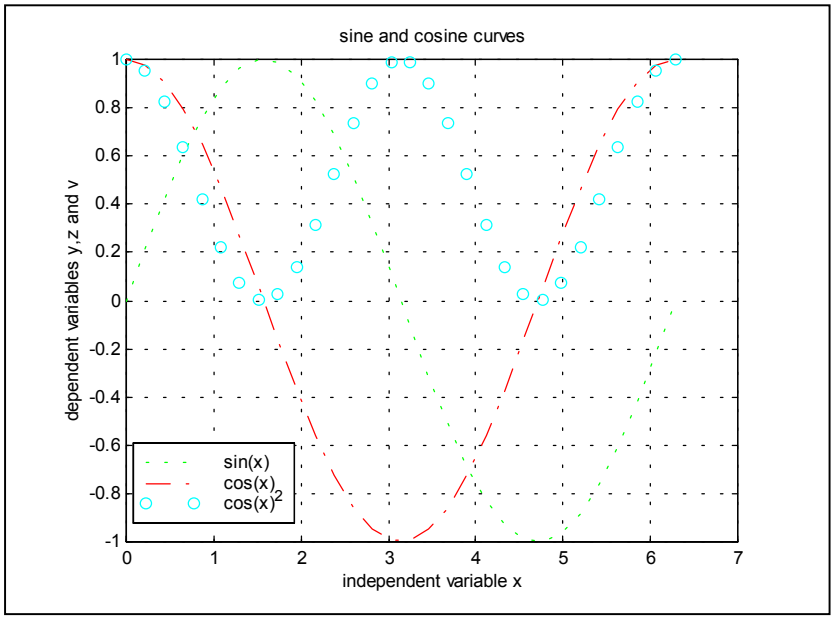
title('string') – הגדרת כותרת תרשים

<u>דוגמה</u>

```
» x=linspace(0,2*pi,30);
» y=sin(x);
» z=cos(x);
» v=cos(x).^2;
» plot(x,y,'g:',x,z,'r-.',x,v,'co');
» grid on;
```

```
» ylabel('dependent variables y,z and v');
» xlabel('independent variable x');
» legend('sin(x)','cos(x)','cos(x)^2');
» title('sine and cosine curves');
```

הפלט שיתקבל:



grid – הוספה/הסרה של רשת הקורדינטות (on/off)

axis([Xmin Xmax Ymin Ymax]) - קביעת תחום התרשים

כברירת מחדל תחום הגרף מוגדר על פי ערכי המקסימום והמינימום של המשתנים - התלוי והבלתי תלוי. הפקודה axis מעדכנת את התחום בפלט לפי Xmax, Xmin ו- Ymin,Ymax.

hold – הפעלה/ביטול הדפסת גרף על גרף בחלון גרפי קיים (on/off)

כברירת מחדל שרטוט פלט חדש מנקה את הפלט הקודם מהחלון הגרפי (figure). שימוש בפקודה **hold on** מבטל את מצב ברירת המחדל ומאפשר הדפסת גרף על גרף. **hold off** מחזירה למצב ברירת המחדל.

<u>SUBPLOTS – מספר גרפים בחלון אחד</u>

להצגת מספר גרפים שונים בחלון גרפי אחד ניתן לחלק את החלון ל $n \times m$ תת-חלונות (subfigures).

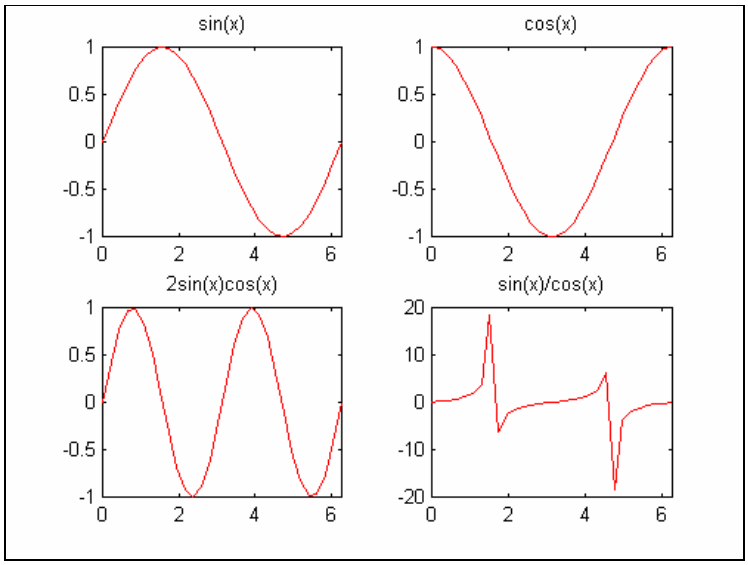
subplot(m,n,i) – חלוקת החלון הגרפי להצגת מספר גרפים

פקודת **plot** שתופיע אחרי פקודה זו תדפיס בתת חלון מס' i (כאשר תת החלונות נספרים משמאל לימין ומלמעלה למטה).

<u>דוגמה</u>

```
» x=linspace(0,2*pi,30);
» y=sin(x);z=cos(x);
» a=2*sin(x).*cos(x);
» b=sin(x)./(cos(x)+eps);
» subplot(2,2,1);
» plot(x,y),axis([0 2*pi -1 1]),title('sin(x)')
» subplot(2,2,2);
» plot(x,z),axis([0 2*pi -1 1]),title('cos(x)')
» subplot(2,2,3);
» plot(x,a),axis([0 2*pi -1 1]),title('2sin(x)cos(x)')
» subplot(2,2,4);
» plot(x,b),axis([0 2*pi -20 20]),title('sin(x)/cos(x)')
```

יתקבל ה- פלט הבא:



גרפיקה במספר חלונות (figures)

ב- Matlab נתן לשרטט מס׳ גרפים בחלונות נפרדים. לכל חלון מס׳, באמצעותו ניתן לפנות אל החלון ולבצע פעולות שתהיינה רלבנטיות רק לגביו.

figure(h) – פתיחת חלון גרפי חדש

דוגמה

```
» figure(2)     % creates new figure number 2
» close(2)      % close figure number 2
» figure(1)
» figure(3)
» close all     % close all figures
```

הפקודה figure(x) גורמת להצגת חלון מספר x במידה והוא כבר קיים.

הפקודה close ללא ציון מספר החלון תסגור את החלון הגרפי הנוכחי.

clf(n) ניקוי חלון מספר n (ברירת מחדל – החלון הנוכחי).

הגדלה או הקטנה

zoom – הגדלה או הקטנה של גרף דו-מימדי.

ניתן להגדיל או להקטין את הגרף באופן ידני ע״י שימוש בפקודת **zoom on**.

אחרי הקשת enter יש למקם את הסמן (באמצעות העכבר) בנקודה הרצויה (נק׳ מרכז התמונה המוגדלת או מוקטנת). לחיצה על הכפתור השמאלי של העכבר תגדיל את הגרף ולחיצה על הימני תקטין אותו. אפשר גם לקבוע תחום מלבני ע״י לחיצה וגרירה של העכבר.

לביטול **zoom on** יש להקיש **zoom off**.

<u>פקודות קלט גרפיות</u>

[x,y]=ginput(n) – קלט גרפי מהעכבר

הקורדינטות של n נקודות שיסומנו על הגרף באמצעות העכבר ישמרו במשתנים x ו-y. ללא ציון מס' n , המחשב יאפשר לסמן נקודות עד לחיצה על המקש ENTER.

הפקודה **[x,y,button]=ginput(n)** תשמור בנוסף לקורדינטות את מספר ה- ASCII של המקש שנלחץ.

שרטוט

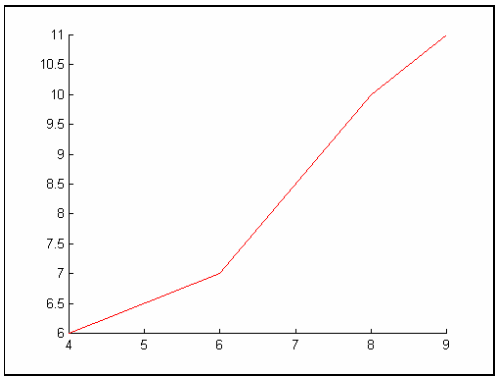
line(x,y) – שרטוט קו

שרטוט קווים מהנקודות y(i),x(i) לנקודות x(i+1), y(i+1) (כמו ב- plot).

<u>דוגמה</u>

```
» x=[4 6 8 9]
x =
     4     6     8     9
» y=[6 7 10 11]
y =
     6     7    10    11
» line(x,y)
```

הפלט המתקבל:



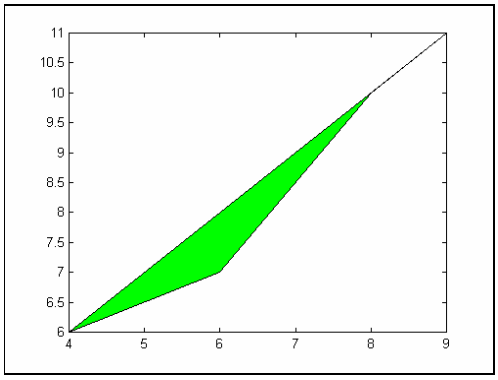
fill([x,y,c]) – שרטוט פוליגון

הפקודה צובעת את התחום המוגדר ע"י x,y בצבע c.

דוגמה נפעיל את הפקודה על הוקטורים המוגדרים x ו-y:

```
» fill(x,y,'g')
```

הפלט המתקבל:



סוגי גרפים

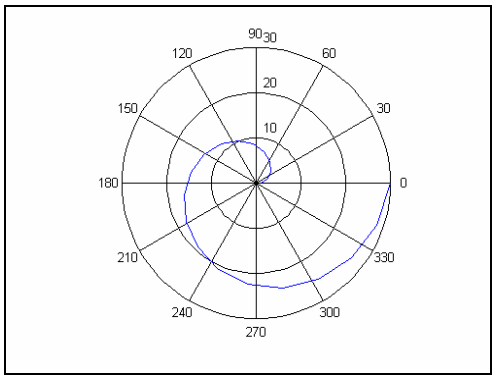
polar(theta,rho,s) – שרטוט גרף פולרי

גרף פולרי של הרדיוס rho כנגד הזוית theta בסגנון s.

<u>דוגמה</u>

```
» r=linspace(1,30,20);
» teta=linspace(0,2*pi,20);
» polar(teta,r,'b')
```

הפלט המתקבל:

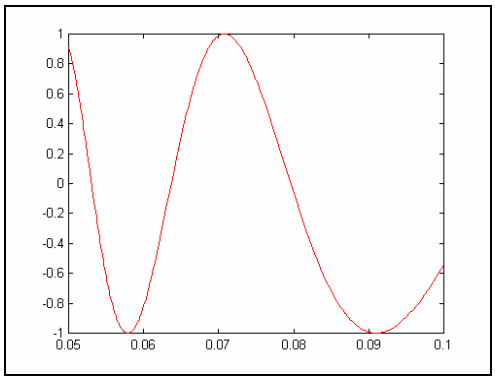


fplot('fun_name',[xmin amax]) – ציור פונקציה בתחום מסוים

דוגמה

```
» fplot('sin(1 ./ x)', [0.05 0.1])
```

הפלט המתקבל:



hist(Y,N) - גרף המתאר את התפלגות האיברים בווקטור Y.

