L5. Noțiuni de bază privind utilizarea MAPLE

5.1. Introducere

5.1.1. Aspecte generale

Mediul MAPLE este un software realizat de Waterloo Maple Inc., Ontario, Canada, special conceput pentru rezolvarea simbolică şi numerică a unei largi game de probleme matematice. Ceea ce deosebeşte MAPLE de alte produse similare este tocmai posibilitatea de a lucra cu variabile simbolice şi de rezolva anumite clase de probleme tot în formă simbolică, oferind astfel utilizatorului o experiență foarte apropiată de calculul matematic clasic.

Deşi toate elementele descrise în continuare sunt optimizate şi testate pe versiunea Maple 11, s-a preferat folosirea modului de scriere tipic maple şi nu varianta matematică mai elegantă disponibilă în această versiune. De aceea, este posibil ca utilizatorul să fie nevoit să modifice setările standard înainte de prima utilizare. Astfel, se accesează meniul *Tools*, apoi *Options*, tabul *Display* şi la opțiunea *Input display* se alege „Maple Notation". În final, se apasă butonul „Apply Globally".

Punctul de pornire pentru a scrie orice program este accesarea meniului *File* de unde se alege *New* şi *Worksheet Mode*.

5.1.2. Limbajul de programare MAPLE

Modul uzual de scriere a unui program în Maple este „linie cu linie". Aceasta înseamnă că fiecare linie este interpretată ca o comandă a cărei verificare şi execuție se face imediat ce utilizatorul apasă tasta *ENTER*. O astfel de linie trebuie să se termine fie prin caracterul „;", dacă se doreşte afişarea pe ecran a rezultatului, sau caracterul „:" dacă rezultatul nu trebuie afişat ci doar memorat pentru utilizare viitoare. În orice linie se poate face referire la rezultatul liniei anterioare prin utilizarea caracterului „%".

Limbajul MAPLE lucrează direct şi foarte intuitiv cu numere complexe, fracții, vectori şi matrici. Trebuie reținut că indexarea matricilor se face pornind de la 1 şi nu de la 0 aşa cum era cazul în C.

In MAPLE variabilele se pot utiliza direct, fără a fi necesară o declarare prealabilă a acestora. Se face diferențierea între litere mici şi litere mari. Dacă o variabilă nu primeşte o valoare numerică, atunci ea este considerată variabilă simbolică.

Trebuie subliniat că în MAPLE, spre deosebire de C, atribuirea unei valori se face cu simbolul „:=" iar simbolul „=" este utilizat de obicei în sensul său matematic uzual.

Structura decizională *if*

Sintaxa structurii *if* este:

if *condiție* then
    <instrucțiuni>
elif *condiție* then
    <instrucțiuni>
else
    <instrucțiuni>
fi;

Utilizarea ramurilor *elif* şi *else* este opțională.

Structura repetitivă *for*

În MAPLE se pot utiliza două varinate pentru structura *for*.

Prima variantă a structurii *for* este următoarea:
for *contor* from *valoare inițială* by *pas* to *valoare finală*
while *condiție*
do <instrucțiuni> od;
Utilizarea ramurii *while* este opțională.
A doua variantă a structurii *for* este următoarea:
for *variabilă* in *expresie* while *condiție*
do <instrucțiuni> od;

Structura *while*
Structura *while* are următoarea sintaxă:
while *condiție* do
<instrucțiuni>
od;

Utilizarea operatorului –>
Operatorul „–>" permite utilizatorului să își definească propriile funcții. Sintaxa este următoarea:
nume:=*var–>expresie analitica nume(var)*
SAU
nume:=(*var_1, var_2,…,var_n*)–>*expresie analitica nume(var_1, var_2, …, var_n)*
După ce a fost definită o funcție, valoarea acesteia într-un punct poate fi calculată apelând numele cu argument punctul repectiv.
Exemplu de utilizare:

```
>f:=x->x^2+7*x:
>f(10);
```

Definirea procedurilor
Sintaxa utilizată pentru a defini o procedură în MAPLE este următoarea:
nume:=proc (*param_1, param_2, …, param_n*)
local *var_1, var_2, …, var_n*;
options *lista_optiuni*;
instructiuni
end;
Utilizarea secțiunilor *local* și *options* nu este obligatorie.
Dacă se dorește ca procedura să întoarcă în mod explicit o anumită valoare, se poate utiliza funcția *RETURN*.

