1. Introducere

Mediul de programare MATLAB este un mediu de performanţă ridicată, dedicat calculului numeric şi reprezentărilor grafice în domeniul ştiinţific şi ingineresc. Numele de MATLAB (produs de The MathWorks Inc., http://www.mathworks.com) provine de la **Mat**rix Laborараtory, sistemul fiind proiectat pentru a facilita calculele matriciale.

In MATLAB sunt implementate un număr foarte mare de funcţii matematice, grafice, conţine biblioteci dedicate pentru diferite domenii, cum ar fi: calcul statistic, prelucrări de imagini, vedere artificială, control, optimizări, prelucrări de semnale etc. şi are capabilităţi de dezvoltare a interfeţelor grafice utilizator (GUI – graphical user interface). De asemenea are putere mare de calcul, se pot construi aplicaţii de sine stătătoare (aplicaţii executabile), se pot dezvolta coduri sursă pentru C++ etc.

In prezent, datorită avantajelor oferite, a devenit o platformă recunoscută în lumea universităţilor tehnice şi a cercetătorilor.

Există câteva caracteristici, care îl recomandă ca un limbaj potrivit pentru începători:

- programele, în general, sunt interpretate, nu compilate,
- tipurile de variabile nu trebuie declarate în avans,
- variabilele de bază sunt de tip vectorial. matricial sau numere complexe;
- are un mediu de dezvoltare şi de depanare integrat;
- are un număr foarte mare de funcţii implementate.

1.1. Mediul de dezvoltare integrat

Mediul de dezvoltare integrat conține o bară de meniu în partea superioară și mai multe ferestre. Selectarea ferestrelor care se doresc a fi afișate se poate face prin bifarea sau debifarea acestora în tabulatorul HOME, meniul Layout.

Printre ferestrele importante se numără:

- **Fereastra de comandă** – Command window (1 în Fig. 1). Ea reprezintă principala interfață interactivă cu Matlab. În momentul tastării unei comenzi în fereastra de comandă și a tastei Enter, comanda se va executa. Tastele de editare a liniilor de comandă sunt săgețile din partea dreaptă a tastaturii. Apăsarea tastei săgeată – sus (↑) determină editarea comenzii anterioare, putându-se face modificări fără a tasta complet linia respectivă.
- **Fereastra spațiului de lucru** – Workspace (3 în Fig. 1)– conține toate variabilele definite la un moment dat.În fereastră sunt indicate, de regulă, numele variabilei, clasa căreia îi aparține, valoarea acesteia.În fereastră mai pot fi afișate și alte proprietăți, care pot fi specificate prin executarea unui click dreapta pe bara ferestrei și bifarea acelor proprietăți care se doresc a fi vizualizate. Setul recomandat pentru afișare conține nume (Name), valoare (Value) și clasă (Class). Executarea unui dublu click pe o variabilă determină afișarea ferestrei de editare a variabilelor (Variable Editor).
- **Browser-ul directorului curent** (2 în Fig. 1) În directorul curent sau directorul de lucru, Matlab salvează fișierele și caută fișiere pentru deschidere (dacă nu se specifică o altă cale). Pictogramele din partea superioară a ferestrei permit utilizatorului să se deplaseze între directoare sau să creeze un director nou. Prin executarea unui dublu click pe un anumit director, acesta devine directorul curent.
- **Fereastra de editare** – Editor window – este zona unde se scriu programele. Lansarea editorului poate fi făcută din fereastra de comandă prin tastare edit+Enter.
- **Fereastra grafică** – Figures window – afișează diferitele grafice ce pot fi construite, instrumentul grafic fiind unul din cele mai utilizate pentru vizualizare,

- **Fereastra de editare variabilelor** – Variable Editor - afișează în formă tabelară valoarea, respectiv valorile unei variabile. Este foarte utilă în cazul matricelor. Pentru deschiderea ferestrei se poate executa click pe o anumită variabilă în fereastra spațiului de lucru (Workspace). Se pot modifica sau șterge anumite valori, iar la închiderea ferestrei de editare variabilele preiau valorile respective.
- **Fereastra cu istoricul comenzilor** – (4 în Fig. 1) sunt evidenţiate ultimele comenzi executate.

Fiecare din ferestre poate fi ancorată sau eliberată din meniul de tip pull down, aflat în colțul din dreapta sus al fiecărei ferestre. După eliberare, ferestrele se pot aranja în varianta dorită de utilizator și apoi ancora în poziția respectivă. Mai multe ferestre se pot găsi în același panou al ecranului, activarea uneia sau alteia putându-se face de la tabulatorii din partea superioară, partea laterală sau partea inferioară a panoului.

Variantă implicită a ferestrelor este prezentată în fig.1. Varianta cea mai recomandată a structurii conține 3 ferestre: fereastra de comandă, fereastra de editare, iar fereastra spațiului de lucru, browser-ul directorului curent și fereastra grafică împart același panou.

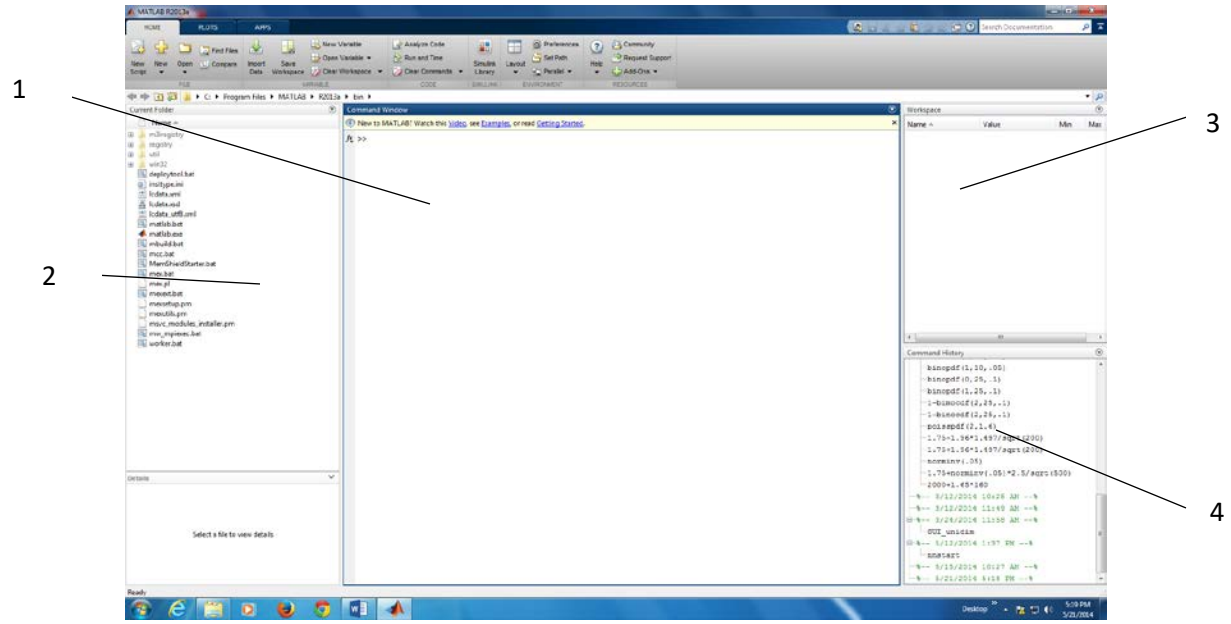

Fig. 1 Mediul de dezvoltare integrat Matlab – varianta implicită a poziționării ferestrelor

