ПРОГРАМИРАНЕ В MATLAB

dimitrova@tu-sofia.bg
pct.tu-sofia.bg/dd/pik3

УПРАВЛЯВАЩИ СТРУКТУРИ

Видове управляващи структури

- Общ вид
  - сложен оператор
  - започва с *ключова дума*
  - завършва с end
- Видове структури
  - последователност
  - алтернатива
  - цикъл

Последователност

- Определение - оператори и функции на MatLab, подредени в предварително определна последователност
- Видове оператори
  - присвояване
  - вход
  - изход

Вход

Въвеждане на данни в диалогов режим

input

въвеждане на числени данни

```
var = input('String')
```

въвеждане на текстови данни

```
var = input('String', 's')
```

Запитване за въвеждане

```
R = INPUT('How many apples')
T = input('Input the value of T: ')
```

```
R = INPUT('What is your name','s')
```

Запитване за въвеждане на текстов низ

Изход

disp(vars)

извеждане на стойности на променливи на печат

warning ('string')

извеждане на предупредително съобщение, след което програмата продължава

error('string')

извеждане на съобщение за грешка, след което програмата прекъсва

```
disp(eye(3,3))
```

```
1 0 0
0 1 0
0 0 1
```

```
x = input('Enter age of student');
if x < 0
  error('wrong age was entered, try again')
end
x = input('Enter age of student')
```

Формат на изхода

COMMAND	MEANING
format short	5 significant decimal digits
format long	15 significant digits
format short e	scientific notation with 5 significant digits
format long e	scientific notation with 15 significant digits
format hex	hexadecimal
format +	+ printed if value is positive, - if negative; space is skipped if value is zero

Вход и изход от форматиран файл

fopen()
fgetl()
fscanf()
fprintf()
fclose()

fid=fopen('file name', 'a')

fprinf(fid, 'format', vars)

fclose(fid)

format:

\n – премини на нов ред

\f – премини на нова страница

%символ – форматни спецификации за извеждане на данни, както в C

Алтернатива

- Оператори, които използват логически условия, за да изберат едно от няколко алтернативни разклонения за продължение на програмата
- Условен преход
  - if … end
  - if … else … end
  - if … elseif … … elseif … … else … end
  - switch …

```
    case
        …
    case
        …
    otherwise
        …
end
```

Синтаксис

```
if логическо условие
      оператори;
end
```

```
if x<0
      flag=-1;
end
```

Примери

Ако условието е изпълнено, операторите се изпълняват, в противен случай се пропускат . В двата случая програмата продължава след end.

```
if логическо условие, оператор; оператор; …; end
```

```
if mem>EPS, sum=sum+mem; nom=-nom; denom=denom+2; end
```

Синтаксис

```
if логическо условие
        оператори 1
else
        оператори 2
end
```

```
if x<0
    flag=-1;
else
    flag=1;
end
```

Ако условието е изпълнено, се изпълняват оператори 1, в противен случай се изпълняват оператори 2 . В двата случая програмата продължава след end.

Многозначно разклонение

```
if логическо условие 1
        оператори 1
elseif логическо условие 2
        оператори 2
…
else
        оператори n
end
```

```
if x<0
    flag=-1;
elseif x>0
    flag=+1;
else flag=0;
end
```

Условията се проверяват последователно, до намиране на съответствие. При първото условие, което се окаже изпълнено, се изпълняват съответстващите му операторите. Ако нито едно условие не е изпълнено, се изпълняват операторите n . Във всички случаи, програмата продължава след end.

Многозначно разклонение

Избира се една от множеството алтернативи, в зависимост от резултата от сравнение на ключов израз switch_израз с алтернативни изрази case_израз.