5.2. Exemple de calcul numeric în Maple

5.2.1. Rezolvarea ecuațiilor

5.2.1.1. Aspecte teoretice

În continuare vor fi prezentate două metode de rezolvare a ecuațiilor algebrice și transcendente. Prima este o transpunere a algoritmului Newton în limbajul MAPLE iar cea de-a doua presupune utilizarea unei variante a funcției predefinite „solve".
Metoda lui Newton
Considerând ecuația algebrică sau transcendentă

$$f(x) = 0$$

care are o singură rădăcină α în intervalul [a,b], aceasta poate fi determinată pe baza relației iterative Newton:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad i = 0,1,2,\ldots$$

Implementarea metodei Newton în MAPLE se poate face ca în exemplul de mai jos. Pentru a calcula expresia derivatei funcției se utilizează funcția predefinită *D(functie)*.

```
>newt:=proc(f,x0,eps,n)
>local i,dx,x,fd,fdd;
>x:=x0: fd:=D(f): fdd:=D(fd):
>if f(x)*fdd(x)<0 then
>ERROR (`Conditia alegerii punctului de start nu este satisfacuta`)
>else
>i:=0; dx:=f(x);
>if abs(dx)<eps then
>print(`Radacina ecuatiei este:`);
>print(evalf(x));
>else
>print(`Rezultatele procesului iterativ:`);
>while abs(dx)>eps and i<n do
>i:=i+1;
>x:=x-dx;
>print(evalf(x));
>dx:=f(x);
>if abs(fd(x))>eps then dx:=dx/fd(x) fi;
>od;
>if i<n then print(`Ultima valoare este radacina ecuatiei`)
>else print(`Numarul iteratiilor este insuficient`)
>fi;
>fi;
>fi;
>end:

>f:=x->x^2-2: x0:=5: eps:=10^(-4): n:=20:
>newt(f,x0,eps,n);

>f:=x->x-sin(x)-0.25: x0:=1.17: eps:=10^(-6): n:=20:
>newt(f,x0,eps,n);

>f:=x->1-(x+1)^2: x0:=3: eps:=10^(-5): n:=6:
>newt(f,x0,eps,n);
```

Funcția „solve”

Sintaxa funcției „solve” este următoarea:

solve(*ecuatie*, *var*)

Parametrul „var” poate fi omis şi în acest caz ecuația va fi rezolvată în raport cu toate variabilele existente. Funcția returnează soluțiile sub forma unei secvențe de expresii sau numeric dacă nu se poate identifica o astfel de expresie. Dacă se dorește exprimarea numerică a expresiei returnate se poate utiliza funcția *evalf()*.

În continuare sunt date câteva exemple de rezolvare a unor ecuații cu funcția „solve”.

```
>ec1:=x^4+4*x^3-2*x^2-4*x=-1;
>solve(ec1,x);
>sol:=[solve(ec1,x)];
>evalf(sol);

>solve(sin(x)-2^y=3,y);

>ec2:=3*2^x+2^(2*x)=16;
>solve(ec2,x);
>evalf(%);
```

5.2.1.2. Desfăşurarea laboratorului

1. Se verifică algoritmii prezentaţi.

5.2.2. Rezolvarea sistemelor de ecuaţii liniare

5.2.2.1. Aspecte teoretice

În continuare vor fi prezentate două metode de rezolvare a sistemelor de ecuaţii liniare. Prima este o transpunere a algoritmului Gauss-Jordan în limbajul MAPLE iar cea de-a doua presupune utilizarea unei variante a funcţiei predefinite „solve".

Metoda Gauss-Jordan

Pentru implementarea metodei Gauss-Jordan în MAPLE există două funcţii predefinite: *gaussjord* şi *backsub*.