1.2. Variabile

Fiecare variabilă are o adresă specifică în memoria calculatorului, adică o locație unde se stochează. Adresa nu este manevrată direct de un program scris în MATLAB dar, variabila are un nume, prin intermediul căruia se lucrează cu informația stocată în ea. De asemenea, variabila aparține unei clase (sau unui anumit tip de variabilă), ce stabilește ce fel de informație este stocată în variabilă. Valoarea variabilei este informația conținută în variabilă și poate fi o valoare, o matrice, un șir de caractere, o anumită informație structurată.

În mod implicit, numerele sunt stocată în variabile de clasă **double**, care ocupă 64 de biți. Șirurile alfanumerice (litere, cifre, caractere speciale) sunt stocate în variabile de clasa **char**. Variabilele logice sau booleene, care pot lua valoarea true sau false se reține în variabile din clasa **logical**. Valoarea logică true este asociată cu 1, iar valoarea fals este 0.

Atribuirea unei valori la o variabilă se face cu semnul =. Ex. A = 5. Nu trebuie declarată variabila înaintea atribuirii, nu trebuie specificată clasa căreia îi aparține, calculatorul execută automat aceste operații, determinând clasa variabilei în funcție de valoarea atribuită.

Ex.:

```
a = 4 ; % class double
nume = 'Popescu Constantin' % class char
temperature = 67.35; % class double
isDone = true % class logical
```

Comentariile se introduc în program cu caracterul % și sunt ignorate de calculator, dar de regulă, sunt utile pentru cel care lucrează cu aplicația respectivă.

Introducerea unei instrucțiuni, fără a fi urmată de caracterul ;, în fereastra de comandă, determină un răspuns din partea calculatorului:

```
>> a = 4.2
a =
      4.2000
>> a = 5.5;
>>
```

Într-o linie se pot scrie mai multe instrucțiuni, separarea lor făcându-se cu ;, dar nu este recomandabil, programul se urmărește mult mai ușor dacă fiecare instrucțiune este scrisă pe linie separată. În cazul unei linii de comandă care ocupă mai mult de o linie în fereastra de editare se poate folosi caracterul …, ce indică continuarea comenzii pe linia următoare.

Membrul drept al unei instrucțiuni de atribuire poate fi o expresie, adică o combinație de numere, operatori și funcții.

```
>> x2 = x1+ 2*sin(teta);
```

Sintaxa general a unei instrucțiuni de atribuire este:

```
 nume_variabila = expresie;
```

Se va evalua membrul drept. Iar valoarea obținută va fi atribuită variabilei al cărei nume apare în membrul stâng. În cazul cînd în expresie apar variabile, se va lucra cu valorile acestora la momentul executării instrucțiunii. Modificarea ulterioară a valorii unei variabile din membrul drept nu va influența valoarea variabilei din membrul stâng, decât dacă se repetă operația de atribuire.

```
>>a = 2;
>>b = 8;
>>c = a + b
>> c =
     10
>> a = 0;
>> c
c =
     10
```

În membrul stâng al unei instrucțiuni de atribuire nu poate să apară decât numele variabilei pentru care se face atribuirea. Instrucțiunea de atribuire diferă esențial de egalitatea matematică. Toate expresiile ce urmează sunt greșite:

```
>> a = c = 1;
>> a + 1 = b – 3;
>> 4 = d;
>> 'NumeStudent ' = 'Popescu ';
```

În schimb, relația care nu are niciun sens din punct de vedere matematic, este corectă

```
>> index = index + 1; %incrementarea indexului
```

Numele variabilelor trebuie să înceapă cu o literă şi pot conţine mai multe caractere, dar primele 31 de caractere sunt semnificative. În numele variabilelor pot apărea litere, cifre și

caracterul _. Nu sunt admise alte semne de punctuație sau spații. Variabilele se pot deosebi prin utilizarea majusculelor, variabila a și A sunt variabile diferite.

Se recomandă utilizarea unor nume semnificative de variabile, care să sugereze semnificația acestora. În acest fel aplicația dezvoltată poate fi urmărită mult mai ușor. De asemenea trebuie evitată utilizarea ca nume de variabilă a cuvintelor cheie sau a funcțiilor predefinite.

Variabilele definite la un moment dat se pot vizualiza în spațiul de lucru, care face parte din memoria dinamică a unui calculator. La închiderea sesiunii de lucru în MATLAB variabilele se pierd. Ele pot fi salvate într-un fișier și reîncărcate ulterior, dar această operație se practică foarte rar.

Există o serie de comenzi ce pot fi utilizate pentru managementul variabilelor:

- clear a b – șterge variabilele a și b din memorie,
- clear – șterge toate variabilele din memorie,
- who – afișează toate variabilele definite la un moment dat,
- whos – afișează o listă detaliată cu variabilele definite,
- save – salvează spațiul de lucru curent în fișierul cu numele matlab.mat
- save nume_fișier - salvează spațiul de lucru curent în fișierul cu numele specificat de utilizator, fișier cu extensia .mat,
- load – încarcă în spațiul de lucru variabilele salvate în fișierul matlab.mat,
- load nume_fisier - încarcă în spațiul de lucru variabilele salvate în fișierul specificat.

În MATLAB există o serie de variabile predefinite:

- ans – numele variabilei răspuns (answer). În modul de lucru imediat, când se introduc comenzile direct în fereastra de comandă, la efectuarea unei operaţii fără alocarea unui nume, se afişează:

  ans=

  rezultat
- pi = 3.1416
- eps – cea mai mică valoare prin care pot diferi două numere ($2.2e^{-16}$);
- inf sai Inf – infinit, adică nedeterminare (poate apare la împărţirea cu 0);
- nan sau NaN – not-a-number, nedeterminare (poate apare la 0/0);
- i şi j – sunt numerele complexe egale cu $\sqrt{(-1)}$;
- date – data curentă sub forma unei variabile tip şir de caractere. Ex.: 19-May-2014.

Afişarea valorilor variabilelor se poate face în mai multe moduri. Cea mai simplă modalitate este tastarea numelui variabilei, urmată de ENTER. Se va afişa numele variabilei, însoţit de valoarea acesteia pe rândul următor.

Formatul implicit de afişare este cu 4 zecimale – formatul short. De ex. numărul 610/12 va fi afişat sub forma 50.8333. Mai există următoarele formate:

- long – cu 16 digiţi pentru partea întreagă şi partea fracţionară, ex. 50.833333333333336;
- short e – cu 5 digiţi plus exponentul, ex. 5.0833e+001;
- long e – 16 digiţi plus exponentul, ex. 5.083333333333334e+001;
- bank – cu 2 digiţi pentru partea fracţionară, ex. 50.83.

