SISTEME nestandard. METODE noi de rezolvare

Laviniu Bejenaru

Modelul uman de viață a făcut ca permanet să ne lovim de rezolvarea unor probleme. A rezolva o problemă înseamnă a pleca de la anumite date cunoscute și a încerca să găsim soluții potrivite. Organizarea datelor și a relațiilor dintre ele atât cunoscute cât și necunoscute este realizată sub forma a ceea ce numim ecuație. Ecuațiile au, în afară de aspectul strict matematic al unor relații reci, o anumită semnificație. Dacă problemele necesită mai multe relații, atunci aceste relații, deși independente, se pot grupa în sisteme de ecuații.

Cu ajutorul sistemelor de ecuații se pot rezolva, în principiu, mai multe probleme dacât cu ajutorul ecuațiilor; de fapt, dacă abstractizăm, o ecuație poate fi privită ca un sistem de ecuații în care numărul de ecuații componente este egal cu unu. Posibilitatea rezolvării sistemelor conduce la rezolvarea problemelor și chiar la o interpretare a lor.

Prezentare

Problemele cărora le căutăm rezolvare conduc adesea la ecuații și sisteme de ecuații care nu se încadrează într-un anumit tip, cum ar fi ecuații de gradul unu, doi, binome etc., sisteme de ecuații liniare sau sisteme simetrice.

Putem vorbi astfel despre ecuații nestandard sau despre sisteme de ecuații nestandard sau, pe scurt, sisteme nestandard.

Desigur, problemele complexe din industrie sau economie generează sisteme a căror tratare nu se atinge în liceu, cum ar fi sistemele de ecuații diferențiale. Totuși, în cele ce urmează, vom trata aici doar cazul unor sisteme elementare, nestandard, care se studiază și în învățământul preuniversitar.

Să dăm un exemplu:

$$(1)\quad \begin{cases} x(x+y+4z)-4z^2=0 \\ y(x+y+4z)-8z^2=0 \\ xyz=8 \end{cases}$$

Rezolvarea sistemelor de acest tip poate fi destul de dificilă. Găsirea anumitor amănunte sau observații poate conduce de cele mai multe ori la rezolvare.

Prin intermediul programului *MAPLE V*, însă, avem la dispoziție mai multe instrumente pentru a rezolva aceste sisteme.

Preambul

Pentru a putea folosi funcții și proceduri dintr-un anumit *unit*, în *Pascal* trebuie să informăm compilatorul prin instrucțiunea **uses** `NumeUnit;` (de exemplu, **uses** `Crt,Graph;`).

În *C/C++* același lucru este realizat prin indicarea numelui bibliotecii *header*. Acest lucru se realizează prin `#include <NumeBiblioteca>` (de exemplu, `#include <stdio.h>`).

În *Maple V* pentru a folosi funcțiile dintr-o anumită bibliotecă va trebui să solicităm includerea bibliotecii respective. Această operație se realizează prin:

```
> with(NumeBiblioteca);          sau
> with(NumeBiblioteca):
```

Sintaxa obligă terminarea acestei comenzi cu unul dintre caracterele `";"` sau `":"`.

Noi vom folosi biblioteca matematică `linalg`, deci vom avea un element de forma:

```
> with(linalg); pentru a afișa funcțiile conținute
                          sau
> with(linalg): pentru a inhiba afișarea funcțiilor conținute
```

Observație

Mediul *MAPLE V* este *case-sensitive*, exact ca și *C++* sau *Java*, adică se face diferență între literele mari și cele mici.

Din acest motiv trebuie să fim atenți la scrierea identificatorilor pentru biblioteci și funcții sau proceduri predefinite.

În general, după fiecare acțiune adresată sistemului și terminată prin ";" are loc efectuarea ei cu ecou, adică este afișat pe centrul ecranului rezultatul, chiar dacă facem și o simplă atribuire prin ":=". Dacă terminăm comanda prin ":", ea este efectuată fără ecou, adică nu va fi afișat rezultatul ei.

