CURS 4

Structura programelor MATLAB

MATLAB-ul lucrează fie în modul linie de comandă, situaţie în care fiecare linie este prelucrată imediat şi rezultatele sunt afişate, fie cu programe conţinute în fişiere. Aceste două moduri formează împreună un „mediu" de programare.

Fişierele ce conţin instrucţiuni MATLAB se numesc fisiere-M (deoarece au extensia „.m") şi sunt programe MATLAB. Un fişier-M constă dintr-o succesiune de instrucţiuni MATLAB, cu posibilitatea apelării altor fisiere-M precum şi a apelării recursive.

Un program MATLAB poate fi scris sub forma fişierelor „script" sau a fişierelor „function". Ambele tipuri de fişiere sunt scrise în format ASCII, iar algoritmul care a fost implementat poate fi urmărit cu foarte multă uşurinţă, dacă se cunosc convenţiile şi sintaxa MATLAB. Aceste tipuri de fişiere, obligatoriu cu extensia *.m, permit crearea unor funcţii noi care le pot completa pe cele deja existente. Prin această facilitate, MATLAB-ul poate fi extins la aplicaţii specifice utilizatorului, care are posibilitatea să scrie noi proceduri.

Fişierele script

Un fişier „script" este un fişier extern care conţine o secvenţă de comenzi MATLAB. Prin apelarea numelui fişierului, se execută secvenţa MATLAB conţinută în acesta. După execuţia completă a unui fişier script, variabilele cu care acesta a operat rămân în zona de memorie a aplicaţiei (workspace). Aceste fişiere nu permit integrarea în programe mari, realizate pe principiul modularizării. Fişierele script sunt folosite pentru rezolvarea unor probleme care cer comenzi succesive atât de lungi, încât ar putea deveni greoaie pentru lucrul în mod interactiv, adică în modul linie de comandă.

Fişierele funcţie

Dacă prima linie a fisierului-M conţine cuvântul „function", fişierul respectiv este declarat ca fişier funcţie. O funcţie diferă de un ‚script" prin faptul că poate lucra cu argumente. Variabilele definite şi manipulate în interiorul fişierului funcţie sunt localizate la nivelul acesteia. Prin urmare, la terminarea execuţiei unei funcţii, în memoria calculatorului nu rămân decât variabilele de ieşire ale acesteia.

Fişierele funcţie sunt utilizate pentru extinderea MATLAB-ului, adică pentru crearea unor noi funcţii MATLAB. Forma generală a primei linii a unui fişier funcţie este:

function [param_ieşire]= nume_funcţie(param_intrare)

unde:

function - este cuvânt cheie care declară fişierul ca fişier funcţie (obligatoriu);

nume_funcţie - numele funcţiei, adică numele sub care se salvează fişierul, fără extensie. Nu poate fi identic cu cel al unui fisier-M preexistent.

param_ieşire - parametri de ieşire trebuie separaţi cu virgulă şi cuprinşi între paranteze drepte. Dacă funcţia nu are parametri de ieşire, parantezele drepte şi semnul egal nu mai au sens.

param_intrare - parametri de intrare trebuie separaţi cu virgulă şi cuprinşi între paranteze rotunde. Dacă funcţia nu are parametri de intrare, parantezele rotunde nu mai au sens.

Aceste fişiere pot fi adăugate ca funcţii noi în structura MATLAB. Comenzile şi funcţiile care sunt utilizate de noua funcţie sunt înregistrate într-un fişier cu extensia .m. Prima linie a fişierului trebuie să conţină definiţia sintactică a noii funcţii. De exemplu, un fişier funcţie numit „medie.m", care calculează media aritmetică, poate avea următoarea formă:

Exemple de implementat la laborator (utilizarea funcţiilor în MATLAB)

1. calculul mediei aritmetice a n numere introduse ca un vector coloană

```
function m=medie(x)

n=length(x);

s=0;

for i=1:n

    s=s+x(i);

end

m=s/n;
```

Apelaţi funcţia în linia de comandă după ce aţi introdus un vector coloană:

```
a=[1 2 3 4 5]

med=medie(a)
```

Integrarea sistemelor de ecuaţii diferenţiale

Pentru o ecuaţie diferenţială de ordin $n>1$, dată sub forma:

$$(1)\qquad \frac{\mathrm{d}^n y}{\mathrm{d}t^n} = f\left(t, y, \dot{y}, \ddot{y}, \ldots y^{(n-1)}\right), \quad t \in \left[t_0; t_f\right], \qquad y(t_0) = y_{1_0},\ \dot{y}(t_0) = y_{2_0}, \ldots,\ y^{(n-1)}(t_0) = y_{n_0},$$

se poate opta pentru alegerea noilor variabile astfel:

$$(2)\qquad y^{(i-1)}(t) = z_i(t), \quad i = \overline{1,n},$$

obţinându-se un sistem de ecuaţii diferenţiale în variabila z:

$$(3)\qquad \begin{cases} \dot{z}_1 = z_2 \\ \dot{z}_2 = z_3 \\ \dots \\ \dot{z}_n = f\left(t, z_1, z_2, \dots, z_n\right) \end{cases},$$

cu condiţia iniţială $\underline{z}(t_0) = \left[z_{1_0} = y_{1_0} \quad \dots \quad z_{n_0} = y_{n_0} \right]^T$.