Formatul de afişare nu modifică reprezentarea internă a numărului. Pentru a modifica formatul de afişare se tastează în fereastra de comandă formatul dorit. Ex:

```
>> format long
```

O altă posibilitate de afişare a variabilelor este folosind comanda disp (display). Ea se poate aplica în două variante:

- disp(variabila) – afişează valoarea variabilei fără numele acesteia;
- disp('text') – afişează textul scris între caracterele apostrof.

Cu ajutorul acestei comenzi se poate controla mai bine modul de afişare al rezultatelor, adică se poate asocial text la nalorile numerice. Pentru a realiza acest lucru, rezultatul care se doreşte a fi vizualizat trebuie convertit din valoare numerică în şir de caractere. Comanda ce realizează acest lucru este:

num2str(nume_variabila) – converteşte variabila din valoare numerică în variabilă tip şir de caractere, iar [ ] realizează concatenarea șirurilor de caractere.

Ex.:

```
>> disp(['v0 [m/s]:',num2str(v0)])
v0 [m/s]:12.0482
```

1.3. Operatori şi funcţii matematice de bază

Operatorii ce pot fi folosiţi în Matlab sunt:

- aritmetici;
- relaţionali;
- logici.
- Operatorii aritmetici, în ordinea priorităţii de efectuare a calculelor, sunt:
- ^ (putere);
- (înmulţire);
- / (împărţire);
- + (adunare);
- (scădere).

Evident pentru schimbarea priorităţii operaţiilor se pot utiliza paranteze rotunde.

Operatorii relaţionali sunt: < (mai mic), <= (mai mic cel mult egal), > (mai mare), >= (mai mare cel mult egal), = = (egal), ~= (diferit).

Operatorii logici sunt: & - ŞI logic, | - SAU logic, ~ negare. Mai există funcţiile:

- **xor(x,y)** – sau exclusiv; returnează 1 (TRUE) dacă numai una din variabile este diferită de 0 (adevărată);
- **any(x)** – returnează 1, dacă un element al vectorului x este diferit de zero; dacă x este o matrice, returnează 1 pentru fiecare coloană în care există un element diferit de zero;
- **all(x)** – returnează 1, dacă toate elementele vectorului x sunt diferite de 0, iar dacă x este o matrice, returnează 1 pentru fiecare coloană în care toate elementele sunt adevărate sau diferite de 0.

Principalele funcţii matematice puse la dispoziţie de Matlab sunt:

- abs(x) – valoarea absolută sau modulul unui număr complex;
- asin(x) – arcsin(x);
- acos(x) – arccos(x);
- atan(x) – arctg(x);
- conj(x) – valoarea complex conjugată a lui x;
- cos(x), sin(x), tan(x) – funcţiile trigonometrice cunoscute pentru unghiurile exprimate în radiani;
- sind(x), cosd(x), tand(x) – funcţiile trigonometrice pentru unghiuri exprimate în grade;
- exp(x) – funcţia exponenţială, e^x;
- imag(x) – partea imaginară a lui x;
- log(x) – ln(x);
- log10(x) – $\log_{10}(x)$;
- real(x) – partea reală a lui x;

- round(x) – cel mai apropiat întreg;
- sign(x) – funcţia semn;
- rem(x,y) – restul împărţirii întregi lui x la y;
- sqrt(x) – radical de ordinul doi din x;
- fix(x) – rotunjire spre 0, adică rotunjire prin eliminarea părţii fracţionare;
- floor(x) – rotunjire spre -∞, adică cel mai apropiat întreg mai mic cel mult egal cu x;
- ceil(x) – rotunjire spre +∞, adică cel mai apropiat întreg mai mare sau cel mult egal cu x;
- isprime(x) – returnează valoarea true dacă x este număr prim;
- factor(x) – returnează un vector ce conţine descompunerea în factori primi ai variabilei x (fără elemental 1)ș
- primes(n) – returnează toate numerele prime mai mici decât n.

Matlab conţine numeroase comenzi şi funcţii, care pot fi vizualizate în documentaţia produsului. Pentru a accesa documentaţia se tastează în fereastra de comandă:

help <nume_functie>

doc <nume_functie>

Comanda help afişează o scurtă documentaţie legată de funcţia specificată, cu explicaţii legate de sintaxă, argumente etc., iar comanda doc va determina afişarea documentaţiei complete, de regulă, cu exemple de implementare.

1.4. Etapele rezolvării unei probleme numerice

Principalele etape de rezolvare ale unei probleme numerice sunt:

a) definirea problemei;
b) crearea unui model matematic;
c) dezvoltarea unei metode de calcul pentru rezolvarea problemei;
d) implementarea problemei de calcul;
e) testarea şi verificarea soluţiei.

Limitele dintre aceste etape pot fi, uneori, greu depistabile, iar pentru anumite probleme specifice, una sau două din etape pot fi mai importante decât altele.

a) Definirea problemei

Acest prim pas în rezolvarea unei probleme include:

- recunoaşterea şi definirea problemei în mod clar (uneori poate fi cel mai dificil pas);
- stabilirea întrebărilor la care trebuie date răspunsuri şi a datelor de ieşire;
- stabilirea cunoştiinţelor teoretice şi experimentale ce pot fi aplicate;
- stabilirea datelor de intrare.

După definirea problemei trebuie colectate toate datele şi informaţiile referitoare la problema respectivă, verificată acurateţea acestora. De asemenea, terbuie stabilite ce informaţii trebuie găsite (rezultate intermediare) înaintea rezultatelor finale.

b) Modelul matematic

Pentru a dezvolta modelul matematic trebuie:

- determinate principiile fundamentale ce pot fi aplicate;
- desenate grafice sau scheme bloc pentru o mai bună înţelegere a problemei;
- stabilite variabilele necesare şi a numelor acestora;
- redus enunţul problemei la expresii matematice;
- extras esenţialul din descrierea fizică a problemei.

c) Metoda de calcul

O metodă de calcul pentru rezolvare trebuie dezvoltată, pe baza modelului matematic. Acest fapt presupune:

- obţinerea unui set de ecuaţii ce permite calcularea parametrilor şi variabilelor dorite;
- dezvoltarea unui algoritm sau a unei metode pas cu pas pentru evaluarea ecuaţiilor implicate în soluţionare;
- descrierea algoritmului în termeni matematici şi apoi implementarea acestuia ca un program de calcul;
- trecerea în revistă a soluţiilor propuse, cu eventualele variante alternative.

Un algoritm este o mulțime de reguli, folosite într-o anumită succesiune, în vederea obținerii soluției unei probleme, prin executarea unui număr finit de operații.