Ecuațiile pot fi indicate sistemului prin instrucțiuni de atribuire sau pot fi scrise direct în funcția de rezolvare a sistemului. Preferăm scrierea ecuațiilor separat, prin atribuiri de forma:

```
> ecI:=descriere_ecuatie;
```

De exemplu, pentru prima ecuație a sistemului, avem

```
> ec1:=x*(x+y+4*z)-4*z^2=0;
```

Precizăm faptul că simbolul "^" înseamnă "*ridicat la puterea*". Astfel pentru a^m vom scrie "a^m".

Așadar, x^2 înseamnă x^2 și datorită faptului că operatorul de ridicare la putere are prioritate mai mare decât operatorul de înmulțire, nu sunt necesare paranteze.

Utilizăm apoi funcția solve pentru rezolvarea efectivă a sistemului.

Liste

Pentru afișarea eficientă a soluțiilor și selectarea celor reale este util să evidențiem toate soluțiile ecuației prezentate sub forma unei liste.

Lista este un mod de a enumera o serie de elemente într-o ordine bine precizată. În *Maple V* lista este creată prin încadrarea între paranteze drepte sau acolade a unor elemente sau a unei funcții generatoare, formule etc. ce poate să furnizeze mai multe elemente. Forma listei este [secv] sau {secv} unde secv este o secvență de expresii.

Elementele listei sunt separate prin virgulă dacă lista este creată manual. Diferența dintre cele două forme este aceea că, în cazul parantezelor drepte, lista poate conține și elemente identice pe diverse poziții, în timp ce la cea cu acolade toate elementele păstrate în listă după crearea ei sunt distincte.

Vom prezenta mai jos acestă diferență prin crearea, în ambele moduri, a unei liste care conține elemente identice.

```
> {x,y,y};
                        {x, y}
> {y,x,y};
                        {x, y}
> [x,y,y];
                      [x, y, y]
> [y,x,y];
                      [y, x, y]
```

O listă poate fi creată și printr-o enumerare, caz în care se poate folosi tipul subdomeniu $l_i...l_f$, unde l_i este limita inferioară iar l_f este limita superioară.

Va trebui să folosim și funcția seq pentru a crea efectiv enumerarea.

Sintaxa acestei funcții este:

```
seq(exp, ind = li .. lf);
```

unde exp este o expresie iar ind este un *identificator de indice* care parcurge domeniul dintre limita inferioară l_i și limita superioară l_f. Atât limita inferioară cât și cea superioară sunt valori numerice. Iată un exemplu în care prezentăm atât acțiunea cât și efectul.

```
>L := [seq(x[i],i=1..4)];
        L := [x[1], x[2], x[3], x[4]]
```

Vom da acum un exemplu pentru crearea unei liste care va conține pătratele numerelor cuprinse între 10 și 25.

```
>L1:=[seq(i^2,i=10 .. 25)];
 L1:=[100, 121, 144, 169, 196, 225, 256, 289,
 324, 361, 400, 441, 484, 529, 576, 625, 676,
              729, 784, 841, 900]
```

În articolul *Rezolvarea unei ecuații cu coeficienți numere întregi foarte mari* din numărul 11/1 (ianuarie 2001) al ***GInfo*** am precizat faptul că *Maple V* este astfel conceput încât să poată lucra cu numere mari. Iată un exemplu în care putem combina o listă în care reținem pătratele unor numere foarte mari cum ar fi numerele cuprinse între 1.900.047.000 și 1.900.047.005:

```
>L1:=[seq(i^2,i=1900047461..1900047465)];
          L1:=[ 3610180354052546521,
  3610180357852641444, 3610180361652736369,
  3610180365452831296, 3610180369252926225 ]
```

Putem afla la fel de ușor lista valorilor radicalilor de orice ordin pentru numere aflate între anumite limite. Vom prezenta lista radicalilor de ordinul 3 pentru numerele cuprinse între 2 și 10.

Pentru afișarea valorilor efective trebuie să folosim și funcția:

```
evalf(exp, nrc);
```

unde exp este expresia care se evaluează iar nrc este numărul de cifre folosit pentru afișarea valorilor.