Ключовият израз приема само целочислени стойности или име на низ.

```
switch switch_израз
    case case_израз,
            оператори
    case {case_израз,...,case_израз}
            оператори
            ...
    otherwise
            оператори
end
```

```
switch  op
    case '+'  res=a+b;
    case '-'   res=a-b;
    case '*'  res=a*b;
    case '/'            res=a/b;
    otherwise
            error ('Друга операция')
end
```

Пример

3-битов А/Д конвертор

Преобразува аналоговия сигнал x в цифров сигнал y съгласно уравнението:

$$y = \begin{cases} 0, & x < -2.5 \\ 1, & -2.5 \le x < -1.5 \\ 2, & -1.5 \le x < -0.5 \\ 3, & -0.5 \le x < 0.5 \\ 4, & 0.5 \le x < 1.5 \\ 5, & 1.5 \le x < 2.5 \\ 6, & 2.5 \le x < 3.5 \\ 7, & x \ge 3.5 \end{cases}$$

```
if x<-2.5 y=0;
elseif x<-1.5 y=1;
elseif x<-0.5 y=2;
elseif x<0.5 y=3;
elseif x<1.5 y=4;
elseif x<2.5 y=5;
elseif x<3.5 y=6;
else  y=7;
end
```

Логически изрази

Резултатът от изчисляването им е True (1) или False (0).

Логически оператори

LOGICAL OPERATOR SYMBOL	MEANING
&	and
!	or
~	not

Оператори за сравнение

RELATIONAL OPERATOR	MEANING
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	equal
~=	not equal

Пример

```
a = [1 2 3 3 3 6];
b = [1 2 3 4 5 6];
a == b
ans =
     1 1 1 0 0 1
```

Цикъл

- Повторение на определена част от програмата
- Видове
  - с фиксиран брой на повторенията

    for променлива = начало:стъпка:край

      операции

    end
  - броят на повторенията зависи от зададено условие

    while израз

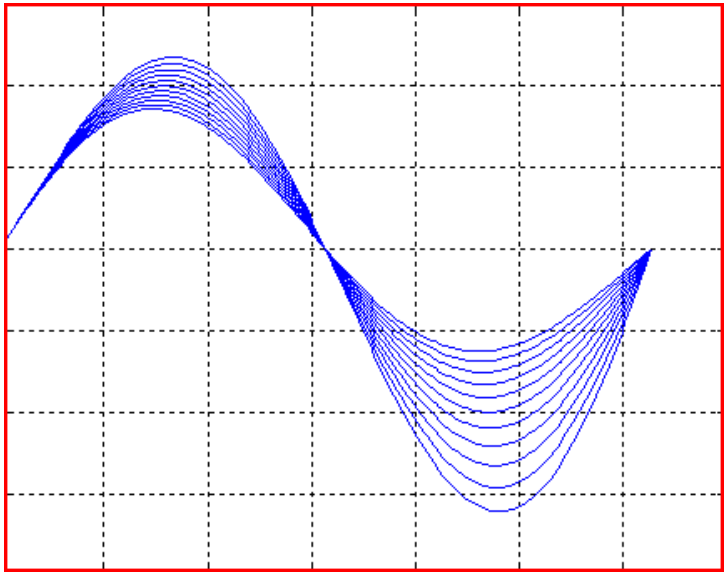
      операции

    end

break – прекратява изпълнението на for или while

Примери

```
x=0:pi/100:2*pi;
for a=-0.1:0.02:0.1
      y=exp(-a*x).*sin(x);
      plot(x,y), hold on
      grid on
end
```



```
x = 0.1;
dx = pi/5;
while x<=pi
  disp([x,sin(x), sqrt(x), exp(x)])
  x = x + dx;
end
```

0.1000	0.0998	0.3162	1.1052
0.7283	0.6656	0.8534	2.0716
1.3566	0.9772	1.1647	3.8831
1.9850	0.9155	1.4089	7.2787
2.6133	0.5041	1.6166	13.6436

Примери

```
n = 10; % number of
rows
m = 20; % number of
columns
for i = 1:n
   for j = 1:m
         b(i,j) = 1;
   end
end
```

```
sum = 0;
for i = 1:100
         sum = sum + i^2;
end
```

Обработка на изключения

Дефинира се ситуация операции 1, при която може да се получи грешка и се планират операции 2 за обработването й.

```
try
        операции 1
catch
        операции 2
end
```

```
try
   A=load('my.dat')
   plot(A)
catch
  disp('Няма такъв файл')
end
```