N סקלר מציין את מס' המרווחים בין האיברים המינימלי והמקסימלי ב- Y. כאשר N הוא ווקטור, מתקבלת עבור כל איבר N(i) עמודה המציינת כמה איברים ב- Y קטנים מ- $\frac{N_i + N_{i+1}}{2}$ וגדולים מ- $\frac{N_i + N_{i-1}}{2}$.

bar(X,Y,'linetype') – שרטוט העמודות של מטריצה M-by-N כ-M קבוצות של N עמודות.

ווקטור x מייצג את מיקום העמודות על גבי הציר האופקי. ללא ווקטור x העמודה y(n) תמוקם בנקודה n על גבי הציר האופקי.

stem(X,Y,'linetype') – שרטוט גרף בדיד.

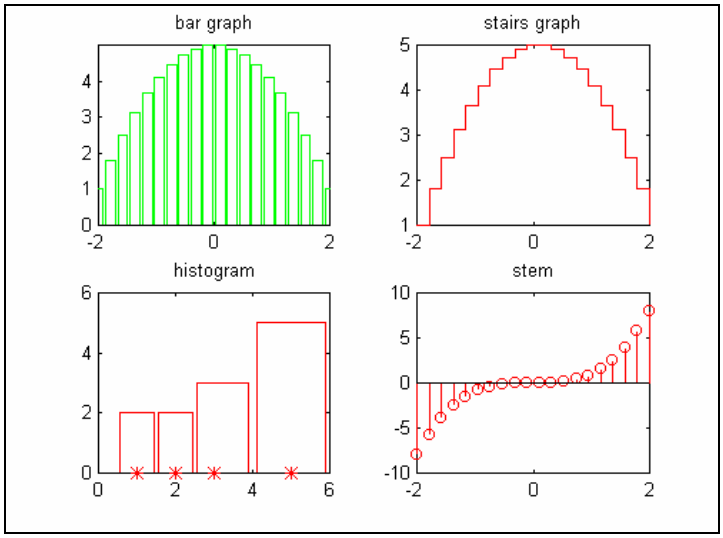
stairs(X,Y) – שרטוט גרף מדרגות.

דוגמה (5.9)

```
» x=linspace(-2,2,20);
» y=-x.^2+5;
```

```
» yy=[1 1 2 3 2 5 6 3 4 5 6 7 ];
» nn=[1 2 3 5];
» z=x.^3;

» subplot(2,2,1);
» bar(x,y,'g')
» axis([-2 2 0 5])
» title('bar graph');

» subplot(2,2,2)
» stairs(x,y);
» title('stairs graph');

» subplot(2,2,3)
» hist(yy,nn);
» title('histogram');

» subplot(2,2,4)
» stem(x,z);
» title('stem');
```

הפלט המתקבל:



✍ errorbar(X,Y,L,U,SYMBOL) – גרף שגיאות

גרף הכולל את סימון העקום המוגדר ע״י וקטור Y כנגד וקטור X וסימון השגיאות הנתונות בווקטורים L (שגיאה כלפי מטה) ו- U (שגיאה כלפי מעלה).

הווקטורים X Y L U חייבים להיות בעלי ממד זהה, עבור מטריצות יתקבלו מס׳ גרפים בהתאם למס׳ וקטורי העמודה.

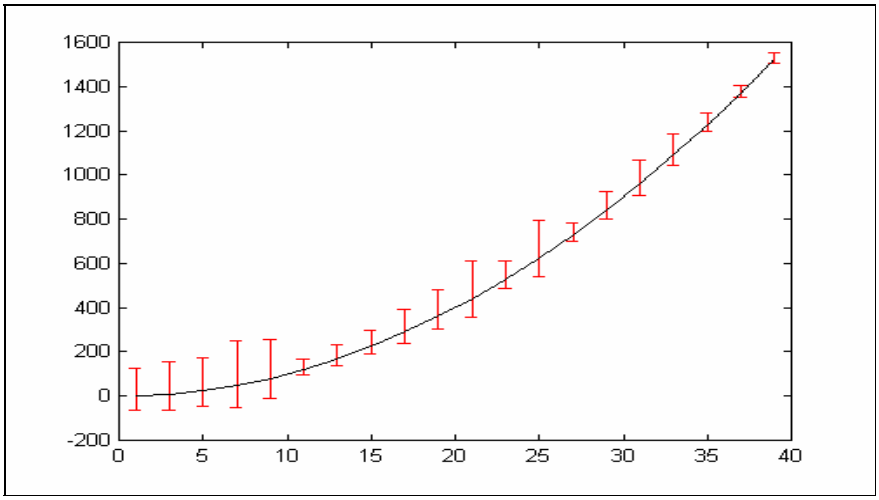
ה- SYMBOL הוא מחרוזת צבע ו/או סימן של העקום (כמו ב- **plot**).

גרף של y(n) כנגד n ועמודות שגיאה בתחום [y-L,y+L] יתקבל על-ידי הפקודה **errorbar(Y,L)**.

☜ דוגמה (5.10)

```
» y=[1:2:40];
» x=rand(size(y))*100;
» errorbar(y,y.^2,x,2.*x,'k');
```

הפלט המתקבל:



compass(X,Y,'S') – שרטוט גרף מצפן

גרף חצים שיוצאים מהראשית, גודלו וכיוונו של כל חץ הוא כשל הווקטור המתאים $X_n\hat{i} + Y_n\hat{j}$ או של המספר המרוכב X_n+iY_n. את X ו- Y אפשר להחליף בווקטור אחד Z שאבריו מספרים מרוכבים.

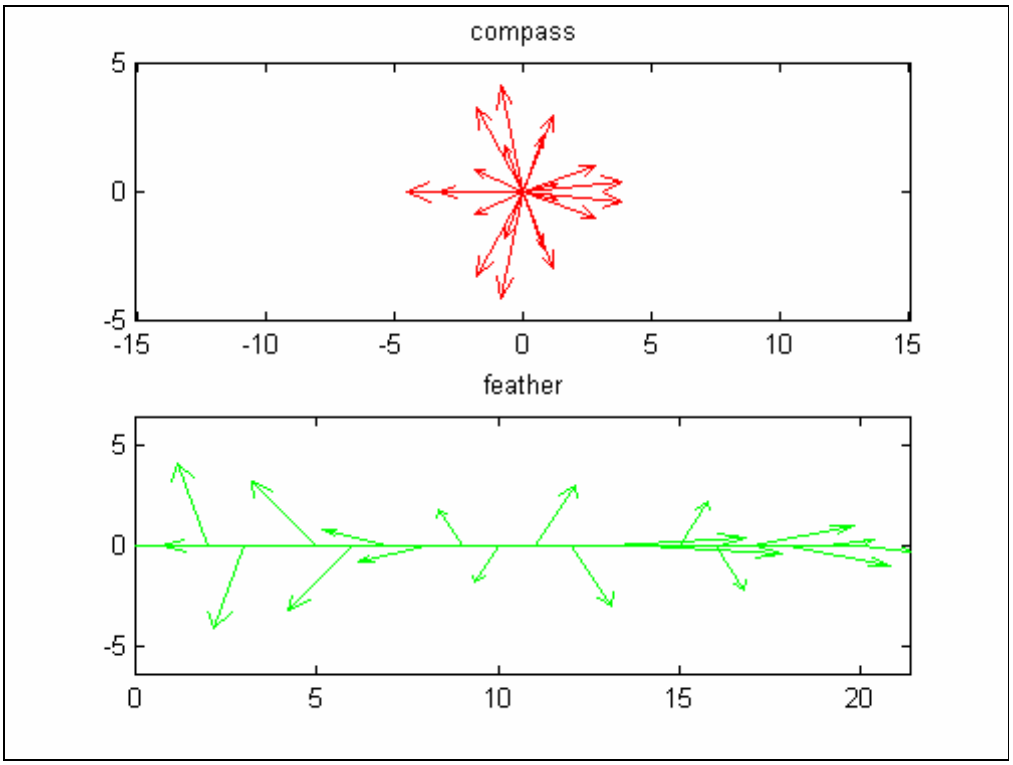
feather(X,Y,'S') – שרטוט גרף feather

גרף חיצים כמו compass אלא שכאן החצים יוצאים מהציר האופקי וממוקמים ברווחים שווים אחד מהשני.

דוגמה (5.11)

```
» z=eig(randn(20,20))     % create vector with 20 random
complex numbers
» subplot(2,1,1)
» compass(z,'r')'
» title('compass');
» subplot(2,1,2)
» feather(z,'g');
» title('feather');
```

הפלט המתקבל:



גרפיקה תלת ממדית

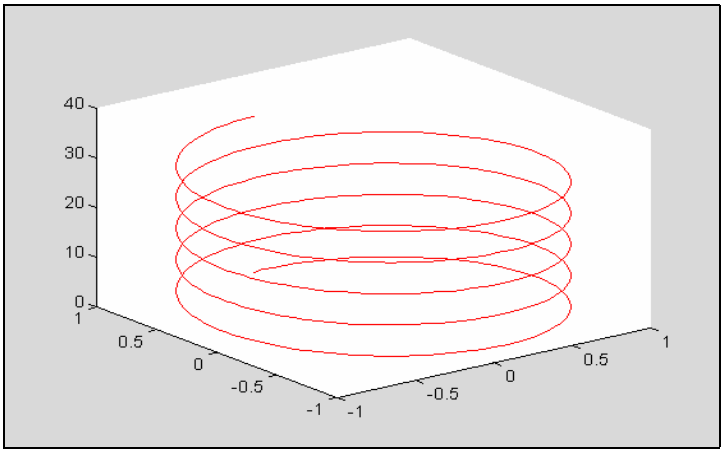
plot3(x,y,z,'c') – ציור עקום במרחב תלת-מימדי

פקודה זו היא המקבילה התלת ממדית לפקודה ***plot***, לציור ישרים ועקומות במרחב תלת-ממדי. כמו ב- ***plot*** הדו-ממדית, **x,y,z** הנם ווקטורים בגודל n, המייצגים n נקודות במרחב. (לדוגמא, נקודה - i בגרף היא - (x(i),y(i),z(i). בפלט שיוצג על המסך יחוברו הנקודות בקווים ישרים. ניתן לקבוע את הצבע ו/או הסימן של העקום ע"י הוספת מחרוזת **c**.

דוגמה שירטוט ספירלה במרחב:

```
» t=0:pi/50:10*pi;
» x=sin(t);
» y=cos(t);
» z=t;
» plot3(x,y,z,'r');
```

הפלט המתקבל:



mesh(x,y,z,c) - שרטוט משטחים המייצגים פונקציות מהצורה z=f(x,y)

משטח מוגדר ב- matlab כקואורדינטה z מעל תחום מלבני במישור x-y. הפלט המוצג על המסך מתקבל ע"י חיבור נקודות שכנות בקווים ישרים.

z- מערך בגודל m על n, **x** - וקטור בגודל m ו-**y** - וקטור בגודל n.

בפקודה ***mesh*** המשטח משורטט כרשת של נקודות בגובה- z, לפי הקשר - $z(i,j) = f(x(i),y(j))$.

ניתן להשתמש בפקודה גם כאשר **x** ו- **y** הם מערכים בגודל של z, במקרה זה יהיה הקשר $z(i,j) = f(x(i,j),y(i,j))$.

צבע המשטח יכול לשמש כממד רביעי ע"י הגדרת מערך נוסף **c** המתאים צבע לכל נקודה. כלומר צבע הנקודה יהיה פונקציה של מיקומה $c(i,j) = g(x(i),y(j),z(i,j))$.