Pentru a calcula radicalul de ordinul n al unui număr a vom scrie a^(1/n). Deci, radicalul de ordinul 3 al valorii unei variabile i care parcurge domeniul se va obține cu ajutorul construcției i^(1/3); valorile propriu-zise se determină prin evaluare, deci vom scrie evalf(i^(1/3),7). În acest caz am indicat faptul că dorim o afișare cu 7 cifre zecimale.

```
>L2_0=[seq(i^(1/3),i=2..10)];
  L2_0=[ 21/3, 31/3, 41/3, 51/3, 61/3, 71/3,
              81/3, 91/3, 01/3 ]
>L2:=[seq(evalf(i^(1/3),7),i=2..10)];
 L2:=[1.259921, 1.442250, 1.587401, 1.709976,
    1.817121, 1.912931, 2.000000, 2.080084,
                  2.154435]
```

Listele în care sunt folosite acoladele sunt considerate mulțimi. Când vorbim despre mulțimi ne punem imediat problema operațiilor care pot fi efectuate asupra lor.

Asupra listelor mulțime putem efectua operațiile uzuale de *reuniune*, *intersecție* și *diferență*. Diferența simetrică apare ca o îmbinare a diferenței și reuniunii, cu alte cuvinte, a operațiilor elementare precizate mai sus.

Reuniunea

Pentru reuniune avem funcția `union`. Sintaxa ei este:

```
`union` (m1,m2,...,mp);
```

unde `m1`, `m2`, ..., `mp` sunt mulțimile care se vor reuni.

Ca exemplu, avem:

```
>`union` ({3,4,67,33,5,89},{5,8,33,21,77},
 {3,77,6});
     {8, 3, 4, 5, 6, 21, 33, 67, 77, 89}
```

Observație

Cuvântul cheie `union` trebuie să fie încadrat de caracterul "`" atât în stânga cât și în dreapta. Acest caracter nu este apostroful normal, ci este caracterul care are codul ASCII `96`.

Se poate vedea că `union` este un operator *n*-ar, adică îl putem aplica simultan asupra a *n* operanzi.

În cazul în care folosim doar doi operanzi, forma operatorului poate fi și `A union B` unde `A` și `B` sunt cele două mulțimi care se reunesc. În acest caz `union` nu mai este încadrat de caracterele "`". Iată și un exemplu:

```
> {3,4,5,6} union {3,6,9,12};
                {3,9,12,4,5,6}
```

Intersecția

Pentru intersecție avem funcția `intersect`. Sintaxa ei este:

```
`intersect` (m1,m2,...,mp);
```

unde `m1`, `m2`, ..., `mp` sunt mulțimile care se vor intersecta.

Ca exemplu, avem:

```
>`intersect` ({3,4,7,33},{4,6,7,10,21,33},
 {4,28,33});
                    {4, 33}
```

Observația de mai sus este valabilă și pentru funcția `intersect`.

Se poate vedea că operatorul `intersect` este și el unul *n*-ar, adică îl putem aplica simultan asupra a *n* operanzi.

În cazul în care folosim doar doi operanzi, forma operatorului poate fi și `A intersect B` unde `A` și `B` sunt cele două mulțimi care se intersectează. Iată și un exemplu:

```
> {3,4,5,6} intersect {3,6,9,12};
                    {3,6}
```

Diferența

Pentru diferență avem funcția `minus`. Sintaxa ei este

```
`minus` (m1,m2);
```

unde `m1` și `m2` sunt mulțimile pentru care se va efectua diferența.

Ca exemplu, avem:

```
>`minus` ({3,4,7,33},{4,6,7,21});
                  {3, 33}
```

Se poate vedea că operatorul minus nu este unul *n*-ar, adică nu poate fi aplicat asupra a *n* operanzi.

Forma acestui operator poate fi și `A minus B` unde `A` și `B` sunt cele două mulțimi pentru care se face diferența. Iată și un exemplu:

```
> {3,4,5,6} minus {3,6,9,12};
                    {4,5}
```

Diferența simetrică

Această operație se realizează prin intermediul diferenței și reuniunii, deoarece $A \Delta B = (A \setminus B) \cup (B \setminus A)$.