Un algoritm are o serie de proprietăți, dintre care se menționează:

- claritatea – studiază toate situațiile posibile, nelăsând nimic la voia întâmplării;
- universalitatea – proprietatea de a fi aplicat la o clasă întreagă de problem sau cel puțin să fie compatibil indifferent de valoarea datelor de intrare;
- finalitatea – proprietatea de a găsi rezultatul într-un timp convenabil și cu o precizie sufficient de bună.

Viteza unui algoritm se apreciază după cum crește, în funcție de dimensiunile problemei, timpul necesar pentru rezolvarea problemei.

Ex. Fie polinomul

$$P(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$$

Pentru evaluarea polinomului într-un punct se poate pleca de la relația de recurență

$$P := P + a_ix^{n-i} \qquad i = 0, n$$

Acest algoritm este de ordinal n^2, realizându-se n(n-1) înmulțiri, deci timpul de rezolvare crește cu pătratul dimensiunii problemei. O variantă mai avantajoasă pleacă de la formula lui Horner

$$P := P * x + a_i \quad i = 0, n$$

Acest algoritm este de ordinul n.

d) Implementarea metodei de calcul

În cadrul acestui pas trebuie stabilit limbajul de programare în care se va lucra şi trebuie stabilit calculatorul pe care se implementează, pe baza unor considerente referitoare la puterea de calcul necesară. Algoritmul dezvoltat în etapa anterioară trebuie transpus într-un algoritm de calcul automat şi implementat ca un program de calculator. În multe situaţii paşii algoritmului se descompun în paşi mai mici, ce devin linii de comandă. Unul din avantajele oferite de Matlab este faptul că liniile de comandă se potrivesc foarte bine cu paşii uzuali ai problemelor inginereşti. În plus, Matlab-ul include o serie de biblioteci de algoritmi pentru analiză numerică, ceea ce simplifică mult rezolvarea problemelor.

e) Testarea şi verificarea soluţiei

Acesta este ultimul pas din rezolvarea unei probleme. Soluţia numerică trebuie verificată cu atenţie:

- se rulează programul pentru valori ale datelor de intrare la care se cunosc datele de ieşire;
- trebuie verificate rezultatele intermediare;
- trebuie verificată semnificaţia fizică a valorilor rezultate.

Exemplu de problemă: Se consideră un obiect de dimensiuni mici, aruncat cu 50 km/h la un unghi de 30° cu orizontala. Se cere: timpul de zbor şi distanţa parcursă.

a) Informaţii adiţionale necesare:

- Proprietăţiile obiectului şi mediului pot afecta traiectoria. Dacă obiectul ar fi uşor şi ar avea suprafaţă mare, rezistenţa aerului ar afecta mişcarea. Dacă mediul în care se desfăşoară

deplasarea ar fi în mişcare (de ex. vânt) acest lucru ar influenţa deplasarea corpului. Aceste informaţii lipsind, se presupune că mediul nu are nici o influenţă asupra deplăsării obiectului.

- Acceleraţia gravitaţională afectează mişcarea. Dacă nu există informaţii suplimentare se admite g = 9,81 m/s^2.
- Trebuie făcută conversia în unităţi SI. 50 km/h = 13,88 m/s. De asemenea unghiul exprimat în grade trebuie convertit în radiani: 30° = π/6 rad. Deci, datele de intrare ale problemei sunt: viteza iniţială v_0 = 13,88 m/s şi alfa = π/6.

b) Modelul matematic

Se folosesc următoarele notaţii:

- la momentul lansării, timpul este t = 0;
- viteza iniţială v = 13,88 m/s;
- unghiul de aruncare alfa = π/6 rad;
- coordonatele mişcării pe orizontală se notează x(t), iar pe verticală y(t);
- acceleraţia gravitaţională g = 9,81 m/s^2

Coordonatele obiectului pot fi exprimate cu relaţia:

$$\begin{cases} x(t) = vt\cos\alpha \\ y(t) = vt\sin\alpha - \frac{1}{2}gt^2 \end{cases}. \qquad (1.1)$$

c) Metoda de calcul

Obiectul va atinge solul când y(t) = 0, de unde rezultă:

$$y(t) = vt\sin\alpha - \frac{1}{2}gt^2 = 0. \qquad (1.2)$$

Ecuaţia (1.2) generează două soluţii $t_1 = 0$ şi $t_2 = \frac{2v\sin\alpha}{g}$. Evident soluţia t_2 este soluţia corectă.

La momentul t_2, distanţa parcursă este:

$$x(t) = vt_2\cos\alpha. \qquad (1.3)$$

d) Implementarea calculului

Programul de rezolvare se prezintă în continuare. Comenzile utilizate vor fi prezentate în continuarea acestui capitol. În cadrul aplicaţiei se trasează şi graficul traiectoriei, prezentat în fig. 1.1. Se declară un nou fişier m (din bara de meniu…), fapt ce va deschide fereastra de editare unde se introduce codul corespunzător. Se va salva fişierul sub denumirea aruncare_v0_alfa.m.

Programul se lansează în execuţie prin apăsarea tastei F5 (cu fereastra editorului activă) sau cu butonul verde Run.

```
%% aruncare_vo_alfa.m
%Calculare distanta si timp zbor
%pentru corp lansat unghi alfa si v0
% autor A. Davidescu
```

```
%% date intrare
g = 9.81;      %acceleratie gravitationala [m/s^2]
v0 = 13.88;    %viteza initiala [m/s]
alfa = pi/6;   %unghiul de lansare [rad]
```

```
%% calcule
t = 2*v0*sin(alfa)/g;  %timpul de zbor
d = v0*t*cos(alfa);   %distanta parcursa
```

```
%% afisare rezultate
disp('Calculare distanta si timp de zbor pentru corp aruncat cu v0 sub unghi
alfa')
disp(['v0 [m/s] = ',num2str(v0)])
disp(['alfa [rad] = ',num2str(alfa)])
disp(['timp [s] = ',num2str(t)])
disp(['distanta [m] =',num2str(d)])
```

```
%% reprezentare grafica a traiectoriei
timp=linspace(0,t,256);  %discretizare timp
x = v0*timp*cos(alfa);   %distanta parcursa
y = v0*timp*sin(alfa)-g*timp.^2/2;  % calculare inaltime
plot(x,y),axis equal,axis([0 17 0 4]),grid,xlabel('s [m]'),ylabel('h
[m]'),title('Traiectorie')
```

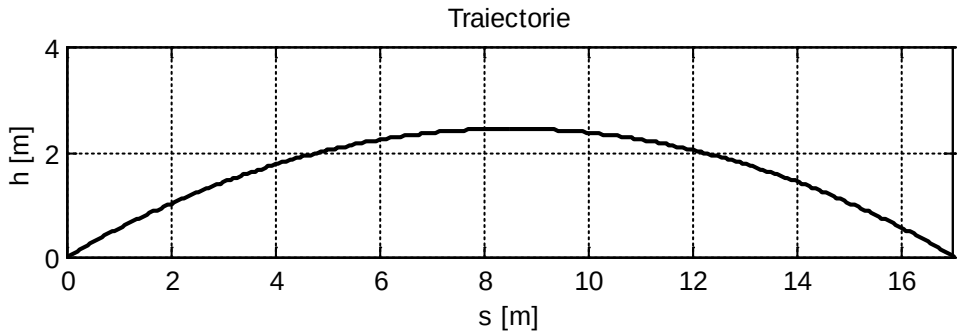

Fig. 1.1 Traiectoria obiectului lansat la 30°

Rezultatele sunt: timpul de zbor este 1,4158 [s], iar distanţa parcursă este 17,0293 [m].