Algoritmul de mai jos rezolvă sistemul de ecuaţii

$$\begin{cases} 2x_1 + x_2 + 2x_3 = 0 \\ x_1 + 3x_2 + x_3 = 0 \\ 2x_1 + x_2 + x_3 = 1 \end{cases}.$$

```
>with(linalg):
>A1:=matrix(3,3,[[2,1,2],[1,3,1],[2,1,1]]);
>b1:=vector([0,0,1]):
>C1:=augment(A1,b1);
>GJ1:=gaussjord(C1):
>x:=backsub(GJ1);
```

Funcţia „solve"

Pentru a rezolva sisteme de ecuaţii se poate utiliza următoarea sintaxă a funcţiei „solve":

solve({*ec_1, ec_2, …, ec_n*},{*variabile*})

Un exemplu de utilizare a funcţiei „solve" este dat mai jos.

```
>sist1:={x+y+z+2*t=12, -3*x+2*y+4*z+t=3, x+y-2*z-t=6, 4*x-y-z+t=7}:
>solve(sist1,{x,y,z,t});

>sist2:={x+3*y+z-t=7, -3*x+y-2*z+2*t=1, x-y-z+3*t=5}:
>solve(sist2,{x,y,z});
```

5.2.2.2. Desfăşurarea laboratorului

1. Se verifică algoritmii prezentaţi.

5.2.3. Interpolarea

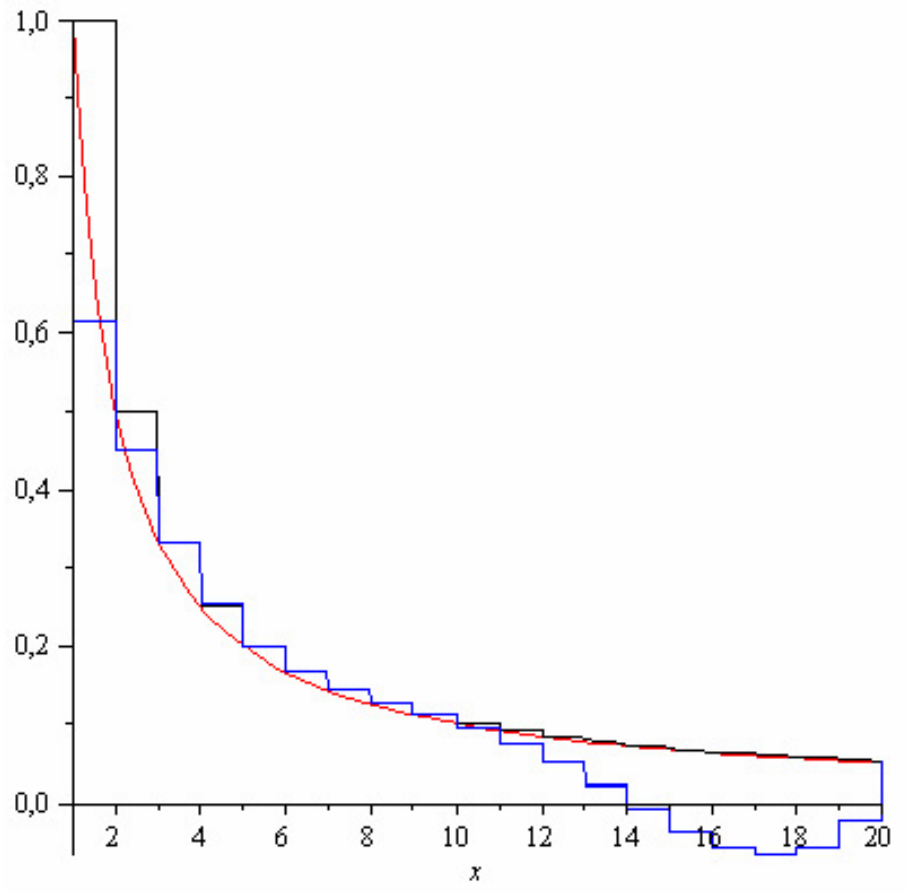
5.2.3.1. Aspecte teoretice

În MAPLE există o funcţie predefinită numită „interp" care determină polinomul de interpolare Lagrange al unei funcţii pentru care se cunosc valorile în anumite puncte. Sintaxa funcţiei „interp" este:

interp(*vector abscise, vector ordonate, var*)