Astfel, vom combina funcțiile `minus` și `union`. Un exemplu este:

```
>a:={3,4,5,6};
                 a:={3,4,5,6}
>b:={3,6,9,12};
                b:={3,6,9,12}
>dif_1:=a minus b;
                 dif_1:={4,5}
>dif_2:=b minus a;
                dif_2:={9,12}
>dif_sim:=dif_1 union dif_2;
            dif_sim:={9,12,4,5}
```

Numere reale și complexe

Mediul *Maple V* acceptă lucrul atât cu numere reale cât și cu numere complexe în forma `a+b*I`. Se știe că un număr real poate fi privit ca un număr complex cu partea imaginară nulă.

Observație

Partea imaginară este indicată cu ajutorul caracterului "`I`", adică prin literă mare pentru a avea efect în calcule.

Suma, diferența sau produsul a două sau mai multe numere complexe poate fi realizată natural, prin operatorii corespunzători. Avem "`+`" pentru adunare, "`-`" pentru scădere și "`*`" pentru înmulțire. Trebuie menționat faptul că atunci când scriem un număr complex, între partea imaginară și simbolul "`I`", trebuie să apară și operatorul de înmulțire "`*`". Iată și un exemplu:

```
>(2+6*I) + (3-4*I);
                   5 + 2I
>(2+6*I) - (3-4*I);
                  -1 + 10I
>(2+6*I)*(3+4*I)*(6+7*I);
                 -290 + 30I
```

Există operații specifice care pot fi efectuate asupra numerelor complexe. Astfel, putem extrage partea reală și partea imaginară a unei expresii de numere complexe prin utilizarea funcțiilor `Re`, respectiv `Im`. Sintaxa acestora este `Re(exp)`, respectiv `Im(exp)`, unde `exp` este o expresie. Iată și exemplele corespunzătoare:

```
>Re( (2+6*I)*(3+4*I)*(6+7*I) );
                    -290
>Im( (2+6*I)*(3+4*I)*(6+7*I) );
                     30
```

Tot pentru numerele complexe există funcția `evalc`. Aceasta realizează o evaluare simbolică a unei expresii, forțând scrierea în forma standard `a+b*I`. Sintaxa ei este `evalc(exp)`, unde `exp` este o expresie. Un exemplu este prezentat în continuare:

```
> 3^(4+2*I);
                        3(4+2I)
> evalc( 3^(4+2*I) );
     81cos(2 ln(3)) + 81 I sin(2 ln(3))
```

Instrucțiuni

Puterea mediului *Maple V* nu constă doar din numeroase funcții matematice predefinite. Putem defini propriile funcții și putem profita din plin de faptul că *Maple V* este special realizat astfel încât să lucreze cu numere mari. Dar nici acestea nu ar fi suficiente; este nevoie, ca în orice limbaj, de *instrucțiuni*.

Mediul *Maple V* pune la dispoziție instrucțiunea de selecție **if - fi**, precum și instrucțiunea repetitivă **for - while - do - od**. Aceste instrucțiuni simulează instrucțiunile tradiționale `if` și `for`, însă le și adaptează. Trebuie să remarcăm de la bun început că fiecare dintre instrucțiuni are și o secvență de terminare a sa, exact ca în limbajul *FoxPro*.

Instrucțiunea if - fi

Sintaxa acestei instrucțiuni este:

```
if exp_cond
  then instr
| elif exp_cond
    then instr  |
    | else  instr  |
fi
```

unde `exp_cond` este o expresie condițională, iar `instr` este o secvență de instrucțiuni.

Construcția `| |` indică faptul că elementul conținut în interior este opțional. Construcția **elif** este folosită pentru a permite o nouă selecție, în măsura în care cazul anterior nu a fost sufiecient de semnificativ.

O mențiune importantă este aceea că prezența elementului **elif** poate fi repetată ori de câte ori este nevoie. Putem realiza astfel și o selecție multiplă mai eficientă decât o serie de instrucțiuni **if** tradiționale imbricate.