Trebuie menționate cîteva observații generale legate de construirea unei aplicații, respective unele recomandări.

Simbolul %% marchează începutul unei secțiuni, iar MATLAB trasează automat o linie de demarcație între diferitele secțiuni.

În mod uzual un program conține mai multe blocuri:

- Un bloc de început ce conține numai comentarii. Începe cu numele fișierului și continuă cu o descriere sintetică a ceea ce face programul. De regulă, se specifică numele autorului. Deși acest bloc conține doar informații ignorate de MATLAB, ele au semnificație deosebită pentru utilizator. După salvarea fișierului, la tastarea în fereastra de comandă a comenzii:

  >>help aruncare_vo_alfa

Matlab va afișa comentariul introdus în primul bloc al programului.

- Un bloc ce specifică datele de intrare;
- Un bloc sau mai multe de efectuare a calculelor;
- Un bloc pentru afișarea rezultatelor, dacă este cazul.

După cum se observă în aplicație, fiecare bloc este demarcat separat și etichetat.

Numele variabilelor sunt legate de semnificația acestora. Evident, aplicația ar fi funcționat și cu alte denumiri (a, b, c, d sau a1, a2, a3,...), dar este mult mai ușor de urmărit și verificat dacă numele variabilei sugerează semnificația acesteia.

Este util să se adauge comentarii la scrierea codului. În exemplul prezentat sunt exagerat de multe comentarii, ținând cont de simplitatea codului, dar acest lucru a fost făcut în scop didactic. Tendința generală este să se scrie puține comentarii, ceea ce complică depanarea programelor și face reutilizarea lor mai greoaie în timp, din acest motiv se recomandă adăugarea comentariilor.

La afișarea rezultatelor sau la introducerea datelor de intrare se recomandă specificarea unităților de măsură, pentru a evita apariția unor erori.

Enunțuri

1. Să se scrie o aplicație atractie_gravit.m care să calculeze forța de atracție gravitațională (în N) dintre două corpuri pe baza relației $F = Gm_1m_2/r^2$. G – constanta gravitațională este $6.673*10^{-11}$ $N*m^2/kg^2$. Se va calcula forța pentru două persoane una de 80 kg și a doua de 60 kg, aflate la distanță de 2 m.
2. Să se scrie o aplicație sfoara_zmeu.m care să calculeze lungimea sforii unui zmeu ce se află la înălțimea H deasupra solului și formează unghiul alfa cu orizontala. Pentru ținerea zmeului este necesară o lungime l0 de sfoară, iar zmeul este ținut la înălțimea h0. Exemplu: H = 8,2 m, alfa = 2π/7, l0 = 0,25 m, h0 = 0.80 m.
3. Să se scrie o aplicație medie5.m care să calculeze media a 5 numere aleatoare între 1 și 10, generate cu funcția rand. Se va reseta generatorul de numere la fiecare rulare cu comanda `rng('shuffle')`.
4. Scrieți o aplicație coliziune_elastica.m, care să calculeze viteza finală a două obiecte după o ciocnire elastică, cunoscând masele și vitezele inițiale. Vitezele finale se calculează cu formulele:

$$V_{if} = \left(\frac{m_1 - m_2}{m_1+m_2}\right)\cdot v_{1i} + \left(\frac{2m_2}{m_1+m_2}\right)\cdot v_{2i}$$

$$v_{2f} = \left(\frac{2m_1}{m_1+m_2}\right)\cdot v_{1i} + \left(\frac{m_2-m_1}{m_1+m_2}\right)\cdot v_{2i}$$

   Date: m_1 = 5 kg, m_2 = 3 kg, v_{1i} = 2 m/s, v_{2i}= -4 m/s
5. Scrieți o aplicație frecare.m ce determină forța de frecare a unui obiect de masă m ce se deplasează pe un plan înclinat cu unghiul alfa cu orizontala, cunoscând coeficientul de frecare miu. Date: m = 0.8 kg, g = 9,81 m/s^2, alfa = 10˚, miu = 0.8.
6. Scrieți un program credit.m, care să calculeze suma de bani totală înapoiată pentru un credit (Cr) luat pe o perioadă de ani (ani) cu o dobândă anuală (dob) și valoarea dobânzii aferente. Valoarea ratei lunare are expresia:

$$R = \frac{Cr*dob\_lunara(1+dob\_lunara)^n}{(1+dob\_lunara)^n-1},$$

   unde dob_lunara este dobânda lunară, iar n este numărul de luni. Date: Cr = 100000, dob = 10%, ani = 5.
7. Scrieți un program triangulatie.m care să determine înălțimea unui obiect aflat la distanța d de instrumentul de măsurare. Se citește unghiul față de orizontală, alfa, iar instrumentul se află la înălțimea h față de sol. Date: d = 100 m, h = 1.2 m, alfa = 5˚ .
8. Scrieți o aplicație Rparalel.m care determină rezistența echivalentă a două rezistoare legate în paralel. Rezistența echivalentă se calculează:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

   Date: a. R_1 = 100 kΩ, R_2 = 100 kΩ, b. R_1 = 100 kΩ, R_2 = 1 Ω; c. R_1 = 100 kΩ, R_2 = 100 MΩ.
9. Scrieți un program vopsea.m care determină necesarul de vopsea pentru acoperirea unei încăperi cu dimensiunile (L x l x H) cunoscute, ținând cont că 1 l de vopsea acoperă 10

m^2, iar în încăoere sunt 2 ferestre cu dimensiunile Lf x lf și o ușă Lu x lu cunoscute. Date: L = 5 m, l = 4 m, H = 3 m, Lf = 1.2 m, lf = 1.5 m, Lu = 2 m, lu = 1 m.

1.5 Variabile tip şir de caractere

Variabilele tip şir de caractere se introduc prin caracterul `

```
>> Nume='Popescu';
>> Prenume='Adrian';
>> Adresa='Timisoara, str. Aries nr 10';
```

Concatenarea variabilelor şir se poate face cu ajutorul parantezelor drepte [ ]:

```
>> sir_nou=[Nume,Prenume,Adresa]
sir_nou =
PopescuAdrianTimisoara, str. Aries nr 10
```

Pentru a introduce spaţiu între şirurile concatenate, spaţiul trebuie prevăzut ca un şir separat introdus între celelate elemente:

```
>> sir_nou=[Nume,' ',Prenume,' ',Adresa]
sir_nou =
Popescu Adrian Timisoara, str. Aries nr 10
```

Trebuie făcută distincţia între o variabilă numerică şi o variabilă tip şir ce conţine numere:

```
>> x=4.23;
>> x_sir='4.23';
```

X şi xsir sunt variabile diferite. Pentru a putea efectua calcule cu valoarea numerică asociată variabilei x_sir, ea trebuie convertită în variabilă numerică cu comanda str2num().