Elemente necesare pentru integrarea numerică a EDO:

Aceste elemente trebuie văzute drept *date de intrare* care sunt furnizate unei metode numerice de integrare. Ele sunt listate mai jos:

1) Ecuaţia diferenţială de rezolvat trebuie mai întâi adusă la forma (3) printr-o *schimbare de variabilă* adecvată; rezultă astfel un sistem de ecuaţii diferenţiale de dimensiune egală cu ordinul ecuaţiei, *n*, care descrie legătura dintre derivatele celor *n* noi variabile, $\dot{z}_i$, şi valorile lor nederivate, z_i;

2) *intervalul de integrare* dat prin capetele acestuia, t_0 şi t_f;

3) *condiţiile iniţiale*, sub forma unui vector de dimensiune *n*, $\underline{y_0} = \left[y_{1_0} \quad \dots \quad y_{n_0} \right]^T$;

4) *pasul de integrare*, notat cu *h* – în toate metodele de integrare se consideră o diviziune de ordinul *m* a intervalului $\left[t_0; t_f \right]$ (o împărţire a acestui interval în intervale de aceeaşi lungime, *h*), în ale cărei puncte, $t_k = t_0 + k \cdot h,\ k \in \{0, \dots, n\}$, se calculează valorile soluţiei numerice a ecuaţiei, notate cu y_k, care sunt aproximări ale valorilor exacte, $y(t_k)$; este intuitiv evident că soluţia numerică se apropie cu atât mai mult de soluţia exactă cu cât diviziunea este mai fină (pasul de integrare, *h*, este mai mic);

5) un *subprogram* (funcţie) care calculează valoarea funcţiei f_z pentru orice moment de timp, *t*, şi care va fi apelat la fiecare pas de integrare, t_k, de programul unde este implementată metoda de integrare numerică;

6) alte date care se constituie ca *parametri constanţi* în funcţia f_z.

Metoda Taylor de ordinul 1, cunoscută şi ca **metoda Euler** sau **metoda Runge-Kutta de ordinul 1**:

$$(4)\qquad y_{k+1} = y_k + h \cdot f\left(t_k, y_k\right),$$

Exemplu: Sa se scrie un program Matlab care sa realizeze integrarea numerica a sistemului de ecuatii diferentiale:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{1}{2}\left(25 - 5x_2 - 25x_1\right)$$

Cerinte: - Se va utiliza metoda de integrare numerica Runge-Kutta de ordinal 1.

- Se vor afisa grafic, in aceeasi figura, rezultatele obtinute prin integrare

Rezolvare:

```
% program principal pentru integrare numerica
clear all;close all;
x=[7 10];
h=0.1;
t0=0;
tf=10;
t(1)=0;
i=1;
for m=0.1:h:tf,
    x(i+1,:)=x(i,:)+h* func(t,x(i,:));
    t(i+1)=m;
    i=i+1;
end
subplot(211);
hold on;
plot(t,x(:,1),'k');
subplot(212);
hold on;
plot(t,x(:,2),'k');

% implementare sistem de ecuatii diferentiale
function xd=func(t,x)
xd(1)=x(2);
   xd(2)=1/2*(25-5*x(2)-25*x(1));
```

În Matlab există metode de integrare implementate (metode ODE). Una din metodele Matlab cele mai utilizate este ode45.

Rezolvare:

```
% program principal pentru integrare numerica
clear all;close all;
x0=[7 10];
t0=0;
tf=10;
[t,x]=ode45('func',[t0 tf],x0);
subplot(211);
hold on;
plot(t,x(:,1),'k');
subplot(212);
hold on;
plot(t,x(:,2),'k');

% implementare sistem de ecuatii diferentiale
function xd=func(t,x)
xd(1)=x(2);
xd(2)=1/2*(25-5*x(2)-25*x(1));
xd=xd';
```

Teme:

1) Se vor rula toate instrucţiunile şi exemplele din cadrul cursului.

2. Să se calculeze funcţia:

$f(x)=\begin{cases} 2x+3, & daca\ x\in[-10,2] \\ 2x^2-1, & daca\ x\in(2,20] \end{cases}$ **pentru toate valorile întregi pentru care este definită.**

3. Să se integreze sistemul de ecuaţii (folosind ambele metode prezentate în cadrul cursului):

$\dot{x}_1 = 2x_1 + 3x_2 - 4x_3$

$\dot{x}_2 = -x_1 + 2x_2$

$\dot{x}_3 = -2x_1 + x_2 + 2$