Instrucțiunea for - while - do - od

Sintaxa acestei instrucțiuni este:

```
| for num | | from expr_i | | by expr_inc |
              | to expr_f | | while expr_c |
    do
      instr
    od;
                        sau
| for num | | in expr_i | | while expr_c |
    do
      instr
    od;
```

- `num` este numele variabilei care controlează ciclul;
- `expr_i` este expresia folosită pentru inițializarea variabilei `num`;
- `expr_inc` este expresia folosită pentru incrementarea variabilei `num` la fiecare executare a corpului ciclului;
- `expr_f` este expresia folosită pentru a indica valoarea finală la care trebuie să ajungă variabila `num`;
- `expr_c` este o expresie condițională în urma evaluării căreia se decide dacă trebuie continuat ciclul;
- `instr` este o secvență de instrucțiuni.

Noutatea introdusă și acceptată de *Maple V* constă în faptul că, pe de o parte, se pot face salturi dinamice ale valorilor variabilei în timpul executării ciclului, iar, pe de altă parte, în *Maple V* se creează o legătură mai strânsă între instrucțiunile **for** și **while** tradiționale, îmbinându-le într-o instrucțiune mai puternică.

Prin salturi dinamice ale valorii variabilei în timpul rulării ciclului se înțelege posibilitatea modificării valorii prin care se incrementează de fiecare dată variabila.

În limbajul *Pascal* trecerea variabilei de ciclare la următoarea valoare se făcea strict cu o unitate corespunzătoare tipului ordinal al contorului, dacă nu era precizată explicit o altă incrementare în corpul instrucțiunii **for**. O astfel de incrementare nici nu este recomandabilă.

În limbajul *FoxPro* se admite o incrementare cu un pas oarecare (fix) prin folosirea clauzei **STEP** a instrucțiunii **for - endfor**, dar nu sunt permise salturi dinamice.

În *C/C++* este posibilă incrementarea cu o valoare oarecare și se admit și salturi dinamice.

Vom da un exemplu de program *C++* care realizează salturi dinamice ale valorii de incrementare, lucru care se vede prin afișarea valorilor.

```
# include <stdio.h>
# include <conio.h>

void main(void){
  int i;
  printf("\n");
  for(i=1;i<40;i=2*(i+2))
    printf("%d",i);
  getche();
}
```

Valorile afișate de acest program sunt: `1 6 16 36`. Se observă cum doar prin antetul instrucțiunii **for** se pot modifica în mod dinamic valoarea variabilei `i` care realizează incrementarea. Saltul între două valori este la început `5`, apoi `10` și apoi `20`.

În *Maple V* salturile dinamice sunt realizate prin adăugarea clauzei **by** `expr_inc`. Iată un exemplu:

```
> summ:=0;
> for i from 11 by 2 while i<20 do
>   summ:=summ+I
> od;
                        summ:=0
                       summ:=11
                       summ:=24
                       summ:=39
                       summ:=56
                       summ:=75
```

Se observă că saltul este inițial 13, apoi 15, apoi 17 și apoi 19; deci el este dinamic, și este incrementat cu valoarea lui exp_inc.

Prin acest exemplu, am prezentat o instrucțiune **for - while - do - od** în care sunt prezente atât clauza **by** cât și clauza **while**.

Prin această instrucțiune putem simula atât instrucțiunea **while** tradițională cât și instrucțiunea **for**, prezente în multe limbaje de programare. Acest lucru îl vom folosi în continuare în rezolvarea problemei noastre.

Rezolvarea sistemelor

Vom prezenta acum soluția problemei rezolvării sistemelor prin îmbinarea elementelor prezentate anterior. Nu vom trata strict această problemă, ci metoda de a rezolva astfel de probleme.

Trebuie să mai precizăm că vom prezenta și fișierul **.ms** însoțit de comentariile de rigoare. Un comentariu în *Maple V* se realizează prin prezența caracterului "#" (diez). Comentariul se întinde pe toată linia pe care este prezent caracterul "#".