Algoritmul de mai jos determină polinomul de interpolare Lagrange asociat funcţiei *f(x)=1/x* şi reprezintă graficul funcţiei, al funcţiei prin punctele de tabelare şi al polinomului prin aceleaşi puncte.

```
> u:=vector(20):
> x:=[3,5,7,9,20]:
> y:=vector(5):
> f:=x->1/x:
> for i from 1 to 5
> do y[i]:=f(x[i]);od:
> x:=convert(x,vector):
> L(z):=interp(x,y,z);
> for i from 1 to 20
> do u[i]:=subs(z=i,L(z));od:
> u(1);
> m:=evalf(u());
> l:=vector(20):
> for i from 1 to 20
> do l[i]:=f(i);od:
> l(1);
> l:=evalf(l());
> plot([f(x),l[x],m[x]],x=1..20,color=[red,black,blue]);
```



5.2.3.2. Desfăşurarea laboratorului

1. Se verifică funcționarea algoritmului prezentat.

5.2.4. Calcularea integralelor unidimensionale

5.2.4.1. Aspecte teoretice

MAPLE dispune de două funcţii, „trapezoid" şi „simpson", care implementează algoritmul metodei trapezului cu control automat al pasului şi respectiv algoritmul metodei Simpson cu control automat al pasului. Cele două funcţii returnează valoarea simbolică a integralei.

Sintaxa funcţiei „trapezoid" este:

trapezoid(*expresie_functie*, *x=a..b*, *n*)

Sintaxa funcţiei „simpson" este:

simpson(*expresie_functie*, *x=a..b*, *n*)

Pentru ambele funcţii *a* şi *b* reprezintă capetele intervalului de integrare iar *n* numărul de subintervale considerate în formula de calcul. Acest ultim parametru este opţional, având valoarea implicită 4.

Algoritmul de mai jos exemplifică utilizarea funcţiilor „trapezoid" şi „simpson" pentru a calcula o serie de integrale.

```
>with(student):
>trapezoid(sqrt(ln(x)+2),x=2..4,6);
>evalf(%);
>simpson(sqrt(ln(x)+2),x=2..4,6);
>evalf(%);

>trapezoid(arctan(x),x=-Pi/2..3*Pi/2,40);
>evalf(%);
>simpson(arctan(x),x=-Pi/2..3*Pi/2,40);
>evalf(%);
```

5.2.4.2. Desfăşurarea laboratorului

1. Se verifică algoritmul prezentat.
2. Se modifică algoritmul pentru a calcula integralele $\int_{-1}^{1} x^{10} dx$ şi $\int_{0}^{\pi} \tan(2+\sin(x))dx$.

5.2.5. Rezolvarea ecuaţiilor diferenţiale ordinare

5.2.5.1. Aspecte teoretice

Rezolvarea unei ecuaţii diferenţiale ordinare se poate face în MAPLE utilizând funcţia „dsolve" care are următoarea sintaxă:

dsolve(*ec*, *var*, *param*)

Ec reprezintă ecuaţia diferenţială sau o mulţime de ecuaţii diferenţiale ale unui sistem şi condiţia sau condiţiile iniţiale. *Var* reprezintă variabila în raport cu care se face rezolvarea.

Funcţia „dsolve" identifică în mod predefinit soluţia explicită analitică a ecuaţiei şi dacă nu se precizează condiţia iniţială se va utiliza o exprimare parametrică, folosind constantele *_C1*, *_C2*.... Acest comportament poate fi modificat cu ajutorul argumentului opţional *param* care poate avea următoarele forme:

- type=exact SAU type=series SAU type=numeric

  Se precizează dacă se doreşte o soluţie exactă a ecuaţiei (valoarea predefinită), o soluţie sub forma unei serii sau o soluţie numerică.
- explicit=true SAU explicit=false

  Se precizează dacă se doreşte ca soluţia să fie exprimată, explicit sau nu (valoarea predefinită), în funcţie de variabilele dependente.
- method=rkf45 SAU method=dverk78 SAU method=laplace

Se precizează algoritmul utilizat pentru rezolvarea numerică: Runge-Kutta de ordin 4-5 (valoare predefinită), Runge-Kutta de ordin 7-8 respectiv transformare Laplace.