בפקודת mesh ללא מערך c יופיע צבע כפונקציה של הגובה z.

הפקודה ***[xi,yi]=meshgrid(x,y)*** מאפשרת לבנות את המערכים x,y,z כך שיתארו פונקציה אנליטית $z = f(x,y)$ ויתאימו לשרטוט ע"י הפקודה mesh.

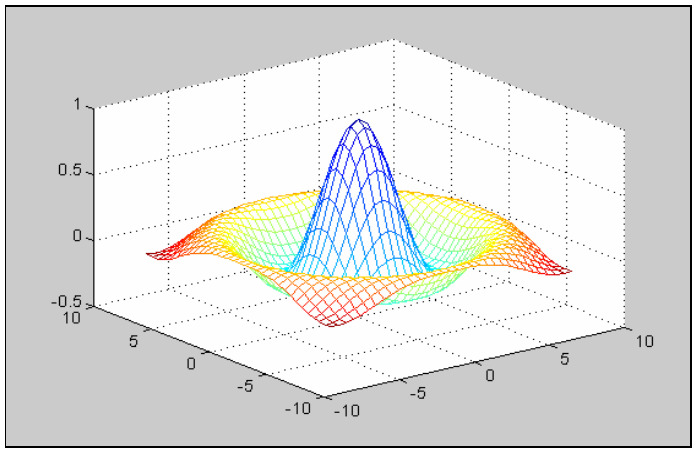
על סמך ווקטורי הקלט: **x** בגודל n ו-**y** בגודל m בונה הפקודה את המערך **xi** המכיל m ווקטורי שורה x ואת המערך **yi** המכיל n ווקטורי עמודה y.

לאחר בניית המערכים yi, xi בגודל m על n ניתן לבנות את z באמצעות פונקציות ופעולות של מערכים, כך ש $z(i,j) = f(xi(i,j),yi(i,j))$ - .

דוגמה שירטוט משטח המתאר את הפונקציה $z=\frac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}$:

```
» x=-8:0.5:8;
» y=x';
» [xi,yi]=meshgrid(x,y);
» r=sqrt(xi.^2+yi.^2)+eps;
» z=sin(r)./r;
» mesh(x,y,z,r)
```

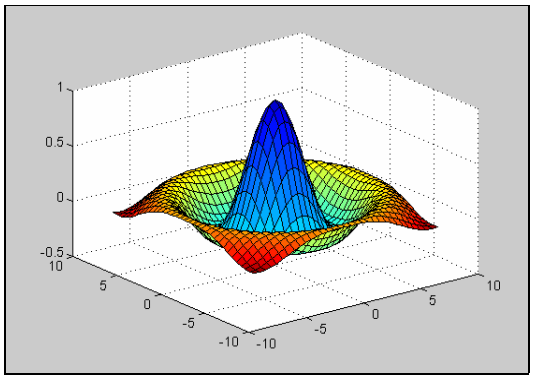
הפלט המתקבל :



הצבע בגרף יחסי למרחק האופקי מהנקודה (x,y)=(0,0) המיוצג ע"י המשתנה r.

הפקודה ***surf(x,y,z,c)*** מציירת משטח כמו הפקודה ***mesh***, אך במקרה זה תהיינה משבצות הרשת מלאות. עבור נתונים (x,y,z,r) זהים לדוגמא הקודמת יתקבל :

```
» surf(x,y,z,r)
```



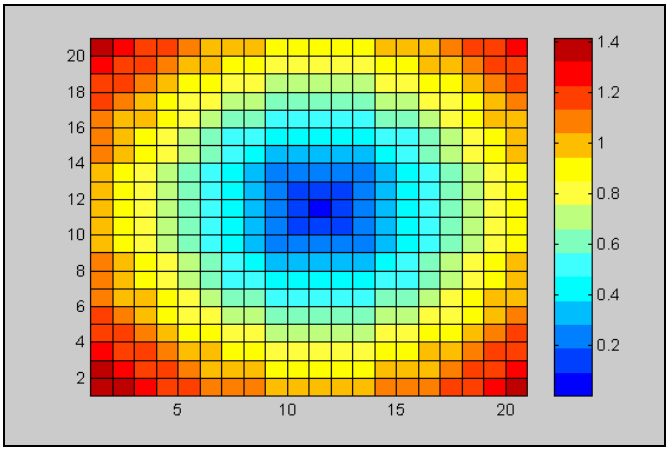
הפקודה **pcolor(x,y,z)** מאפשרת להמחיש מטריצת מספרים בצורה גרפית, כרשת משבצות בה כל משבצת מייצגת איבר במטריצה. צבע המשבצת תלוי בערך האיבר. המשטח המתקבל הוא למעשה ***surf(x,y,z)*** במבט מלמעלה. ללא ציון הווקטורים **x,y** תתקבל תמונה של המטריצה על פי האינדקסים המתאימים לכל איבר במטריצה (כאשר הראשית בצד שמאל למטה).

ניתן לקבל סקלת צבע ע״י הוספת הפקודה **colorbar**.

דוגמה

```
» [xi,yi]=meshgrid(-1:0.05:1);
» zi=sqrt(xi.^2+yi.^2);
» pcolor(zi) ;
» colorbar;
```

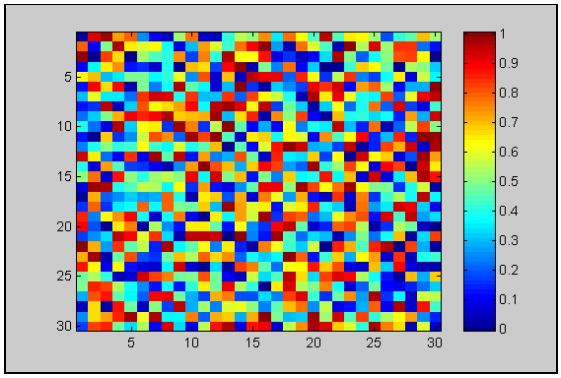
יתקבל הפלט:



פקודה המבצעת פעולה דומה היא **imagesc(z,[cmin cmax])**. הפקודה יוצרת תמונה דו-ממדית על סמך מערך מספרים המייצגים צבע. כאשר z הוא המערך ו- [cmin cmax] הוא תחום המספרים שעבורם יתקבלו הבדלי צבע.

דוגמה הצגה גרפית של מטריצת מספרים אקראיים בין 0 ל- 1 :

```
» imagesc(rand(30,30),[0 1]);
» colorbar
```



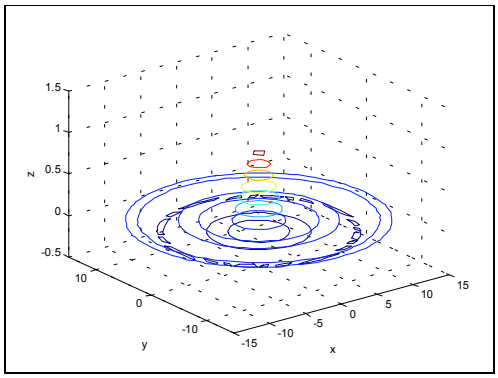
contour(x,y,z,n) - ציור משטח באמצעות קווים שוי גובה (טופוגרפיה)

הפקודה משרטטת מפת קווי גובה דו-ממדית (המתארת משטח תלת-ממדי). x,y,z כמו ב mesh, n סקלר שמציין את מספר קוי הגובה. ניתן לאלץ ציון גבהים מסוימים ע"י הגדרת n כווקטור המכיל את הגבהים.

מפת קווי גובה תלת ממדית תתקבל ע"י שימוש בפקודה contour3(x,y,z,n).

דוגמה

```
» x=-15:1:15; y=x';
» [xi,yi]=meshgrid(x,y);
» r=sqrt(xi.^2+yi.^2)+eps;
» z=sin(r)./r;
» figure(1)
» contour(x,y,z,8);
» xlabel('x'); ylabel('y');
» figure(2)
» contour3(x,y,z,8);
» xlabel('x'); ylabel('y'); zlabel('z');
```



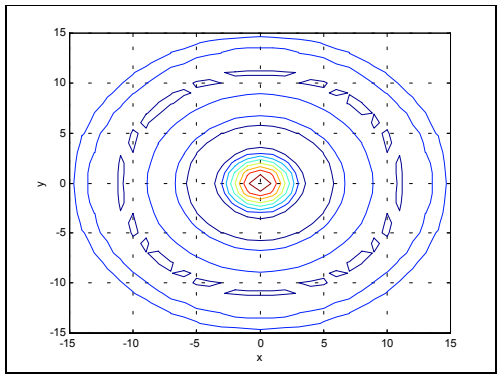


figure 2 figure 1

ניתן לקבל קווי גובה, הכוללים גם את ציון ערך הגובה על-ידי הוספת הפקודה ***clabel*** בצורה הבאה:

```
» [c,h]=contour(x,y,z,n);
» b=clabel(c,h);
```

במקרה זה, בנוסף לציור קווי הגובה לפי הקורדינטות x,y,z תתקבל גם מטריצה c ווקטור h.

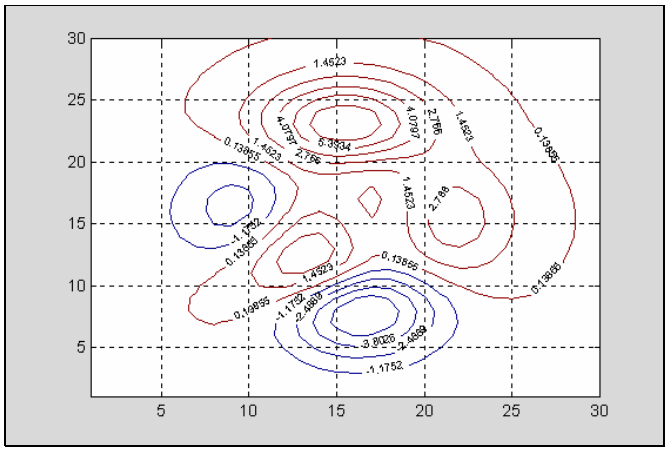
המטריצה c בעלת 2 שורות בנויה מתת- מטריצות המתארות קווי גובה. האיבר השמאלי עליון בכל תת מטריצה מציין את גובה הקו הספציפי, האיבר השמאלי תחתון מציין את מספר הנקודות היוצרות את קו הגובה, בהמשך השורה הראשונה והשניה נמצאות הקאורדינטות x,y .

h הינו ווקטור עמודה בגודל n שמשייך מספר לכל קו גובה (handle), לצורך זיהוי קו הגובה ושינוי תכונותיו (צבע, צורה, מיקום וכו').

באמצעות הפקודה ***clable(c,h)*** יאותרו קווי הגובה המתאימים ולכל אחד מהם תצורף תווית המציינת את ערך הגובה.