De asemenea, precizăm faptul că secvențele de instrucțiuni dintr-un fișier pot fi executate secvențial alegând din meniul *Format* opțiunea *Execute Worksheet*. Iată fișierul cu soluția, însoțit de comentariile de rigoare și de răspunsul sistemului *Maple V*:

```
>#  Rezolvarea sistemului (1)

> with(linalg):

> # Ecuațiile sistemului:

> ecc1:=x*(x+y+4*z)-4*z^2=0;
            ecc1:=x(x+y+4z)-4z2=0
> ecc2:=y*(x+y+4*z)-8*z^2=0;
            ecc2:=y(x+y+4z)-8z2=0
> ecc3:=x*y*z=8;
                 ecc3:=xyz=8

> # Soluția finală (se caută valori reale)

> sol_final:={ };
               sol_final:={ }

> # Scriem soluția sistemului sub formă de
> # listă pentru a o putea prelucra pe
> # componente

> sol_vect:=[solve({ecc1,ecc2,ecc3},{x,y,z})];
  sol_vect:=[ { z=RootOf(_Z3-9), y= , x= =},
    {y=-4, z=1, x=-2}, {z=RootOf(_Z3+_Z+1),
              y=-4RootOf(_Z3+_Z+1),
           x=-2 RootOf(_Z3+_Z+1)} ]

> # Caut soluția reală

> sol_vect[2];
              {y=-4, z=1, x=-2}

> # Soluțiile reale le adaug soluției finale

> sol_final:=sol_final union {sol_vect[2]};
       sol_final:={ {y=-4, z=1, x=-2} }

> # Tratez cazul generat prin RootOf (aici am
> # două situații). Mai pot adăuga rădăcini la
> # soluția finală

> # prima situație

> zz1:=[ allvalues(RootOf(z^3-9)) ];
             zz1:=[ 91/3,  ,   ]

> # Tratez un caz mai general; pot avea
> # diverse dimensiuni.
> # Aleg dimensiunea

> dimm:=3;

> # Calculez soluțiile primului caz

> for i from 1 to dimm do
>   sol_exact[i]={ (2/3)*zz1[i],
    (4/3)*zz1[i],  zz1[i] }
> od;
         sol_exact1={ 91/3 ,   ,  }
           sol_exact2={  ,   ,  }
           sol_exact3={  ,   ,  }

> #Folosesc estimări exacte ale soluțiilor

> nr_cifre:=16;
> for i from 1 to dimm do
>   sol_apr[i]:={evalf(Re(evalc((2/3)*
    zz1[i])), nr_cifre) + I *
    evalf(Im(evalc((2/3)*zz1[i])), nr_cifre),
    evalf(Re(evalc((4/3)*zz1[i])), nr_cifre)
    + I * evalf(Im(evalc((4/3)*zz1[i])),
    nr_cifre), evalf(Re(evalc(zz1[i])),
    nr_cifre) + I * evalf(Im(evalc(zz1[i])),
    nr_cifre) }
> od;
       sol_apr1:={ 1.386722548701269,
   2.080083823051904, 2.773445097402538 }
       sol_apr2:={ -.6933612743506346
  +1.200936955176002 I, -1.386722548701269
 +2.401873910352005 I , -1.040041911525952
           +1.801405432764004 I}
      sol_apr3:={ -.6933612743506346 -
  1.200936955176002 I, -1.386722548701269
 - 2.401873910352005 I, -1.040041911525952
           -1.801405432764004 I }
```

```
> # adaug la soluţia finală doar soluţia reală

> sol_final:=sol_final union { sol_aprox[1] };
      sol_final:={ {z=1, y=-4, x=2}, {
    1.386722548701269, 2.080083823051904,
           2.773445097402538 } }

>#  Tratez cazul generat prin RootOf (am două
>#  situaţii). Mai pot adăuga rădăcini la
>#  soluţia finală

>#  a doua situaţie

> zz2:=[allvalues(RootOf(z^2+z+1))];
               zz2:=[   ,    ]

> # Tratez un caz mai general ; pot avea
> # diverse dimensiuni.
> # Aleg dimensiunea

> dimm:=2;

> # Calculez soluţiile cazului al doilea

> for i from 1 to dimm do
>    sol_exact[i]={ (2/3)*zz2[i],
    (4/3)*zz2[i], zz2[i] }
> od;
           sol_exact1:={ ,   ,   }
          sol_exact2:={{ ,   ,   }}

> # Observ că nu am nici o soluţie reală
> # Adaug la soluţia finală doar soluţia reală
> # (este cea vidă)

> sol_final:=sol_final union {   };
      sol_final:={ {z=1, y=-4, x=2}, {
    1.386722548701269, 2.080083823051904,
           2.773445097402538 } }

> # SOLUŢIA FINALĂ:

> sol_final;
  { {z=1, y=-4, x=2}, { 1.386722548701269,
  2.080083823051904, 2.773445097402538 } }
```