În continuare sunt date câteva exemple de rezolvare a unor ecuații diferențiale ordinare utilizând funcția „dsolve". Pentru a exprima derivata unei funcții se utilizează funcția „diff".

```
>ecd1:=diff(y(x),x)+3*y(x)=0;
>dsolve(ecd1,y(x));
>dsolve({ecd1,y(0)=2},y(x));

>ecd2:=diff(y(x),x)+3*y(x)=4*x+3;
>print(`Solutia analitica`);
>sol(x):=dsolve({ecd2,y(0)=1},y(x));
>dsolve({ecd2,y(0)=1},y(x),series);
>s1:=dsolve({ecd2,y(0)=1},y(x),numeric):
>print(`Val. functiei determina cu met. Runge-Kutta de ordin 4-5:`);
>for i from 1 to 5 do
>print (s1(i));
>od;
>s2:= dsolve({ecd2,y(0)=1},y(x),type=numeric,method=dverk78):
>print(`Val. functiei determina cu met. Runge-Kutta de ordin 7-8:`);
>for i from 1 to 5 do
>print (s2(i));
>od;
>Digits:=20:
>print(`Val. functiei determina analitic:`);
>for i from 1 to 5 do
>subs(x=i,sol(x));
>evalf(%);
>od;
```

5.2.5.2. Desfăşurarea laboratorului

1. Se verifică algoritmul prezentat.
2. Se va modifica algoritmul pentru a rezolva ecuația $y' = \frac{3-4y}{2x}$, $y(1) = -4$.

L6. Rezolvarea problemei transportului folosind MAPLE

6.1. Introducere

6.1.1. Probleme de optimizare

În numeroase domenii de activitate, în special cel economic, se întâlnesc frecvent probleme care presupun optimizarea unor cerințe în sensul de maxim sau minim iar cel mai adesea singurele soluții de interes sunt cele care au componente pozitive. Exprimarea matematică a unei astfel de probleme poate fi: să se determine valorarea maximă (sau minimă) a unei funcții f (numită de obicei funcție obiectiv) ale cărei variabile trebuie să satisfacă:

- un sistem de relații date sub forma unor ecuații şi/sau inecuații liniare nestricte, denumite generic restricții;
- cerința de a lua numai valori numerice nenegative.

Soluțiile problemei enunțate mai sus care satisfac restricțiile şi condiția de nenegativitate se numesc soluții admisibile. Dintre aceastea, cea care îndeplineşte cerința de maxim sau minim se numeşte soluție optimă. Este posibil ca o problemă să aibă soluții dar acestea să nu fie admisibile, aşa cum este posibil ca soluția optimă să aibă valoare finită sau infinită.

Există două forme uzuale de scriere a problemelor de optimizare: forma canonică şi forma standard.

O problemă *în formă canonică de maximizare* se scrie astfel:

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j \le b_i \quad i=1,\dots,m \\ x_j \ge 0 \quad j=1,\dots,n \\ (\max) f = \sum_{j=1}^{n} c_j x_j \end{cases} \quad \text{sau matricial} \quad \begin{cases} Ax \le b \\ x \ge 0 \\ (\max) f = cx \end{cases}$$

O problemă *în formă canonică de minimizare* se scrie astfel:

$$\begin{cases} \sum_{j=1}^{n} a_{ij}x_j \ge b_i \quad i=1,\dots,m \\ x_j \ge 0 \quad j=1,\dots,n \\ (\min) f = \sum_{j=1}^{n} c_j x_j \end{cases} \quad \text{sau matricial} \quad \begin{cases} Ax \ge b \\ x \ge 0 \\ (\max) f = cx \end{cases}$$

Orice problemă de optimizare poate fi adusă la forma canonică, fără a-i modifica soluțiile, pe baza următoarelor proprietăți:

- o egalitate se poate înlocui cu două inegalități de sens contrar;
- sensul unei restricții poate fi schimbat prin înmulțire cu -1;
- sensul optimizării funcției obiectiv poate fi modificat având în vedere că:

$$\min f(x) = -\max[-f(x)]$$

O problemă de optimizare este *în formă standard* dacă toate restricțiile sunt egalități. Aducerea la forma standard nu modifică soluțiile problemei inițiale şi se poate face astfel:

- o restricție inegalitate din problema inițială de tipul „≤” (respectiv de tipul „≥”) se transformă în egalitate prin adăugarea (respectiv prin scăderea) unei variabile nenegative din membrul său stâng;
- restricțiile egalități nu se modifică;
- noile variabile introduse nu apar în funcția obiectiv a problemei inițiale.