שיוך תוויות הטקט לווקטור b מאפשר את שינוי התכונות, לדוגמא שינוי גודל הפונט.

```
» z=peaks(30);                    %  produces an 30-by-30
matrix
» [c,h]=contour(z,10);
» b=clabel(c,h);
» set(b,'fontsize',7);            %  change the font size to
7
```



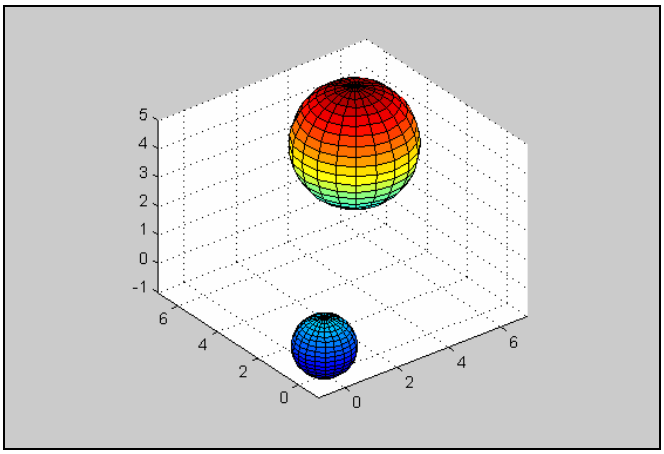
[x,y,z]=sphere(n) - ציור כדור

הפונקציה sphere בונה מטריצות x,y,z בגודל (n+1)*(n+1) כך שהפקודה ***surf(x,y,z)*** תצייר כדור בגודל יחידה שמרכזו בראשית. להזזת הכדור והגדלת רדיוסו יש לשנות את x,y,z בהתאם לצורך.

ציור מיידי של כדור יחידה אפשרי באמצעות ***sphere(n)***.

דוגמה

```
» sphere(20)
» [x,y,z]=sphere(20);
» x=x.*1.5+1;y=y.*1.5+4;z=z.*1.5+1
» axis('equal');
```



ציור גופי סיבוב בקואורדינטות גליליות

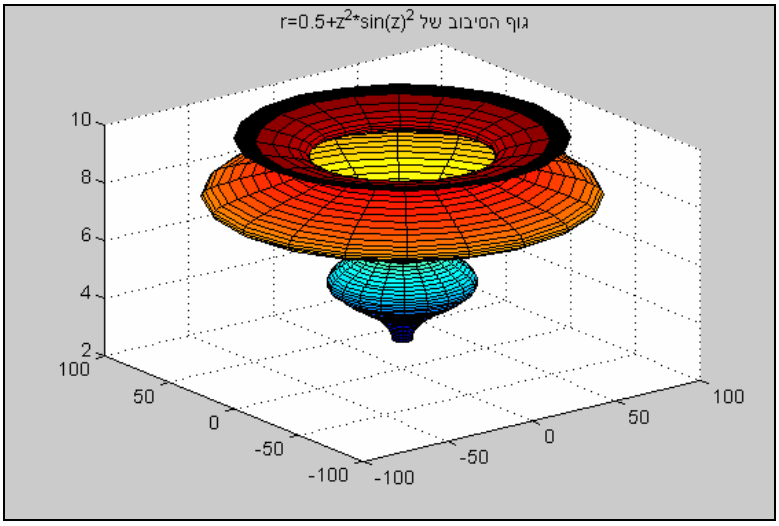
לציור גוף הסיבוב סביב ציר ה- z של הפונקציה R=f(z) בתחום z=[a,b] נבנה תחילה ווקטורים של z ו R על פי התחום.

באמצעות הפונקציה ***[x,y,z']=cylinder(R,n)*** נקבל את המטריצות הקרטזיות לתיאור גוף הסיבוב כאשר z' בתחום [0,1].

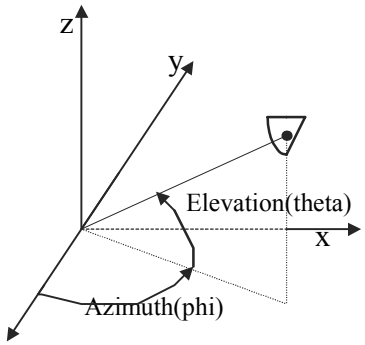
לציור גוף הסיבוב בתחום המקורי נבצע טרנספורמציה z=z'*(b-a)+a) (z'=(z-a)/(b-a)). גודל המטריצות x,y,z יהיה (length(R),n+1) .

דוגמה

```
a=3;b=10;
z=3:0.1:10;
r=(0.5+sin(z).^2).*z.^2;
[x,y,zl]=cylinder(r,20);
zm=zl*(b-a)+a;
surf(x,y,zm)
title('  r=0.5+z^2*sin(z)^2 גוף הסיבוב של ')
»rotbd
```



שינוי כיוון המבט (view point) בפלט תלת מימדי



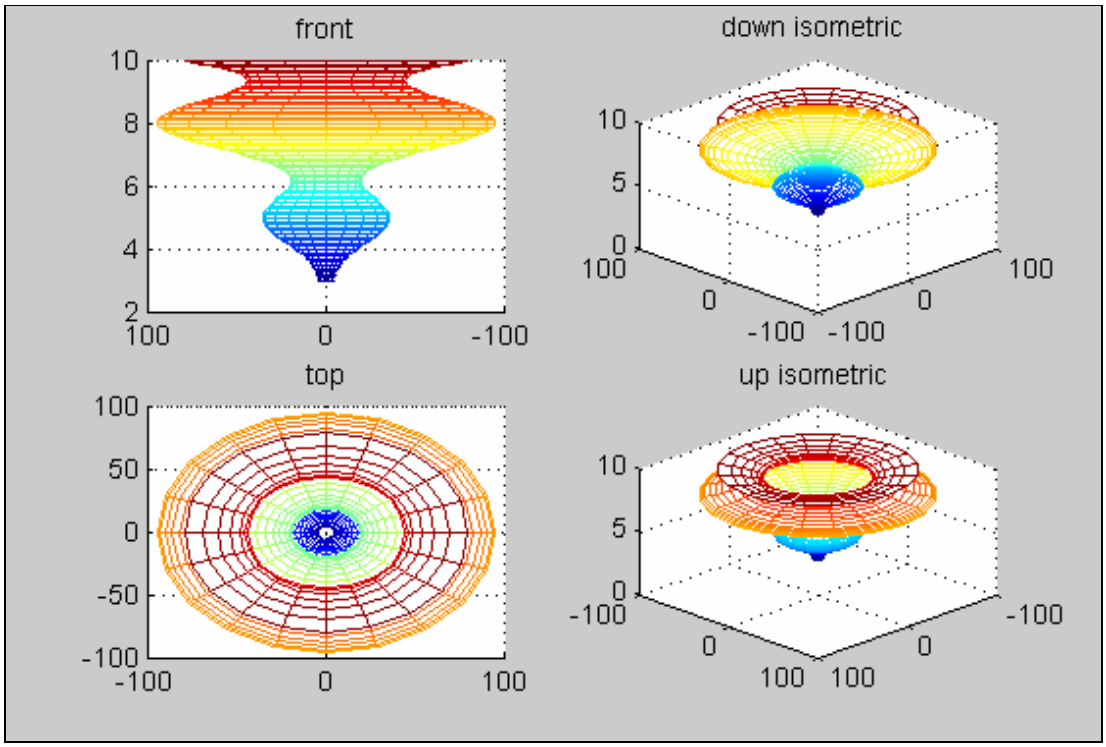
שינוי כיוון המבט אפשרי בשתי דרכים:

1. שימוש בפקודה ***view([phi,theta])***, כאשר הקלט הוא ווקטור בעל שני איברים, האיבר השמאלי הוא זוית הסיבוב (Azimuth) סביב ציר z והימני זוית ההגבהה (elevation) יחסית למישור ה-xy (הזוויות במעלות).

2. שימוש בפקודה view([x,y,z]), כאשר הקלט הוא ווקטור תלת ממדי היוצא מהראשית כך שכיוון המבט הוא מקצה הווקטור לכיוון הראשית, אין משמעות לגודל הווקטור אלא רק לכיוונו.

דוגמה עבור הציור בעמוד הקודם(גוף הסיבוב)

```
 subplot(2,2,1)
 mesh(x,y,zm);view([0 1 0]);title('front');
 subplot(2,2,2)
 mesh(x,y,zm);view([1 1 -1]);title('down isometric ');
 subplot(2,2,3)
 mesh(x,y,zm);view([0 0 1]);title('top');
 subplot(2,2,4)
 mesh(x,y,zm);view([1 1 1]);title('up isometric ');
»proj
```



ערכי ברירת המחדל של כיוון המבט בציור תלת מימדי:

Azimuth(phi)=-37.5^{o}

Elevation(theta)=30°

שינוי כיוון המבט באופן ידני באמצעות העכבר (ב matlab5) יעשה על-ידי הפקודה ***rotate3d on***.

rotate3d off יוצא ממצב זה. ***rotate3d*** עובר ממצב למצב.

ב matlab- 5 ניתן לשנות את תכונות המבט בעזרת טיפול במיקום המצלמה, זוית הסיבוב שלה, זוית הפתיחה ונקודת מרכז התמונה. ההגדרות תתבצענה באמצעות הפקודה

set(gca,'property',N).

הפקודה ***gca*** משייכת את ההגדרה ל axes הנוכחי.

תכונות חשובות:

הפקודה ***set(gca,'cameraposition',[x,y,z])*** תמקם את המצלמה בנקודה x,y,z .

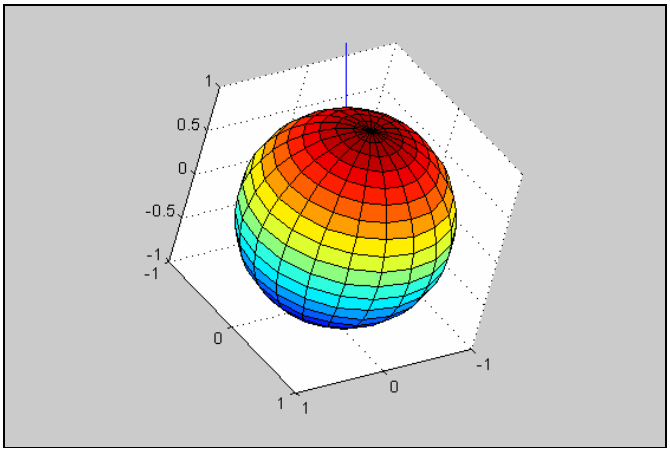
הפקודה ***set(gca,'cameratarget',[x,y,z])*** תכוון את המצלמה לנקודה x,y,z, כך שהנקודה תופיעה במרכז התמונה. ביחד עם נקודת מיקום המצלמה מגדירות 2 הנקודות את כיוון המבט.

הפקודה ***set(gca,'cameraupvector',[nx,ny,nz])*** תסובב את המצלמה סביב וקטור כיוון המבט, כך שהיטלו של הווקטור [nx,ny,nz] על גבי המישור המאונך לווקטור כיוון המבט, יצביע מעלה .

הפקודה ***set(gca,'cameraviewangle',deg)*** תגדיר את זווית הפתיחה deg (במעלות) של המצלמה.
ככל שהזוית גדלה שטח הכיסוי גדל כך שהאובייקט קטן וההפך.

דוגמה

```
[x,y,z]=sphere(20);
theta=45;
theta=pi*theta/180;
cp=[2 2 2];
ct=[0 0 0];
n=ct-cp;
b=cross([0 0 1],n);
zt=cross(n,b);
vup=zt.*cos(theta)+b.*sin(theta);%create cameravectorup with 45 degree rotation
vup=2*[vup]/norm(vup);
surf(x,y,z);%draws the sphere ;
axis('equal');
set(gca,'cameraposition',cp);
set(gca,'cameratarget',ct);
set(gca,'cameraupvector',vup);
set(gca,'cameraviewangle',50)
hold on;
plot3([0 vup(1)],[0 vup(2)],[0 vup(3)],'b');
»camer
```



עיבוד נתונים וסטטיסטיקה

תוכנת MATLAB מאפשרת ביצוע פעולות עיבוד נתונים וסטטיסטיקה, כאשר המידע נמצא במבנה וקטורים או בעמודות של מטריצות. הפונקציות הבסיסיות מוגדרות במחיצה datafun. פונקציות עיבוד נתונים אחרות, ניתן למצוא ב-toolbox (כגון עיבוד אותות, התאמת עקומות, סטטיסטיקה מתקדמת ועוד).