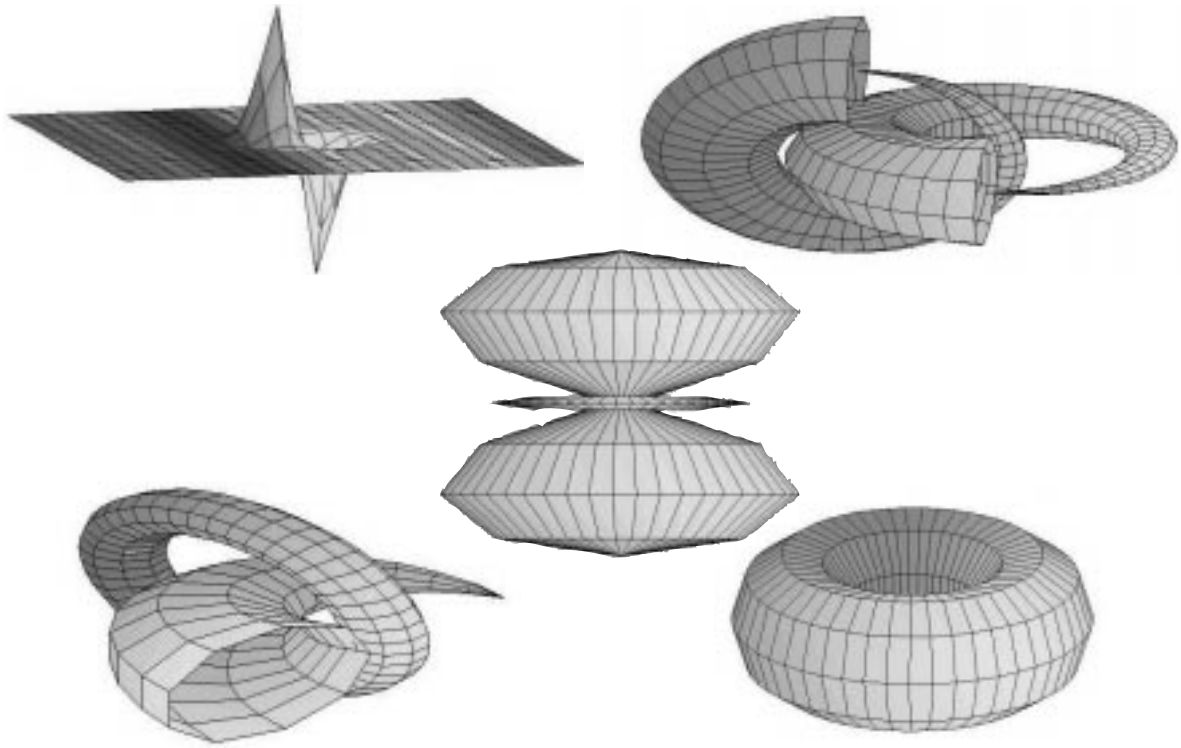
Soluţia obţinută este dată în forma `{z,y,x}` . Ea are valori exacte sau o aproximaţie de `15` zecimale (şi încă se mai pot adăuga), adică suficient de bună.

Putem spune că legătura dintre un limbaj puternic calculatoriu (dar totuşi limbaj) şi problemele sistemice este foarte eficientă.

Bibliografie

1. http://www.maplesoft.com
2. Ion Nanu, Lucian Tuţescu, *Ecuaţii nestandard*, Editura Appolo, Craiova, 1994

Dl. prof. drd. Laviniu Bejenaru este cadru didactic la Colegiul Naţional "Traian" din Drobeta Turnu Severin. Poate fi contactat prin e-mail la adresa bejenaru@traian.expert.ro.



Imagini obţinute cu MAPLE V