Mai jos este dat un exemplu de aducere a unei probleme la forma standard:

$$\begin{cases} (\max) f = 7x_1 + 9x_2 + 8x_3 \\ 5x_1 + 2x_2 - x_3 \geq 4 \\ 3x_1 + x_2 + x_3 = 5 \\ x_1 + 2x_2 + 3x_3 \leq 9 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases} \Rightarrow \begin{cases} (\max) f = 7x_1 + 9x_2 + 8x_3 \\ 5x_1 + 2x_2 - x_3 - x_4 = 4 \\ 3x_1 + x_2 + x_3 = 5 \\ x_1 + 2x_2 + 3x_3 + x_5 = 9 \\ x_j \geq 0, j = 1, \ldots 5 \end{cases}$$

6.1.2. Metoda simplex

Metoda simplex a fost inventată de matematicianul englez G.B. Dantzig şi reprezintă un procedeu de rezolvare a problemelor de maxim. Metoda porneşte de la o soluție inițială cunoscută a problemei scrisă în forma standard şi construieşte un şir de soluții pentru care valoarea funcției obiectiv creşte progresiv. De asemenea, metoda oferă un test simplu de recunoaştere a optimalității unei soluții precum şi a situației în care optimul este infinit.

Metoda simplex este utilizată în calculul matematic împreună cu o serie de tabele standard de valori. Pentru calculul numeric au fost dezvoltați o serie de algoritmi care implementează această metodă şi oferă direct soluția problemei de optimizare.

6.1.3. Problema transportului

Problema transportului este un exemplu foarte des întâlnit de problemă de optimizare. În general, în practică trebuie deplasată o cantitate Q ce poate fi materie, energie, informație etc. din unele locuri numite „surse” în alte locuri numite „destinații” pe anumite rute de legătură.

În studiul metodelor numerice, problemele de transport sunt interesante numai dacă sunt îndeplinite următoarele ipoteze:

- cel puțin o sursă poate aproviziona mai multe destinații şi cel puțin o destinație poate fi aprovizionată de la mai multe surse;
- există un cost al deplasării de la un punct al rețelei de transport la altul, cost care poate fi exprimat, de exemplu, în bani, timp sau distanță.

Problema clasică de transport poate fi formulată astfel: un produs se află disponibil la sursele S_1, S_2, ..., S_m în cantitățile a_1, a_2, ..., a_m şi este cerut pentru consum la destinațiile D_1, D_2, ..., D_n în cantitățile b_1, b_2, ..., b_n. Se presupune cunoscut costul c_{ij} al transportului de la S_i la D_j şi se pune problema satisfacerii cererii în punctele de consum la un cost total de transport minim.

Evident, o condiție necesară şi suficientă pentru existența unei soluții a problemei formulate este ca totalul cantităților disponibile să acopere totalul cererilor:

$$\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j .$$

Considerând îndeplinită condiția de mai sus şi notând cu x_{ij} cantitatea livrată de la S_i la D_j, problema clasică de transport se transpune matematic astfel:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} \le a_i \quad i = 1,\ldots,m \\ \sum_{i=1}^{m} x_{ij} \ge b_j \quad j = 1,\ldots,n \\ x_{ij} \ge 0 \quad i = 1,\ldots,m;\ j = 1,\ldots,n \\ (\min) f = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \end{cases}$$

Problema de transport este *echilibrată* dacă:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \,.$$

Din punct de vedere matematic, o problemă de transport echilibrată se scrie astfel:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i \quad i = 1,\ldots,m \\ \sum_{i=1}^{m} x_{ij} = b_j \quad j = 1,\ldots,n \\ x_{ij} \ge 0 \quad i = 1,\ldots,m;\ j = 1,\ldots,n \\ (\min) f = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \end{cases}$$

Din relația anterioară se observă că problema de transport echilibrată este o problemă de minim scrisă în forma standard. Prin urmare, aceasta ar putea fi rezolvată utilizând algoritmul simplex dar în teoria metodelor numerice a fost dezvoltată o metodă specifică, derivată din acest algoritm, care este mult mai eficientă.