Există câteva funcţii asociate variabilelor tip şir care se utilizează frecvent. Printre acestea se numără:

num2str(x) – returnează un şir de caractere corespunzător valorii stocate în variabila x;

str2double(s) – returnează un număr corespunzător şirului de caractere stocat în variabila s;

length(x) – returnează numărul de caractere din şirul s;

lower(s) – returnează şirul de caractere din variabila s, transformând toate caracterele în litere mici;

upper(s) – returnează şirul de caractere din variabila s, transformând toate caracterele în majuscule;

Nume(4) – returnează al patrulea caracter din variabila Nume;

Nume(2:5) – returnează caracterele de la 2 la 5 din variabila Nume.

La afişarea rezultatelor unui program de multe ori se recurge la concatenarea şirurilor de caractere pentru a putea tipări numele variabilei, valoarea acesteia şi unitatea de măsură asociată. Acest lucru se realizează cu comenzile num2str() şi disp()

```
>> disp(['Viteza initiala = ',num2str(vi),' m/s']);
Viteza initiala = 12.3 m/s
```

Frecvent la rularea unui program sunt necesare informaţii din partea utilizatorului, iar acest lucru se realizează cu comanda input(). Comanda se apelează diferit, în funcţie de tipul informaţiei solicitate. Pentru informaţii numerice:

```
>> NrAni=input('Introduceti numarul de ani = ');
```

In fereastra de comandă se va tipări:

```
Introduceti numarul de ani =
```

şi se aşteaptă din partea utilizatorului introducerea valorii respective şi apăsarea tastei Enter. Valoarea introdusă va fi alocată variabilei NrAni. In cazul când utilizatorul nu va introduce un număr, se va genera o eroare:

Error using input

Undefined function or variable 'test'.

Nu este obligatoriu ca ceea ce introduce utilizatorul să fie un număr, poate fi o expresie numerică sau o expresie în care intervin variabile definite anterior.

Pentru introducerea, cu comanda input(), a unei variabile tip şir de caractere, în argumentul funcţiei input apare argumentul s, care semnalizează variabila tip şir.

>> Prenume=input('Introduceti prenume: ','s')

1.6 Numere complexe, vectori si matrice

Numerele complexe se găsesc în aplicaţii din numeroase domenii: matematică, ştiinţe aplicate, inginerie, mai ales la reprezentarea naturii armonice a sistemelor vibratorii şi a câmpurilor oscilatorii.

MATLAB are predefinită variabila complexă i, respectiv j, $i^2 = j^2 = -1$.

Declararea numărului complex poate fi făcută în trei moduri:

- **z=1+j*2** sau **z=1+i*2**, apare explicit semnul înmulţirii între coeficientul părţii imaginare şi partea imaginară, ordinea acestora fiind indiferentă (i*2 sau 2*i);

- **z=1+2i** sau **z=1+2j**; nu apare semnul înmulţirii, dar în acest caz, obligatoriu, coeficientul părţii imaginare trebuie să preceadă variabila complexă; dacă se foloseşte atribuirea **z=1+j2**, se primeşte mesajul de eroare “undefined function or variable j2”, ceea ce indică faptul că această linie se interpretează ca o linie de atribuire în care variabila z este egală cu o variabilă j2 (necunoscută) plus o unitate.

- **z=complex(1,2)**.

Există mai multe funcţii definite pentru variabile complexe:

- **real(x)** – returnează partea reală a unui număr z (**z=3+4i → real(z) → 3);**

- **imag(z)** – returnează partea imaginară a unui număr z (**imag(z) → 4**);

- **conj(z)** – returnează numărul complex conjugat a lui z (**conj(z) →3-4i).**

În reprezentarea polară, numărul complex z se defineşte z = rcosθ +irsinθ, unde r este modulul numărului complex, $r = \sqrt{x^2 + y^2}$, iar $\theta = arctg(y/x)$. Funcţiile corespunzătoare din Matlab sunt:

- **abs(z)** – returnează modulul numărului complex;

- **angle(z)** – returnează argumentul în radiani, unghiul fiind cuprins între [-π,π].

Datorită faptului că un număr complex poate fi exprimat şi în formă exponenţială, $z = re^{i\theta} = r(\cos\theta + i\sin\theta)$, el poate fi exprimat în Matlab şi sub forma **z=r*exp(j*teta)**, unde r este modulul, iar teta este argumentul.

Indiferent de modul de exprimare a numărului complex, toate operaţiile uzuale (adunare, scădere, înmulţire, împărţire) se efectuează corect.

Definirea matricelor se poate face în mai multe feluri:

- printr-o listă explicită a elementelor, ce se introduc între paranteze drepte, fiind separate prin spaţiu sau virgulă; trecerea de la o linie la alta se face cu ";".

Ex.: A=[1 1 1;1,1,1] generează o matrice $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

- **x=(punct_start:increment:punct_stop).** Dacă incrementul nu este precizat, se consideră implicit 1. Ex.:

```
>>x=(0:.1:1)
  x =
 0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1
```

- **x=linspace(punct_start,punct_stop,numar_elemente)**. Vectorul generat are elementele spaţiate echidistant între punctul de start şi cel de stop. Dacă nu se specific numărul de elemente, se generează 100. Ex.:

```
>> x=linspace(0,90,11)

x =

    0    9   18   27   36   45   54   63   72   81   90
```

Accesarea elementelor unui vector se face cu ajutorul indicilor:

- **x(3)** – al treilea element a lui x;
- **x(1:5)** – primele cinci elemente a lui x;
- **y=x([8 2 9])** – y este un vector cu 3 elemente preluate din vectorul x, y(1) = x(8), y(2) = x(2) şi y(3) = x(9). Se poate folosi şi indicele "end", care înlocuieşte ultima valoare a indicelui. Ex.: **x(3:end)** este vectorul x din care au fost eliminate primele două elemente.

Se pot genera vectori cu spaţierea logaritmică a elementelor cu funcţia: **logspace(exponent_start,exponent_stop,numar_elemente).**
Ex.: **logspace(0,2,11)** – generează un vector ce începe cu $10^0 = 1$, se termină cu $10^2 = 100$ şi are 11 elemente.

Dimensiunea unui vector se obţine apelând funcţia **length(x)**, ce returnează numărul de elemente a vectorului x.

În cazul matricelor, adresarea se face similar A(2,3) este elementul matricei aflat pe linia 2, coloana 3. Pentru a apela o linie sau o coloană a matricei, se înlocuieşte indicele respectiv cu ":" Ex.: **A_1=A(:,1)** – variabila A_1 este egală cu prima coloană a matricei A.