טעינת נתונים, לצורך עיבודם ב- MATLAB מתבצעת במבנה של מערכים. כל עמודה במערך מייצגת משתני מדידה שונים, וכל שורה במערך מייצגת נתונים של דגימה אחת.

דוגמה נתוני הטמפרטורה בשלוש ערים נמדדו אחת ליום במשך חודש אחד בן 31 ימים. לצורך עיבודם נכניס את הנתונים ל- MATLAB במבנה של מערך בעל - 3 עמודות ו- 31 שורות.

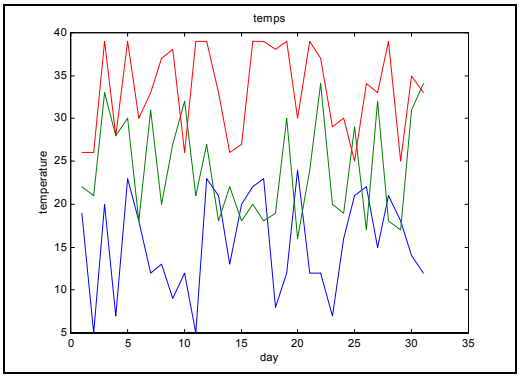
```
» temps =[19      22    26
          5       21    26
          20      33    39
          7       28    28
          23      30    39
          18      18   30
          12      31   33
          13      20   37
          9       27   38
          12      32    26
          5       21    39
          23      27    39
          21      18    33
          13      22    26
          20      18    27
          22      20    39
          23      18    39
          8       19    38
          12      30    39
          24      16    30
          12      24    39
          12      34    37
          7       20    29
          16      19    30
          21      29    25
          22      17    34
          15      32    33
          21      18    39
          18      17    25
          14      31    35
          12      34    33];
```

להצגה גרפית של הנתונים:

```
» d=1:31;
```

```
» plot(d,temps)
» title('temps')
» xlabel('day')
» ylabel('temperature')
```

הפלט המתקבל:



בדרך כלל פקודה לעיבוד נתונים תבצע את פעולת העיבוד על כל ווקטור עמודה בנפרד, כך שעבור קלט שהינו מערך בעל n עמודות יתקבל פלט שהוא ווקטור שורה בעל n איברים.

mean(V) – חישוב ממוצע של וקטור V או של עמודות המטריצה V

לחישוב ממוצע הטמפ' בכל אחת מהערים (בכל עמודה במערך temps) נשתמש בפונקציה ***mean***:

```
» avg_temps=mean(temps)
avg_temps =
   15.4516    24.0645    33.2258
```

התקבל ווקטור שורה שאיבריו הם ממוצעי העמודות.

לחישוב ממוצע כללי נוכל להפעיל את הפונקציה mean על המטריצה temps, כשהיא מסודרת בצורה של ווקטור עמודה:

```
» avg_t=mean(temps(:))
avg_t =
   24.2473
```

התוצאה תהיה זהה לממוצע של ווקטור השורה avg_temps:

```
» avg_t=mean(avg_temps)
avg_t =
```

```
    24.2473
```

חישוב מינימום ומקסימום של מערך

[M,I]=max(X), [M,I]=min(X) – פקודה המחזירה את המקסימום / מינימום של מטריצה X.

ווקטור השורה M יכיל את האיבר המינימלי / מקסימלי של כל עמודה במערך X. ווקטור השורה I יכיל את האינדקסים של האיברים שנמצאו (מינימלי / מקסימלי), כך ש- I(j) הוא האינדקס של האיבר המינימלי/מקסימלי בעמודה j של המטריצה X.

דוגמה מציאת הטמפרטורה המינימלית במטריצה **temps**:

```
» [min_temps,I]=min(temps)
min_temps =
     5    16    25
I =
     2    20    25
```

דוגמה שתי דרכים אפשריות לחישוב המינימום/מקסימום הגלובלי של temps:

```
» [min_tot,J]=min(min_temps)   % first method
min_tot =
     5
J =
     1
» temps(I(J),J)
ans =
     5


» [min_tot,k]=min(temps(:))    % second method
min_tot =
     5
k =
     2
```

```
» temps(k)
ans =
     5
```

שימוש נוסף של פקודות ה- ***min/max*** הוא לנפות מתוך מערך איברים הגדולים או קטנים מערך מסוים או לחילופין לבנות מטריצה מאיברים מתאימים גדולים/קטנים ביותר משתי מטריצות.

דוגמה

```
» X=[1 2 3;4 5 6];
» Y=[0 3 0;4 6 1];
» min(X,3)
ans =
     1     2     3
     3     3     3
» [X]=max(X,n)
ans =
     1     3     3
     4     6     6
```

rand(m,n) – קבלת מערך אקראי במימד m x n.

R יהיה מערך אקראי בגודל $m \times n$, כך שאיבריו הם מספרים בין 0 ל- 1 ומפולגים בצורה אחידה.

randn(m,n) – קבלת מערך אקראי שאיבריו מפולגים התפלגות נורמלית

Rn יהיה מערך אקראי בגודל $M \times N$ כך שאיבריו מפולגים גאוסית (התפלגות נורמלית) עם ממוצע 0 וסטיית תקן 1.

הטבלה הבאה מרכזת את פונקציות עיבוד הנתונים העיקריות ב-MATLAB:

הפונקציה	תאור
cumprod	מכפלה מצטברת של מטריצה/וקטור
cumsum	סכום מצטבר של מטריצה/וקטור
cumtrapz	אינטגרציה נומרית בשיטת הטרפז – מצטברת
diff	פונקצית הפרש וקירוב נגזרת
max	מציאת איבר מקסימלי
mean	חישוב הממוצע
median	מציאת ערך אמצעי
min	מציאת איבר מינימלי
prod	סכום איברים
sort	מיון מטריצה/וקטור בסדר עולה
sortrows	מיון שורות בסדר עולה
std	חישוב סטיית תקן
sum	חישוב סכום איברים בוקטור
trapz	אינטגרציה נומרית בשיטת הטרפז

פולינומים

פולינום מיוצג ב- MATLAB על-ידי וקטור שורה המכיל את מקדמיו. לדוגמא, יצוג פולינום ממעלה n - $(\mathbf{a_1X^n + a_2X^{n-1} + a_3X^{n-2} \ldots\ldots a_nX + a_{n+1}})$ בווקטור **P** יהיה

$\mathbf{P}=[a_1\ a_2\ a_3\ \ldots\ldots\ldots\ldots a_n\ a_{n+1}]$.

וקטור המקדמים יכלול גם מקדמים השווים אפס.

R=roots(P) - חישוב שורשים של פולינום

דוגמה שורשי הפולינום המיוצג בווקטור השורה **P** יתקבלו בווקטור שורה **R**:

```
» P
P =
     1   -12     0    25   116
» r=roots(P)
r =
  11.7473
   2.7028
  -1.2251 + 1.4672i
  -1.2251 - 1.4672i
```

פעולות בפולינומים

הפעולה מתבצעת לפי כללי חיבור וקטורים. כדי לחבר שני פולינומים יש לייצג אותם בשני וקטורים בעלי ממד זהה (אם יש צורך, אפשר להוסיף אפסים משמאל לפולינום מהמעלה היותר נמוכה).

דוגמה חיבור הפולינומים $\left(\mathbf{X}^3+2\mathbf{X}^2+3\mathbf{X}+4\right)+\left(\mathbf{X}^2+1\right)$:

```
» [1 2 3 4]+[0 1 0 1]
ans =
     1     3     3     5
```

C=conv(A,B) – כפל פולינומים

התוצאה בווקטור השורה C תהיה הייצוג הווקטורי של מכפלת הפולינומים A ו- B.

דוגמה

```
» A=[1 4 9 16 0 1];
» B=[1 2 3 4];
» C=conv(A,B)
 C=
  1     6    20    50    75    85    66     3     4
```

[C,R] = deconv(A,B) - חלוקת פולינומים

חלוקת פולינום **A** בפולינום **B** תיתן פולינום **C** ושארית **R** כך ש A=C*B+R.

דוגמה

```
» A=[6 8 19 -3 -3 -24];
» B=[3 4 5];
» [C,R]=deconv(A,B)
C =
     2     0     3    -5
R =
     0     0     0     0     2     1
```

dP=polyder(P) – נגזרת של פולינום

דוגמה חישוב הביטוי $\frac{\mathbf{d}}{\mathbf{dX}}\left(\mathbf{X}^3+2\mathbf{X}^2+3\mathbf{X}+4\right)=3\mathbf{X}^2+4\mathbf{X}+3$:

```
» polyder([1 2 3 4])
ans =
     3     4     3
```

Y=polyval(P,a) – חישוב ערך הפולינום בנקודה a

הפקודה מציבה את a בפולינום P ומחשבת את ערכו :

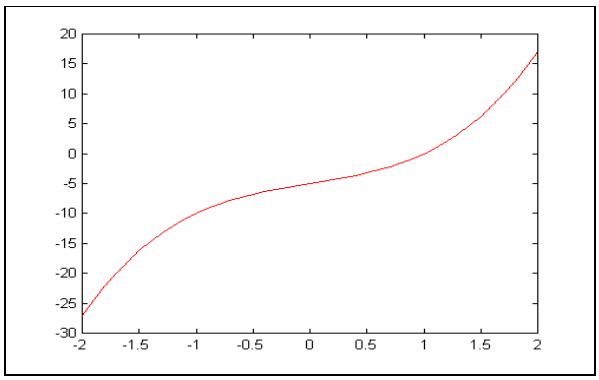
$$P = a_1 x^n + a_2 x^{n-1} + a_3 x^{n-2} \ldots\ldots a_n x + a_{n+1} \quad \Rightarrow \quad Y = a_1 a^n + a_2 a^{n-1} + a_3 a^{n-2} \ldots\ldots a_n a + a_{n+1}$$

אם a היא מטריצה בגודל $n \times m$ תתקבל מטריצה Y בגודל זהה, כך שכל איבר בה יהיה ערך הפולינום בנקודה המתאימה ב- a.

<u>דוגמה</u> שירטוט הפולינום $2x^3 + 3x - 5$ בתחום [-2,2]:

```
» a=[-2:0.1:2];
» P=[ 2 0 3 -5];
» Y=polyval(P,a);
» plot(a,Y)
```

הפלט המתקבל:



P=poly(R) – חישוב פולינום על-סמך שורשיו

לפולינום המתקבל P, יהיו השורשים המופיעים בווקטור השורה **R**.

<u>דוגמה</u>

```
» p=poly([9 8 1])
p =
     1   -18    89   -72
» R=roots(p)
R =
    9.0000
    8.0000
    1.0000
```

[R,P,K]=residue(B,A) - פירוק מנת פולינומים לגורמים

במקרים רבים קיים צורך בפירוק מנת פולינומים

$$\frac{a_1X^n + a_2X^{n-1} + \ldots + a_nX + a_{n+1}}{b_1X^m + b_2X^{m-1} + \ldots + b_mX + b_{m+1}}$$

לסכום מהצורה :

$$\frac{c_1}{X - r_1} + \frac{c_2}{X - r_2} + \ldots\ldots + \frac{c_m}{X - r_m} + k$$

פעולה זו נקראת פירוק לשברים חלקיים.