6.2. Algoritmul simplex şi problema transportului în Maple

6.2.1. Introducere

Maple nu dispune de funcții dedicate metodei transportului însă are implementată o bibliotecă, numită chiar „simplex", cu funcții care rezolvă probleme de optimizare folosind algoritmul simplex. Dintre acestea, cele mai utile sunt „feasible", „maximize" şi „minimize".

Sintaxa funcției „feasible" este următoarea:

feasible(*mult_inec*, *opt*)

Funcția „feasible" întoarce valoarea *true* sau *false* după cum mulțimea de inecuații *mult_inec* are sau nu soluție. Parametrul *opt* poate lua valoarea „NONNEGATIVE" sau „UNRESTRICTED"; în primul caz se iau în considerare numai soluțiile admisibile pe când în al doilea caz se iau în considerare toate soluțiile.

Funcțiile „maximize" şi „minimize" au aceeaşi sintaxă dar, aşa cum sugerează numele lor, diferă prin rezultat. Sintaxa uzuală a funcției „minimize" este:

minimize(*func_obiectiv*, *mult_inec*, NONNEGATIVE)

Funcția „minimize" întoarce valorile pozitive ale variabilelor pentru care este minimizată valoarea funcției obiectiv *func_obiectiv*, respectând restricțiile din mulțimea de inecuații *mult_inec*.

Din descrierierea de mai sus se poate observa că rezolvarea unei probleme de transport în Maple se reduce la transpunerea acesteia în forma matematică şi utilizarea funcţiei „minimize”.

6.2.2. Exemplu de rezolvare a unei probleme de transport în Maple

Să se rezolve problema de transport echilibrată definită prin următoarele date:

	D_1	D_2	D_3	D_4	Disponibil
S_1	3	3	1	4	15
S_2	3	4	3	6	17
S_3	4	3	6	2	18
Necesar	10	12	9	19	50

Transpunerea matematică a problemei de mai sus este:

$$\begin{cases} x_{11}+x_{12}+x_{13}+x_{14}=15 \\ x_{21}+x_{22}+x_{23}+x_{24}=17 \\ x_{31}+x_{32}+x_{33}+x_{34}=18 \\ x_{11}+x_{21}+x_{31}=10 \\ x_{12}+x_{22}+x_{32}=12 \\ x_{13}+x_{23}+x_{33}=9 \\ x_{14}+x_{24}+x_{34}=19 \\ x_{ij}\geq 0 \\ (\min) f=3x_{11}+3x_{12}+x_{13}+4x_{14}+3x_{21}+4x_{22}+3x_{23}+6x_{24}+4x_{31}+3x_{32}+6x_{33}+2x_{34} \end{cases}.$$

Soluţia problemei este:

0	5	9	1
10	7	0	0
0	0	0	18

Programul de mai jos verifică dacă sistemul de inecuaţii are soluţie admisibilă şi rezolvă problema de transport dată.

```
>with(simplex):
>f:=3*x11+3*x12+x13+4*x14+3*x21+4*x22+3*x23+6*x24+4*x31+3*x32+
6*x33+2*x34;
>inec:={x11+x12+x13+x14=15,x21+x22+x23+x24=17,x31+x32+x33+x34=18,
x11+x21+x31=10,x12+x22+x32=12,x13+x23+x33=9,x14+x24+x34=19}:
>feasible(inec,NONNEGATIVE);
>print(`Solutia problemei de transport:`);
>a:=minimize(f,inec,NONNEGATIVE);
>print(`Costul minim este:`);
>eval(f,a);
```

6.2.3. Desfășurarea laboratorului

a) Se verifică exemplul prezentat la punctul 6.2.2.
b) Să se scrie un program în Maple care să rezolve problema de transport echilibrată definită prin următoarele date:

	D_1	D_2	D_3	D_4	Disponibil
S_1	4	2	5	4	110
S_2	6	7	3	8	80
S_3	3	5	4	5	90
Necesar	120	90	50	20	280

Soluția problemei este:

0	90	0	20
30	0	50	0
90	0	0	0