- Se pot apela doar anumite părţi dintr-o matrice:
- **A_2=A(:,2:3)** – Matricea A_2 va fi formată din coloanele 2 şi 3 ale matricei A.
- **A_3=A(3:4,1:2)** – Matricea A_3 are 2 linii şi 2 coloane, conţinând elementele matricei A aflate pe linia 3 şi 4, coloana 1 şi 2.

Funcţia **size(A)** returnează un vector cu două elemente, primul este numărul de linii, iar al doilea este numărul de coloane.

Există câteva funcţii cu care se generează anumite matrici particulare:

- **zeros(n)** – generează o matrice nxn cu toate elementele egale cu 0;
- **zeros(n,m)** – generează o matrice nxm cu toate elementele egale cu 0;
- **zeros(size(a))** – generează o matrice de aceleaşi dimensiuni cu a, formată numai din elemente nule;
- **ones(n)** – generează o matrice nxn cu toate elementele egale cu 1;
- **ones(n,m)** – generează o matrice nxm cu toate elementele egale cu 1;
- **ones(size(a))** - generează o matrice de aceleaşi dimensiuni cu a, formată numai din elemente egale cu 1.
- **eye(n)** – generează matricea unitate nxn.
- **diag(x)** – comanda lucrează diferit în funcție de tipul variabilei x. Dacă x este o matrice comanda generează un vector ale cărei elemente sunt preluate de pe diagonala principală a matricei x. Dacă x este un vector, comanda generează o matrice pătrată având elementele egale cu 0, iar elementele de pe diagonala principală sunt preluate din vectorul x.

Se pot executa operaţii între scalari şi matrici. Ex.: **y=x-2**, dacă x este o matrice, y va fi o matrice cu aceleaşi dimensiuni, având elementele cu două unităţi mai mici decât matricea x.

Se pot executa operaţii între vectori, cu condiţia să aibe aceeaşi lungime. Pentru efectuarea operaţiilor de înmulţire, împărţire sau ridicare la putere, între elementele corespunzătoare fiecărui vector, se introduce înaintea operatorului semnul ”.”

Ex.: **c=a.*b** $\Rightarrow$ $c_i = a_i * b_i$;

c=a./b $\Rightarrow$ $c_i = a_i / b_i$;

c=a.^2 $\Rightarrow$ $c_i = a_i^2$.

În Matlab sunt definite funcţii ce permit efectuarea produsului scalar şi vectorial dintre doi vectori: **dot(a,b)** şi **cross(a,b)**. Vectorii a şi b trebuie să aibe aceleaşi dimensiuni, iar în cazul produsului vectorial (cross), vectorii trebuie să fie definiţi în spaţiul tridimensional.

O altă funcţie vectorială este **norm(x)**, ce returnează modulul sau norma vectorului x ($\|x\| = \sqrt{x_1^2 + x_2^2 + \mathrm{K} + x_n^2}$). Cu ajutorul acestei funcţii se poate normaliza un vector, adică se generează versorul având aceeaşi direcţie şi sens cu vectorul iniţial.

$$u_x = \frac{1}{\|x\|} x \quad \Rightarrow \quad \textbf{ux=x/norm(x)}$$

De asemenea, se poate determina unghiul dintre doi vectori x şi y, conform formulei:

$$\cos\theta = \frac{(x,y)}{\|x\| \cdot \|y\|} \quad \Rightarrow \textbf{teta=acos(dot(x,y)/(norm(x)*norm(y)))}$$

Proiecţia unui vector x pe o direcţie cunoscută, dată de vectorul y, se defineşte astfel:

$$z = \frac{(x,y)}{(y,y)} y = \frac{(x,y)}{\|y\|^2} y \,, \qquad (1.4)$$

ceea ce în Matlab este echivalent cu **z=(dot(x,y)/norm(y)^2)*y**.

Există o serie de funcţii utile în calculul vectorial și matricial:

- transpusa matricei A este A’: A’ = A^T;

- inversa unei matrici se face cu funcţia **inv(a);**
- calcularea determinantului unei matrice pătratice se face cu **det(a).**
- rezolvarea unui sistem de ecuaţii liniare se poate realiza prin utilizarea împărţirii la stânga. Dacă A este matricea sistemului şi b vectorul termenilor liberi, ecuaţia Ax = b, se rezolvă cu **x=A\b.**
- **max(a)** – returnează valoarea maximă a unui vector a sau dacă a este o matrice returnează un vector ce conține valorile maxime ale fiecărei coloane din matricea a; pentru a obține valoarea maximă a unei matrice se poate apela max(max(a));
- **min (a)** - returnează valoarea minimă a unui vector a sau dacă a este o matrice returnează un vector ce conține valorile minime ale fiecărei coloane din matricea a;
- **sum(a)** – returnează suma vectorului a sau suma fiecărei coloane din matricea a;
- **mean (a)** – returnează media aritmetică a unui vector, respectiv a coloanelor unei matrice;
- **sort(a)** – ordonează crescător un vector, respectiv dacă a este o matrice ordonează fiecare coloană; pentru ordonarea descrescătoare **sort(a, 'descend')**;
- **trace(a)** – returnează suma elementelor de pe diagonala principală a matricei a;
- **sortrows(x,nr_coloana)** – x trebuie să fie o matrice - ordonează crescător coloana cu numărul nr_coloană și rearanjează celelalte coloane pentru a menține corespondența între elemente. Pentru a face o sortare descrescătoare se introduce semnul – în fața numărului de coloană.
- **find(condiție)** – dacă x este vector se va genera un vector ce conține indicii elementelor care îndeplinesc condiția specificată (Ex. find(x>3) sau find(x==0). În cazul când se lucrează cu o matrice sintaxa este [I, J] = find(condiție). Se returnează doi vectori I și J, care conțin indicii de linie (I), respectiv de coloană (J) ai elementelor ce îndeplinesc condiția.

Se pot genera numere aleatoare cu comenzile rand(), randi() respectiv randn(). Sintaxa este:

rand(dimensiune) – generează numere aleatoare din repartiția uniformă cu limitele [0, 1].
rand(1,n) – generează un vector de n elemente aleatoare cuprinse între [0,1]

randi(valoare_maxima,dimensiune) – generează numere întregi, mai mici decât valoare_maximă

randi(30,1,200) – generează un vector de 200 de elemente aleatoare în intervalul [1,30].

randn(dimensiune) – generează numere aleatoare din repartiția normală standard (medie 0 și abatere standard 1).

În toate cazurile, dacă este specificată o singură dimensiune, se va genera o matrice pătrată.

Aceste două funcții generează numere pseudoaleatoare. Numerele aleatoare nu pot fi generate printr-un program, ci doar printr-un process fizic aleator. Secvența de numere generate este aleatoare, dar se repetă la fiecare sesiune de lucru. Pentru a evita acest lucru, trebuie modificată valoarea inițială, utilizând un număr generat pe baza ceasului intern al calculatorului. Pentru a realiza acest lucru se utilizează comanda:

rng('shuffle')

la începutul programului.