בפקודה הנ״ל, A הוא פולינום המכנה ו- B הוא פולינום המונה. בווקטור העמודה R מתקבלים מקדמי המונים לאחר הפירוק $(c_1,c_2..c_m)$, בווקטור העמודה P מתקבלים שורשי המכנים $(r_1,r_2...r_m)$ ובווקטור k מתקבל פולינום השארית (אם קיים כזה).

<u>דוגמה</u>

פירוק הביטוי :

$$\frac{10X + 20}{(X+1)(X+3)(X+4)} = \frac{-6.6667}{X+4} + \frac{5}{X+3} + \frac{1.6667}{X+1} + 0$$

```
» A=[10 20];
» B=poly([-1 -3 -4])
B =
     1     8    19    12
» [R,P,k]=residue(A,B)
R =
   -6.6667
    5.0000
    1.6667
P =
   -4.0000
   -3.0000
   -1.0000
k =
     []
```

את הפעולה ההפוכה, כלומר הפיכת הסכום למנת פולינומים, ניתן לחשב באמצעות אותה פונקציה, אך הפעם עם שלושה נתוני קלט ושני נתוני פלט - [B,A]=residue(R,P,K) :

דוגמה

```
» [A,B]=residue(R,P,k)
A =
    0.0000   10.0000   20.0000
B =
    1.0000    8.0000   19.0000   12.0000
```

Y=polider(A,B) – נגזרת של מכפלה

נקבל ב- **Y** נגזרת של מכפלת הפולינומים **A** ו- **B**.

[Q,D]=polider(A,B) – נגזרת של מנה

נקבל מונה **Q** ומכנה **D** כך ש- **Q/D** היא הנגזרת של **A/B**.

דוגמה

```
» A=[2 5 1 -3 0];
» B=[1 7 0 -4 6 5];
» y=polyder(A,B)
y =
    18   152   252   -24  -145   144   129   -26   -15

» [Q,D]=polyder(A,B)
Q =
    -2   -10   -38   -18    79   100    69    10   -15
D =
     1    14    49    -8   -44    94    86   -48    -4
60    25
```

אינטרפולציה

אינטרפולציה היא דרך להערכת ערכי פונקציה בין נקודות המדידה, על-ידי מציאת פונקציה f, שערכיה בנקודות המדידה שווים לערכים המדודים, כלומר:

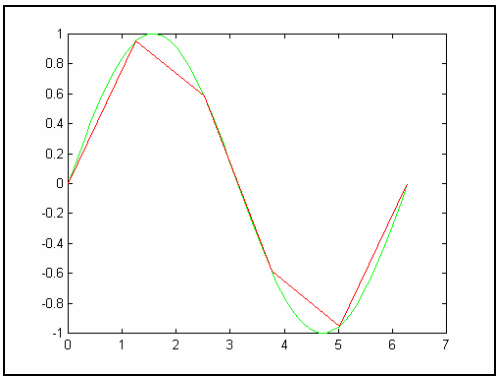
$$Y_i = f(X_i)$$

ברור אם כן, שעבור נקודות מדידה רבות יותר האינטרפולציה תהיה מדוייקת יותר. כך למשל פקודת ה- ***plot*** כברירת מחדל, מחברת בקו ישר בין נקודות המדידה. זוהי אינטרפולציה ליניארית. ככל שהתחום יכלול נקודות רבות יותר - הדיוק יהיה גבוה יותר.

דוגמה אינטרפולציה ליניארית באמצעות דגימת 6 נקודות (הקו המקוטע) ו 60 נקודות (הקו החלק):

```
» x1=linspace(0,2*pi,60);
» x2=linspace(0,2*pi,6);
» plot(x1,sin(x1),'g',x2,sin(x2),'r');
```

הפלט המתקבל:



אינטרפולציה חד מימדית

YI=interp1(X,Y,XI) – אינטרפולציה ליניארית

בדיוק כמו ב- ***plot*** שבדוגמה הנ"ל, באמצעות פקודה זו נקבל עבור **Y** - ווקטור ערכי המדידה בנקודות **X**, את ערכי ווקטור העמודה **YI**, שהינם ערכי הפונקציה בנקודות המצוינות בווקטור **XI**. איברי **XI** חייבים להיות בתחום איברי **X**. איברי **X** חייבים להיות מונוטוניים.

שיטות נוספות לאינטרפולציה

YI=interp1(X,Y,XI,'method') – הפעלת אינטרפולציה בשיטה מסויימת

בשיטה זו נבנית מערכת משוואות (implicit) על סמך האילוצים הבאים:

דרך שלוש הנקודות הראשונות יעבור פולינום מסדר 3. כל נקודה נוספת תשמש לחישוב פולינום העובר דרכה ודרך הנקודה שמשמאלה, כאשר נגזרתו הראשונה והשניה של פולינום זה בנקודה השמאלית תהיינה שוות לנגזרות הפולינום הקודם (משמאל) באותה נקודה. בכך מתקבלת רציפות גם בנגזרת השנייה בין כל שני פולינומים סמוכים.

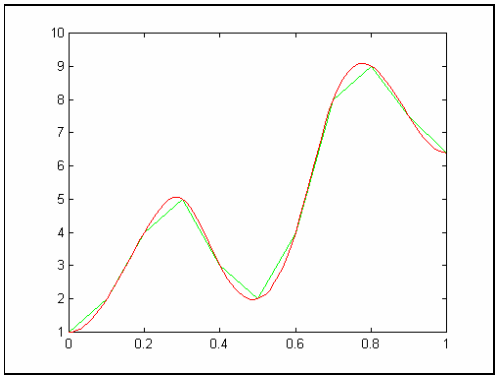
מאחר ומדובר בפולינום מסדר 3 נקראת השיטה CUBIC SPLINES .

שיטה נוספת, לפיה האילוץ הוא רק לגבי הנגזרת הראשונה נקראת CUBIC.

דוגמה

```
» x=[0:0.1:1];
» y=[1 2 4 5 3 2 4 8 9 7.5 6.4];
» x1=[0:0.01:1];
» y1=interp1(x,y,x1,'spline');
» plot(x,y,'og',x1,y1,'r')
» plot(x,y,'g',x1,y1,'r')
```

הפלט המתקבל:



קירובים

לכל אוסף נתונים ממדידה או מחישובים בדידים, אפשר להתאים מספר רב של פונקציות, שעקומותיהן מתנהגות באופן דומה לנתוני המדידה. עבור כל פונקציה כזאת אפשר לחשב את המקדמים עבורם השגיאה תהיה מינימלית. בדרך כלל השגיאה הכללית מוגדרת כסכום הריבועי של ההפרשים בין ערכי הפונקציה המקורבת בנקודות המדידה לבין הערכים המדודים בנקודות אלו . כלומר :

$$e=\sum_{i=1}^{n}\left(f(x_i)-y_i\right)^2$$

על מנת למצוא את מקדמי הפונקציה כך שהשגיאה תהיה מינימלית, יש לגזור את e לפי כל אחד מהמקדמים ולהשוות לאפס . שיטה זו נקראת ריבועים פחותים (least squares) .

p=polyfit(x,y,n) – התאמת פולינום

פונקציה זו מבצעת least squares לחישוב מקדמי פולינום מסדר n. הווקטור **x** מהווה את ערכי המשתנה הבלתי תלוי ,ו- **y** את ערכי המשתנה התלוי.

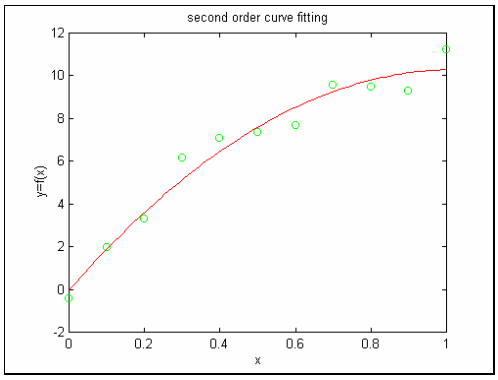
בווקטור השורה p מתקבלים מקדמי הפולינום.

דוגמה נניח שבמדידה מסויימת התקבלו התוצאות הבאות :

```
» x=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1] ;
» y=[-0.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2];
```

תוצאות המדידה וכן גרף הפולינום מסדר 2 עבורו השגיאה מינימלית :

```
» n=2;
» p=polyfit(x,y,n);
» xi=linspace(min(x),max(x),100);
» yi=polyval(p,xi);
» plot(x,y,'og',xi,yi,'r')
» xlabel('x')
» ylabel('y=f(x)')
» title('second order curve fitting')
```

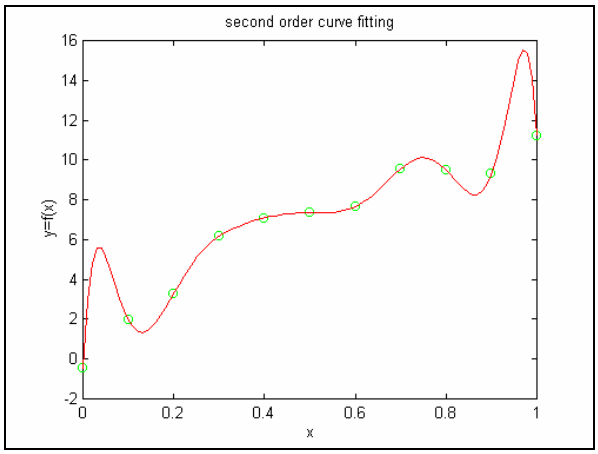


סדר הפולינום האופטימלי אינו בהכרח הגבוה ביותר והוא תלוי בתוצאות המדידה. כמו כן, עבור סדר n דרושים לפחות n+1 נקודות מדידה (כמס' מקדמי הפולינום), במקרה זה יתקבל פולינום שעובר דרך n+1 הנקודות - מצב שאינו בהכרח אופטימלי.

דוגמה עבור נתוני המדידה הנ"ל :

```
» p=polyfit(x,y,max(size(x))-1);
» xi=linspace(min(x),max(x),100);
» yi=polyval(p,xi);
» plot(x,y,'og',xi,yi,'r')
» xlabel('x')
» ylabel('y=f(x)')
» title('second order curve fitting')
```

הפלט המתקבל :



כמו שניתן לראות הקירוב אמנם עובר בכל הנקודות אך הוא גרוע לצורך אינטרפולציה, אינטגרציה או נגזרת.

אנליזה נומרית

אנליזה נומרית הינה תחום מתמטי בה מחפשים פתרונות לבעיות מתמטיות, תוך שימוש בקירובים נומריים והתחשבות בשגיאות ובתנאי קצה.

MATLAB מספק כלים נומריים לפעולות הבאות:

- שרטוט פונקציה
- חישוב נקודות קיצון
- מציאת אפסים
- אינטגרציה
- נגזרות
- פתרון משוואות דיפרנציאליות

שרטוט פונקציה

fplot – שרטוט גרף של פונקציה

הפקודה ***fplot*** כוללת אלגוריתם המבטיח שהשרטוט לא יחסיר פרטים כתוצאה מתהליך הדגימה. את הפונקציה ניתן לשמור כמשתנה מחרוזת או כ- M-file function.