Други оператори за управление

break

прекратява изпълнението на for или while

continue

прекратява изпълнението на текущата итерация на на for или while

pause

pause(n)

задържа изпълнението на програмата до натискането на клавиш или n секунди

keyboard

K>>return

временно прекратява програмата и предава управлението към клавиатурата до появата на return

Примери

```
function [A] = area(a,b,c)
% Compute the area of a triangle whose sides have length a, b and c
% Inputs: a,b,c
% Output: A
% Usage: Area=area(2,3,4);
if (a+b>c && a+c>b && b+c>a)
   s = (a+b+c)/2;
   A = sqrt(s*(s-a)*(s-b)*(s-c));
else
   A=['Not a triangle'];
end
```

```
XX=area(3,4,5)
XX =
     6

>> YY=area(1,1,2)
YY =
     Not a triangle
```

Програмиране на функции

Скрипт и функция

- скрипт
  - ASCII файл, съдържащ блок от команди на MatLab
  - FILE/SAVE AS .m-file
  - няма заглавен ред
  - всички работни променливи се съхраняват в общо пространство Workspace
  - стартират се чрез името на файла, без аргументи
- функция
  - програмен модул с входни аргументи и изходни параметри

    y=f(x)
  - има заглавен ред и ключова дума

    function y = fname(x)

    function [y1, y2, …, ym] = fname(x1, x2, … xn)

Скрипт

```
%Програма distance
A=input('Координати на A:');
K=input('Координати на B:');
d=sqrt((A(1)-B(1))^2 +(A(2)-B(2))^2  );
disp(['Разстоянието между P1 и P2 е ',
num2str(d)]);
```

Функция

```
function d = dist(A,B)
% програма distance
d=sqrt((A(1)-B(1))^2 +(A(2)-B(2))^2  );
.
.
.
>>dist([3 2 1], [2,4,6])
```

Операции с полиноми

- В MatLab

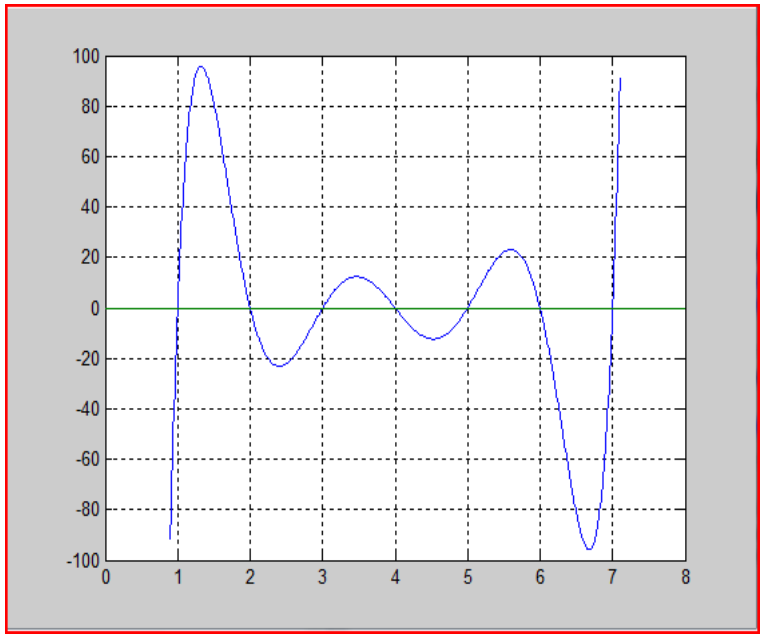
$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1$$

- вектор с елементи коефициентите на полинома
- функции за изчисление и анализ на полинома

```
r = roots(p)      % намира корените на полинома
p = poly(r)       % намира полином по зададени корени
```

Пример

```
% Polynom 1
r = [1 2 3 4 5 6 7];    % vector r with the roots
p = poly(r);            %  generation of polynom
                           with r roots
x = 0.9:0.01:7.1;       % x axis
y = polyval(p,x);       % the values of the
                           polynom for every x
plot(x, y, [0,8], [0,0])% graph
grid on
```



