УДК 512.5; 519.6;519.8; 372.851; 378.02

БИБЛИОТЕКА ПРОГРАММНЫХ ПРОЦЕДУР ДЛЯ МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ КУРСА ВЫСШЕЙ АЛГЕБРЫ В СИСТЕМЕ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ "MAPLE"

© Ю.Г.Игнатьев, А.Р.Самигуллина

В статье описана библиотека программных процедур для методического обеспечения курса высшей алгебры в пакете компьютерной математики "Maple".

Ключевые слова: линейная алгебра, системы компьютерной математики, алгоритмы, библиотеки программных процедур.

1. Введение

Учебный модуль высшей алгебры является составной и базовой частью курса высшей математики для нематематических факультетов. Основная цель этого модуля – изучение теории линейных алгебраических уравнений, необходимой как в других модулях курса высшей математики и предметах естественнонаучного цикла, так и имеющей самостоятельную ценность для решения многочисленных прикладных задач. Кроме того, в этот модуль включается изучение основ матричного исчисления и теории определителей, необходимых для изучения теории систем линейных алгебраических уравнений (СЛАУ), а также имеющих многочисленные приложения в других модулях курса высшей математики и, следовательно, обладающих самостоятельной ценностью. Поэтому информатизация этого модуля имеет большое значение для изучения курса высшей математики.

В работе А.Р.Самигуллиной были рассмотрены основные принципы математического и компьютерного моделирования объектов линейной алгебры и аналитической геометрии в системе компьютерной математики (СКМ) Maple [1]. В частности, в указанной работе была представлена программа автоматизированного решения системы линейных алгебраических уравнений в СКМ Maple с выводом решений в стандартной для Российской системы образования форме. Однако описанная программная процедура обладает существенным недостатком: при наличии нулевых коэффициентов перед неизвестными эти переменные не считываются и соответствующие нули не попадают в расширенную матрицу системы. Кроме того, встроенные программные процедуры Maple не содержат программы нахождения фундаментальных решений СЛАУ, имеющих большую ценность в многочисленных приложениях и составляющих основу математической культуры.

Решение указанных двух задач программными способами в СКМ Maple потребовало значительного усложнения программных процедур. Здесь следует отметить общее правило создания программных продуктов: чем менее профессиональным является пользователь программного продукта, тем большая степень программного сервиса требуется от этого продукта, а значит, и более сложные соответствующие программные процедуры. В данной статье мы укажем пути решения этих задач и опишем библиотеку соответствующих программных процедур *Algebra* .

2. Программные процедуры нахождения общего решения систем линейных алгебраических уравнений

Рассмотрим систему алгебраических линейных уравнений из m уравнений относительно n неизвестных:

$$\mathbf{AX} = \mathbf{B} \Rightarrow \sum_{i=1}^{n} a_{ki} x_i = b_k; \quad \left(k = \overline{1,m}\right), \qquad (1)$$

где $\mathbf{A}$ – основная матрица системы, $\mathbf{B}$ – матрица-столбец свободных членов, $\mathbf{X}$ – матрица-столбец неизвестных:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}; \quad \mathbf{B} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}; \quad \mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}. \qquad (2)$$

Пользователь (студент-нематематик) вводит указанную систему в Maple не в матричной, а в стандартной форме упорядоченного списка уравнений:

$$\begin{aligned} \mathrm{Sys} := [&\alpha_1 * x + \alpha_2 * y + \dots + \alpha_n * u = b_1, \\ &\beta_1 * x + \beta_2 * y + \dots + \beta_n * u = b_2, \\ &\dots, \delta_1 * x + \delta_2 * y + \dots + \delta_n * u = b_m], \end{aligned} \qquad (3)$$

где $\alpha_i, \beta_i, \dots, \delta_i, b_j$ – коэффициенты уравнений (1), т.е. конкретные числа, а $x, y, \dots, u$ – имена переменных, которые в конкретном задании могут быть произвольными. Поэтому первой задачей является компьютерное распознавание системы

(3) и приведение ее к стандартному виду (1), а также нахождение расширенной матрицы системы:

$$\tilde{A}=\left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right) \qquad (4)$$