דוגמה

שירטוט הפונקציה:

$$f(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04} - 6$$

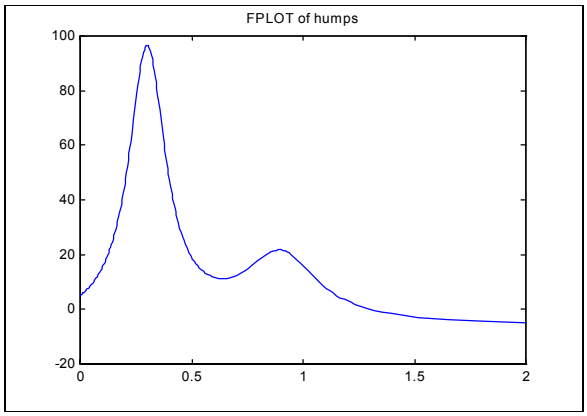
, בקטע $0 \le x \le 2$. התוכנית humps.m:

```
function y = humps(x)
   y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
```

הרצת התוכנית:

```
» fplot('humps',[0 2],'b')
» title('FPLOT of humps')
```

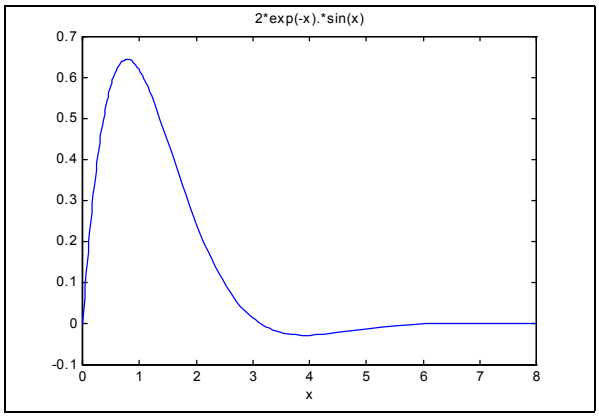
הפלט המתקבל:



דוגמה

שרטוט הפונקציה $f(x)=2\cdot e^{-x}\cdot \sin(x)$ בקטע $0 \le x \le 8$. בדוגמה זו, הפונקציה מוגדרת כמשתנה מחרוזת.

```
» f='2*exp(-x).*sin(x)';
» fplot(f,[0 8],'b')
» title(f),xlabel('x')
```



מציאת נקודות קיצון (מינימום מקסימום)

מציאת נקודות קיצון מחושבת אנליטית על-ידי השוואת הנגזרת לאפס. ב- MATLAB שתי פונקציות המאפשרות חישוב נומרי של נקודת מינימום.

fmin - מציאת מינימום מקומי של פונקציה בעלת משתנה אחד

fmins - מציאת מינימום של פונקציה מרובת משתנים

הכוונה במושג מינימום מקומי היא למינימום בתוך קטע מסוים הנקבע על-ידי המשתמש. כאשר מעוניינים למצוא מקסימום, מחשבים מינימום של -f(x) .

את הפונקציה אפשר לשמור כמשתנה מחרוזת או כ- M-file function.

דוגמה התוכנית הבאה משמשת לחישוב נקודות מינימום ומקסימום של הפונקציה $f(x)=2\cdot e^{-x}\cdot \sin(x)$ שהוצגה בדוגמא האחרונה. נכתוב תוכנית (exp_fmin.m):

```
% exp_fmin.m

fn='2*exp(-x).*sin(x)'; % define function for min
xmin=fmin(fn,2,5)       % search over rang 2<x<5
x=xmin;            % need x since fn has x as its variable
ymin=eval(fn)      % evaluate at xmin

fx='-2*exp(-x).*sin(x)';      % defin for max: note minus sign
xmax=fmin(fx,0,3)       % search over range 0<x<3
x=xmax;
ymax=eval(fn)                 % evaluate at xmax
```

הרצת התוכנית:

```
» exp_fmin
xmin =
    3.9270
ymin =
   -0.0279
xmax =
    0.7854
ymax =
    0.6448
```

פתרון אנליטי (מדויק) לבעיה זו:

$$Xmin=\frac{\pi}{4}\approx 0.785$$

$$Xmax=\frac{5\cdot\pi}{}\approx 3.93$$

השימוש בפקודה ***fmins*** מעט שונה. במקום לציין תחום, מציינים ווקטור ראשוני להתכנסות - Xo.

דוגמה מציאת מינימום של הפונקציה הדו-ממדית $z=x^2+y^2$ באזור הקרוב ל- (x,y) = (1,1) (תוכנית two_var.m):

```
function z=two_var(v);
  x=v(1);
  y=v(2);
```

```
  z=x.^2+y.^2;
```

הרצת התוכנית:

```
» v=[1 1];
» xymin=fmins('two_var',v)
xymin =

   1.0e-05 *

   -0.1148   -0.6231

» Zmin=two_var(xymin)     % verifing the result
Zmin =
  4.0138e-011
```

פתרון אנליטי: $(x,y) = (0,0)$

מציאת אפסים

fzero - מציאת אפסים של פונקציה בעלת משתנה יחיד

אפסים של פונקציה הם המקומות שבהם ערך הפונקציה הוא אפס. הפקודה ***fzero*** פועלת אך ורק על פונקציה המוגדרת כ- M-file function . בנוסף לציון שם הפונקציה יש לקבוע ערך ראשוני להתכנסות.

דוגמה מציאת אפס של הפונקציה humps, הקרוב ביותר לנקודה $x = 1.2$:

```
» xzero=fzero('humps',1.2)     % look for zero near 1.2
xzero =
    1.2995
» yzero=humps(xzero)           %  evaluate at xzero
yzero =
  3.5527e-015
```

אם מעונינים בנקודת חיתוך בין פונקציה וקבוע, $f(x) = c$, אפשר להגדיר פונקציה חדשה $g(x) = f(x) - c$ ולפתור ב- MATLAB ע"י מציאת אפסים של $g(x)$.

אינטגרציה

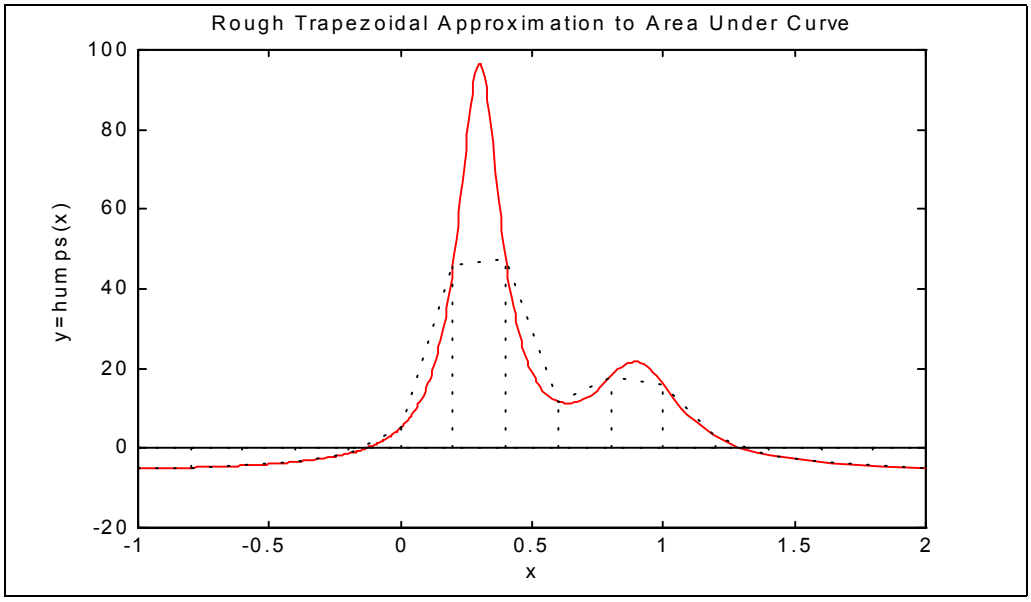
הפקודה ***trapz*** מחשבת שטח מתחת לפונקציה על-ידי סיכום שטחי הטרפזים הנוצרים מחיבור ליניארי של נקודות דגימה.

דוגמה

חישוב השטח מתחת לפונקציה humps בתחום $-1<x<2$. מרווח דגימה $\Delta x = 0.2$:

```
» x=0:0.2:2;
» y=humps(x);
» area=trapz(x,y)        % call trapz just like the plot command
area =
   26.4616
```

קירוב גס של האינטגרל:



ככל שנקטין את מרווח הדגימה תתקבל תוצאה קרובה יותר לשטח האמיתי.

קירוב יותר טוב נקבל כאשר נבחר $\Delta x = 0.02$

```
» x=-1:0.02:2;    % better approximation
» y=humps(x);
» area=trapz(x,y)
area =
   26.3449
```

פקודות אינטגרציה נוספות:

quad - אינטגרציה בשיטת **Simpson**

quad8 - אינטגרציה בשיטת **Newton Cotes**

שתי השיטות מדויקות יותר משיטת הטרפזים. כאשר משתמשים בפקודות אלה אין צורך לקבוע מרווחי דגימה, את הפונקציה צריך לשמור כ- M-file function.

דוגמה חישוב השטח מתחת לפונקציה humps בתחום $-1 < x < 2$:

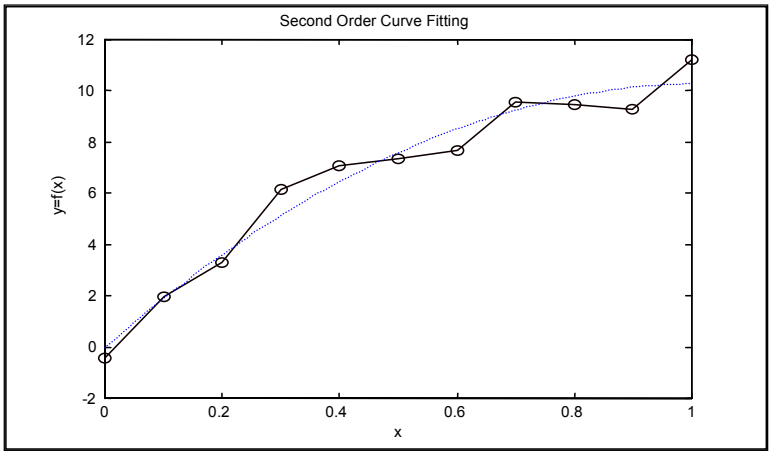
```
» area=quad('humps',-1,2)   %  find area between -1 and 2
area =
   26.3450

» area=quad8('humps',-1,2)   %  find area between -1 and 2
area =
   26.3450
```

נגזרות

polyder(p) - פקודה זו משמשת לגזירת הפולינום - p.