Exemplu de problemă

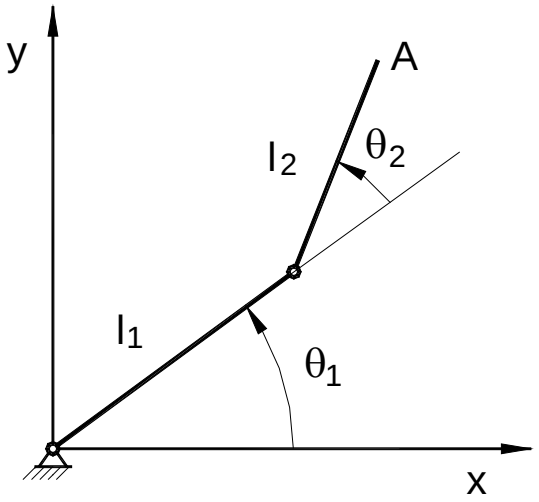

Fig. 1.2. Mecanism articulat

Fie un braţ articulat având 2 elemente cu lungimea l_1, respectiv l_2. Există câte un motor în fiecare cuplă cinematică. Se pune problema determinării unghiurilor θ_1 şi θ_2 pentru a deplasa efectorul final (punctul A) între două poziții cunoscute. Coordonatele efectorului final sunt conform fig. 1.2:

$$\begin{cases} x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{cases}. \qquad (1.5)$$

Mecanismul pleacă din repaus, iar în punctul final trebuie să ajungă cu viteză şi acceleraţie nulă.

Notând θ_{10} unghiul în poziţia iniţială şi θ_{11} în poziţie finală, se constată că se pot scrie următoarele ecuaţii pentru primul motor:

$$\begin{cases} \theta_1(0) = \theta_{10} \\ \theta_1(t_f) = \theta_{11} \\ \theta_1'(0) = \theta_1'(t_f) = 0 \\ \theta_1''(0) = \theta_1''(t_f) = 0 \end{cases},$$

adică 6 ecuaţii, de unde se constată faptul că legea de mişcare trebuie să fie o funcţie polinomială de gradul 5, pentru ca sistemul să fie determinat. Se admite legea de mişcare:

$$\theta_1(t) = a_0 + a_1 t^5 + a_2 t^4 + a_3 t^3 + a_4 t^2 + a_5 t .$$

Ţinând cont că $\theta_1(0) = \theta_{10}$, rezultă $a_0 = \theta_{10}$. Viteza are expresia:

$$\theta_1'(t) = 5a_1 t^4 + 4a_2 t^3 + 3a_3 t^2 + 2a_4 t + a_5 .$$

Datorită condiţiei, $\theta_1'(0) = 0$, rezultă $a_5 = 0$. Se calculează acceleraţia:

$$\theta_1''(t) = 20a_1 t^3 + 12a_2 t^2 + 6a_3 t + 2a_4 .$$

Datorită celei de-a treia condiţii, $\theta_1''(0) = 0$, rezultă $a_4 = 0$.

Pentru valorile corespunzătoare timpului t_f, se obţine următorul sistem matricial:

$$\begin{bmatrix} t_f^5 & t_f^4 & t_f^3 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 \\ 20t_f^3 & 12t_f^2 & 6t_f \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \theta_{11} - \theta_{10} \\ 0 \\ 0 \end{bmatrix}.$$

Similar, se raţionează pentru motorul ce acţionează a doua cuplă cinematică şi se obţine:

$$\begin{bmatrix} t_f^5 & t_f^4 & t_f^3 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 \\ 20t_f^3 & 12t_f^2 & 6t_f \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \theta_{21} - \theta_{20} \\ 0 \\ 0 \end{bmatrix}.$$

Considerând următorul exemplu numeric: t_f = 2 s, θ_{10} = -19°, θ_{20} = 44°, θ_{11} = 43°, θ_{21} = 151°, l_1 = 400 mm, l_2 = 300 mm, se prezintă aplicaţia de calcul a coeficienţilor mişcării şi graficul traiectoriei efectorului final.

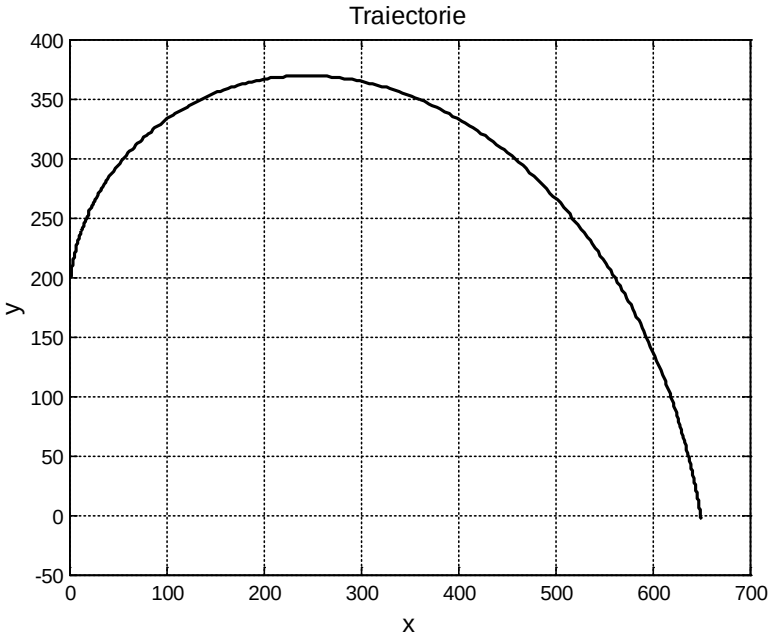

Fig. 1.3. Traiectoria mecanismului articulat

```
%% brat_articulat_2R.m
%Calculare lege de miscare in cuple motoare functie polinomiala grad 5
%pentru corp lansat unghi alfa si v0
% autor A. Davidescu
```

```
%% date intrare
tf=2;
teta10=-19*pi/180;teta11=43*pi/180;
teta20=44*pi/180;teta21=151*pi/180;
%calcul coeficienti
T=[tf^5 tf^4 tf^3;5*tf^4 4*tf^3 3*tf^2;20*tf^3 12*tf^2 6*tf];
c=[teta11-teta10;0;0];
disp('Coeficientii pentru legea teta1')
a=T\c
```

```
d=[teta21-teta20;0;0];
disp('Coeficientii pentru legea teta2')
b=T\d
%Traiectoria miscarii
l1=400;l2=300;
t=linspace(0,2,401);
tq=[t.^5;t.^4;t.^3];
teta1=teta10+a'*tq;
teta2=teta20+b'*tq;
x=l1*cos(teta1)+l2*cos(teta1+teta2);
y=l1*sin(teta1)+l2*sin(teta1+teta2);
plot(x,y),xlabel('x'),ylabel('y'),title('Traiectorie'),grid
```

În urma rulării acestei aplicaţii se obţin coeficienţii pentru θ_1: 0.2029, -1.0145, 1.3526, respectiv pentru θ_2: 0.3502, -1.7508, 2.3344, iar traiectoria braţului articulat se prezintă în fig. 1.3.