Геометричен анализ

delaunay - Delaunay triangulation.

delaunay3 - 3-D Delaunay tessellation.

delaunayn - N-D Delaunay tessellation.

dsearch - Search Delaunay triangulation for nearest point.

dsearchn - Search N-D Delaunay tessellation for nearest point.

tsearch - Closest triangle search.

searchn - N-D closest triangle search.

convhull - Convex hull.

convhulln - N-D convex hull.

voronoi - Voronoi diagram.

voronoin - N-D Voronoi diagram.

inpolygon - True for points inside polygonal region.

rectint - Rectangle intersection area.

polyarea - Area of polygon.

Функции за полиноми

roots - Find polynomial roots.

poly - Convert roots to polynomial.

polyval - Evaluate polynomial.

polyvalm - Evaluate polynomial with matrix argument.

residue - Partial-fraction expansion (residues).

polyfit - Fit polynomial to data.

polyder - Differentiate polynomial.

polyint - Integrate polynomial analytically.

conv - Multiply polynomials.

deconv - Divide polynomials.

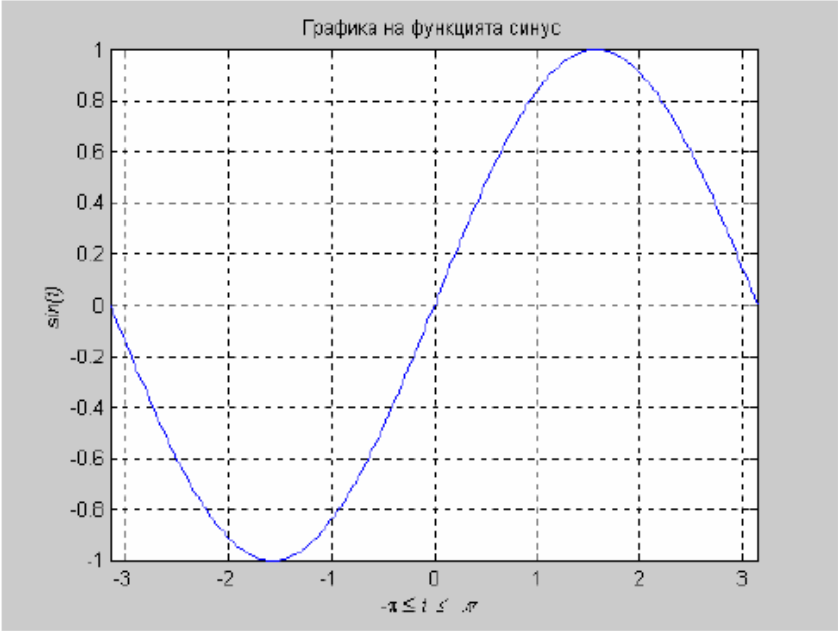
Функции за интерполация

- pchip - Piecewise cubic Hermite interpolating polynomial.
- interp1 - 1-D interpolation (table lookup).
- interp1q - Quick 1-D linear interpolation.
- interpft - 1-D interpolation using FFT method.
- interp2 - 2-D interpolation (table lookup).
- interp3 - 3-D interpolation (table lookup).
- interpn - N-D interpolation (table lookup).
- griddata - Data gridding and surface fitting.
- griddata3 - Data gridding and hyper-surface fitting for 3-dimensional data.
- griddatan - Data gridding and hyper-surfacefitting (dimension >= 2).

Числено решаване на уравнения

- Нули на функция
  - X=fzero(FUN,X0) %търси корените на FUN около X0

```
X=fzero('sin(x)',3)          % корен на sin(x) около 3
X =
   3.1416
X=fzero('sin(x)',[-1,1])     % корен на sin(x) в интервала ) [-1,1]
X =
    0
```



Решаване на системи линейни уравнения

Представяне на системата

$$Ax = b$$

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots a_{2,n}x_n &= b_2 \\ &\vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots a_{n,n}x_n &= b_n \end{aligned}$$