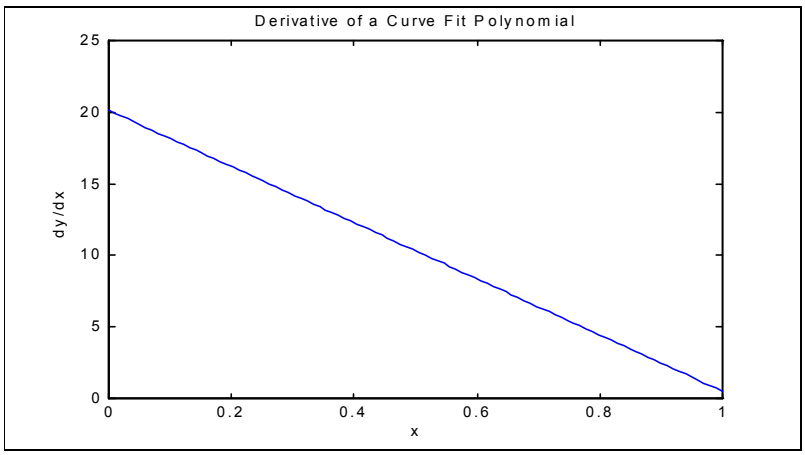
דוגמה **x** ו- **y** הם שני ווקטורים המכילים אוסף תוצאות של ניסוי. נתונה סידרה של נקודות במישור x ,y. מעוניינים לחשב בקרוב את הנגזרת- $\frac{dy}{dx}$.

```
» x=0:0.1:1;
» y=[-.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2]; % data
» p=polyfit(x,y,2)            %  find polynomial coefficients
p =
   -9.8108   20.1293   -0.0317

» xi=linspace(0,1,100);       % drawing the result polynom
» z=polyval(p,xi);            % evaluate polynomial
» plot(x,y,'o',x,y,xi,z,':')
» xlabel('x'),ylabel('y=f(x)'),title('Second Order Curve Fitting')
```



נגזור ב- MATLAB:

```
» pd=polyder(p)
pd =
  -19.6217   20.1293

» z=polyval(pd,xi);              %  evaluate derivative
» plot(xi,z)
» plot(xi,z,'b')
» xlabel('x'),ylabel('dy/dx'),title('Derivative of a Curve Fit Polynomial')
```



קרוב נגזרת באמצעות הפקודה diff

הגדרה של נגזרת -

$$\frac{dx}{dy} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{(x+h) - x}$$

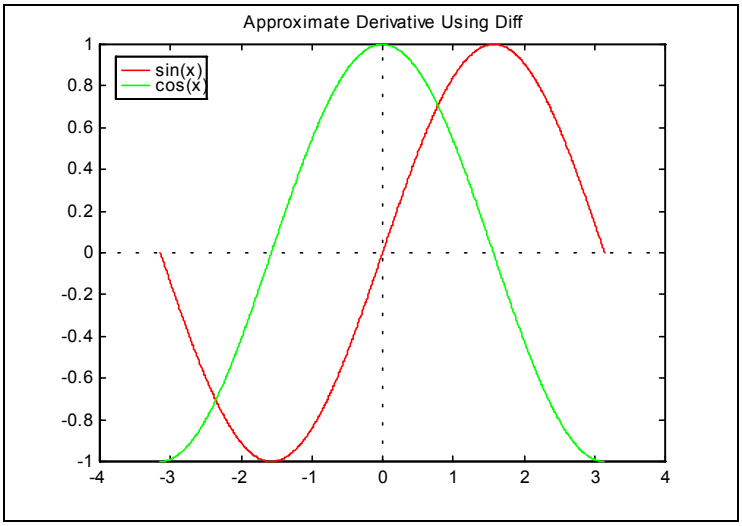
קרוב ($h > 0$) -

$$\frac{dx}{dy} \approx \frac{f(x+h) - f(x)}{(x+h) - x}$$

דוגמה

נגזרת של $\sin(x)$ (התוצאה היא קרוב של $\cos(x)$):

```
» x=linspace(-pi,pi,1000);
» y=sin(x);
» dy=diff(y)./diff(x);  % compute differences and use array division
» dx=x(1:length(x)-1);  % create new x axis since dy is shorter than y
» plot(x,y,dx,dy)
» title('Approximate Derivative Using Diff')
» legend('sin(x)','cos(x)')
```



השימוש בפקודה ***diff*** מומלץ אך ורק כאשר לא ניתן לגזור אנליטית. במקרה זה יש להשתמש במרווחי דגימה קטנים ככול האפשר. שימוש בפקודה ***diff*** אינו מומלץ כאשר מדובר בתוצאות של ניסוי (כמו בדוגמה הקודמת) .

מערכי תאים Cell arrays (ב - Matlab 5 בלבד)

מערך תאים הוא מערך אשר כל אלמנט בו הוא תא, היכול להכיל כל סוג של משתנה הקיים ב-MATLAB. לדוגמא, תא אחד בתוך מערך תאים יכול להכיל מטריצה, תא אחר מחרוזת ותא נוסף מספר מרוכב.

לדוגמה, מבנה אפשרי של מערך תאים:

cell(1,1) [3 4 2 9 7 6 8 5 1]	**cell(1,2)** ['Anne smith' ' 9 / 12 / 94']	**cell(1,3)** [0.25+3j 8 - 16j 34 + 5j 7 + 92j]
cell(2,1) [1.43 2.98 5.76]	**cell(2,2)** [3 4 7 6]	**cell(2,3)** ['text' [4 2 1 5] [4 2 7] [0.02+8j]

בניית מערך תאים אפשרית בשתי דרכים:

1. שימוש במשפטי השמה וסוגריים מסולסלים { }.

דוגמה בניית המערך המוצג למעלה:

```
» A(1,1)={[3 4 2;9 7 6;8 5 1]};
» A(1,2)={char('Anne smith','9/12/94')};
» A(1,3)={[0.25 8;34 7]+i*[3 -16;5 92]};
» A(2,1)={[1.43 2.98 5.67]};
» A(2,2)={[3 4;7 6]};
» A(2,3)={{['text'],[4 2 ;1 5];[4 2 7],[0.02+8i]}};
» A
A =
    [3x3 double]    [2x10 char  ]    [2x2 double]
```

```
    [1x3 double]    [2x2  double]    {2x2 cell  }
```

2. בנייה מוקדמת של מערך התאים A בגודל- m על- n באמצעות הפקודה – ***A=cell(m,n)*** ולאחר מכן השמת ערכים בתאים על-ידי ציון מיקומם בסוגריים מסולסלים. לדוגמה, מיקום וקטור בתא (i,j) - ***A{i,j}=[1 2 3]***.

הבנייה המוקדמת מקצרת את זמן הריצה של התוכנית אך אינה הכרחית .

הצגת תכולתו של מערך תאים -**A** אפשרית ע"י שימוש בפקודה - ***celldisp(A)***.

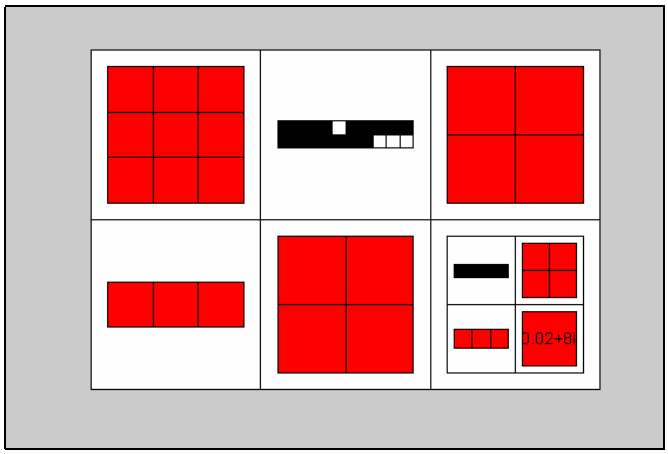
דוגמה

```
» celldisp(A)
A{1,1} =
     3     4     2
     9     7     6
     8     5     1
A{2,1} =
     1.4300    2.9800    5.6700
A{1,2} =
     Anne smith
9/12/94
A{2,2} =
     3     4
     7     6
A{1,3} =
     0.2500+ 3.0000i   8.0000-16.0000i
     34.0000+ 5.0000i   7.0000+92.0000i
A{2,3}{1,1} =
     text
A{2,3}{2,1} =
     4     2     7
A{2,3}{1,2} =
     4     2
```

```
     1     5
A{2,3}{2,2} =
     0.0200+ 8.0000i
```

הצגה גרפית של מערך התאים- A אפשרית ע"י שימוש בפקודה ***cellplot(A)***.

```
» cellplot(A)
```



שימוש באינדקסים

קריאה לתא מסוים תבוצע ע"י ציון מיקומו בסוגריים מסולסלים.

דוגמה

```
» name=A{1,2}
name =
     Anne smith
     9/12/94
```

קריאה לערך מסוים מתוך מערך, הנמצא בתוך תא מסוים, תתבצע ע"י ציון מיקום התא בסוגריים מסולסלים. ומימינם בהמשך, מיקום הערך במערך, בסוגרים רגילים.

דוגמה

```
» A{1,1}(2,2)
ans =
```

```
     7
» A{2,3}{1,2}(2,2)
ans =
     5
```

ניתן להשתמש בנקודתיים לצורך ציון שורה של אינדקסים, כך שכל איבר בשורה ייצג מיקום של תא (הדבר שקול לשורה המכילה את האינדקסים עם פסיקים ביניהם).

דוגמה

```
» b{1,1}=[1 2 3];
» b{1,2}=[2 9 4];
» b{2,1}=[5 6 7];
» b{2,2}=[8 0 2];
» b
b =
    [1x3 double]    [1x3 double]
    [1x3 double]    [1x3 double]
» b{1:2,2}
ans =
     2     9     4
ans =
     8     0     2
» cross(b{1:2,2})
ans =
    18    28   -72
```

תם ונשלם (זמני)

<u>רשימת פקודות שלא נלמדו או שנלמדו אך אינן מופיעות בקובץ הנ"ל</u>

repmat([1 3 5],2,4)

gcd(x,y)
lcm(x,y)
[t,n]=rat(x)

floops
floops(0)

fliplr(A)
flipud(A)
rot90(A)

<u>Strings</u>

הגדרת שמות משתנים כמחרוזת (str=string) הכוללת אותיות מספרים וסימנים מיוחדים: 'text'
abs(str)
length
num2str
mat2str
lower(str)
upper(str)
disp('str')
input('out',*in*)

mod(a,b)
factor(a)
primes(a)
isprime(a)

intersect(a,b)
union(a,b)
unique(a)

<u>cell matrix</u>
cell(m,n)
celldisp(B)
cellplot(B)

hist(x)
bar(x)
stairs
stem
pie

det(A)
rank(A)
inv(A)

(sparse matrices) - לא נלמד בכל הקבוצות
sparse(A)
sparse(m,n)
sparse(u,v,a)
find(x)
[u,v]=find(A)
[u,v,s]=find(A)
full(sparse_matrix)

randperm(8)

text(x,y,'text')
gtext('text')
legend('text1','text2', . . .)

axis([xmin xmax ymin ymax])
axis auto
axis square
axis tight
axis off
axis on

figure(n)

fplot('function',[xmin xmax])

zoom

[x,y]=ginput(n)
plotyy(x,y,x,z) – two different y scales

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

fill(x,y,'color')

3-D Graphics

[X,Y]=meshgrid(x,y)
Z=function(X,Y)
mesh(X,Y,Z)
surf(X,Y,Z)
contour(X,Y,Z)

plot3(x,y,z,'symbol')
plot3(x1,y1,z1,'symbol1', x2,y2,z2,'symbol2', ….)
zlabel('text')
text(x,y,z,'text')
axis([xmin xmax ymin ymax zmin zmax])

cylinder(r)
cylinder(r,n)

Polinomials

conv(a,b)
[q,r]=deconv(numerator,denumerator)

$$\frac{numerator(x)}{denumerator(x)} = q(x) + \frac{r(x)}{denumerator(x)}$$

$$numerator(x) = q(x)denumerator(x) + r(x)$$

Functions

fplot('function',[xmin xmax])
fzero('function',Xo)
fmim('function',xmin,xmax)
max of function – with: fmin('minus_function', xmin,xmax)

Interpolation

yi=interp1(x,y,xi,'method')
method - nearest
linear