За квадратна матрица, където броят на уравненията е равен на броя на неизвестните:

$$\mathbf{x} = A^{-1}\mathbf{b}$$

Друг метод:

x = A \ b

по-ефективен, напр. при голям брой неизвестни

Пример

A = [2 -1 0; 1 -2 1; 0 -1 2]

b = [1; 0; 1]

$$A = \begin{bmatrix} 2 & -1 & 0 \\ 1 & -2 & 1 \\ 0 & -1 & 2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

A =

2	-1	0
1	-2	1
0	-1	2

b =

1

0

1

x = inv(A)*b

x =

1.0000

1.0000

1.0000

x=A\b

x =

1.0000

1.0000

1.0000

Минимум и максимум на функция

- Минимум

```
min=fminbnd('sin(x)',-2,-1)
                % минимум на sin(x) в интервала -2<x<-1
min =
  -1.5708
```

- Максимум

```
max(f(x))=min(-f(x))          % няма специална функция
max=fminbnd('-sin(x)',1,2)
                % максимум на sin(x) в интервала 1<x<2
max =
   1.5708
```

Диференциране

diff(f) Връща $\frac{df}{dx}$, x е подразбиращата се символна променлива

diff(f,n) Връща $\frac{d^n f}{dx^n}$, x е подразбиращата се символна променлива

diff(f,x,n) Връща $\frac{d^n f}{dx^n}$, x е явно зададена символна променлива

```
syms x y;       % дефинира реални символни променливи x и y
diff(x^y)       % първа производна спрямо x  или
diff(x^y,x)     % първа производна спрямо x
 ans =
     x^y*y/x
diff(sin(y*x),x,3) % трета производна спрямо x
 ans =
    -cos(y*x)*y^3
```

Интегриране

int(f) Връща $I = \int f(x)dx$, x е подразбиращата се символна променлива

int(f,x) Връща $I = \int f(x)dx$, x е явно зададена символна променлива

int(f,a,b) Връща $I = \int_a^b f(x)dx$, x е подразбиращата се символна променлива

int(f,x,a,b) Връща $I = \int_a^b f(x)dx$, x е явно зададена символна променлива

```
I=int(x^2,x)                    % неопределен интеграл
 I =
    1/3*x^3
I=int((x^2-2)/(x^3-1),x,2,5)    % определен интеграл
 I =
    -2/3*log(7)-2/3*3^(1/2)*atan(5/3*3^(1/2))-
      2/3*log(2)+2/3*log(31)+2/3*3^(1/2)*atan(11/3*3^(1/2))
```

Програмиране на анимация

1. Предварителна подготовка и запис на кадрите (movein, getframe, movie)

   M=movein(nfr) дефинира фрейм-матрица M със стълбове за всеки кадър на движещи се обекти от графиката

   отделните кадри се генерират с plot() и се записват като стълбове на фрейм-матрицата с M(:,i)=getframe,

   проиграват се зададен брой пъти с movie(M,к)

```
% Animation 1
nfr=60;                 % frames
fi=linspace(0,2*pi,nfr);   % angle of circulation
xA=cos(fi); yA=sin(fi);    % A coordinates
M=moviein(nfr);         % Frame matrix
for i=1:nfr
      plot([0,xA(i)], [0,yA(i)], 'k')
      axis([-1 1 -1 1]);  %[xmin xmax ymin ymax]
      axis('off')       % remove axis
      M(:,i) = getframe;
end
movie(M,3);             % run 3 times
close, clear
```

Програмиране на анимация

Всеки кадър се изчислява и изобразява в реално време със set и drawnow, докато старият се изтрива

```
% Animation 2
n=360;                          % positions
fi=linspace(0,2*pi,n);          % angle of circulation
xA=cos(fi); yA=sin(fi);         % A coordinates
axis('equal'); axis('off');
axis([-1 1 -1 1]);              %[xmin xmax ymin ymax]
point = line('Color', 'black', 'EraseMode', 'xor', ...
   'LineStyle', '-', 'XData', [], 'YData', []);
hold on
for j=1:3                       %3 cicles
   for k=1:n
      set(point, 'XData',[0,xA(k)], 'YData', [0,yA(k)])
      pause(0.01)
      drawnow
   end
end
close, clear
```