2.1. Распознавание системы линейных алгебраических уравнений

Для распознавания системы (3) библиотека *Algebra* имеет программную процедуру *Algebra[InfoEq]*, которая предоставляет информацию об одиночном линейном уравнении в виде упорядоченного списка, первый элемент которого есть упорядоченный список имен переменных уравнения, второй элемент – упорядоченный список коэффициентов при этих переменных, третий элемент – правую часть уравнения:

```
> Algebra[InfoEq]:=proc(Eq) local n,i,xxx:
n:=Algebra[number_members](Eq):
xxx:=(i)->Algebra[coef_var](Eq,i):
[[seq(xxx(i)[2],i=1..n)],[seq(xxx(i)[1],i=1..n)],
rhs(Eq)]:end proc:
```

Здесь использована процедура *number_coef*, которая находит коэффициент у *i*-той переменной, причем если коэффициент равен 1, т.е. множитель перед переменной отсутствует, то результат действия программы будет "1", в других ненулевых случаях результат равен "2":

```
>Algebra[number_coef]:=(Eq,i)->
nops([op([op(lhs(Eq))])][i]):
```

Пример:

```
>Algebra[number_coef](3*x+2*y+z=23,3);
```

1

Покажем пример распознавания одиночного линейного алгебраического уравнения с помощью созданной процедуры

```
> Algebra[InfoEq](3*x+2*y+z=23);
```

$$[[x,y,z],[3,2,1],23]$$

Далее, процедура *associate* находит объединение подмножеств x, представленных в виде неупорядоченного списка неупорядоченных списков:

$$x=\{\{a\_1,\ldots,a\_r\},\{b\_1,b\_2,\ldots,b\_s\},\ldots,\{c\_1,\ldots,c\_m\}\}.$$

При этом результат записывается в виде упорядочного списка:

```
> Algebra[associate]:=proc(x) local elem,u:
u:=x[1]: for elem in x do u:=u union elem:
end do:[op(u)]: end proc:
```

Пример:

```
> Algebra[associate]([{x,y,z},{y,u,z},{y,t,w},
{r,u,s}]);
```

$$[r,u,y,z,x,t,w,s]$$

2.2. Приведение СЛАУ к стандартному виду

Процедура *StandartSys* выводит СЛАУ в стандартном виде с унифицированными именами переменных X_i, где $i=\overline{1\ldots nnn}$:

```
>Algebra[StandartSys]:=proc(Eqs)local
nn,Eq,i,vars,Vars,
nnn,k,SB,EQS:
nn:=nops(Eqs):Eq:=(i)->Eqs[i]:
vars:=(i)->{op(Algebra[InfoEq](Eq(i))[1])}:
Vars:=Algebra[associate]({seq(vars(i),i=1..nn)}
):
nnn:=nops(Vars):SB:={seq(Vars[k]=X[k],k=1..
nnn)}:
EQS:=subs(SB,Eqs):EQS:end proc:
```

Пример:

```
>Algebra[StandartSys]([x+2*y-z=5,x+y-
3*z=7,5*x-3*y+2*z=9,x+y+z=1]);
```

$$[X_1+2X_2-X_3=5,\ X_1+X_2-3X_3=7,\ 5X_1-3X_2+2X_3=9,\ X_1+X_2+X_3=1].$$

2.3. Программа нахождения основной и расширенной матрицы системы

Процедура *Algebra[MatrSys]* создает упорядоченный список, состоящий из двух матриц – основной матрицы системы, A, и расширенной, $\tilde{A}$:

```
>Algebra[MatrSys]:=proc(Eqs)local
nn,Eq,i,vars,
Vars,nnn,k,SB,EQS,aa,bb,AA,AB:
nn:=nops(Eqs): Eq:=(i)->Eqs[i]:
vars:=(i)->{op(Algebra[InfoEq](Eq(i))[1])}:
Vars:=Algebra[associate]({seq(vars(i),i=1..nn)}):
nnn:=nops(Vars):SB:={seq(Vars[k]=X[k],k=1..
nnn)}:
EQS:=subs(SB,Eqs):aa:=(i,k)-
>coeff(lhs(EQS[i]),X[k]):
bb:=(i)-
>rhs(EQS[i]):AA:=convert([seq([seq(aa(i,k),
k=1..nnn)],i=1..nn)],Matrix):
AB:=convert([seq([seq(aa(i,k),k=1..nnn),bb(i)],
i=1..nn)],Matrix):[AA,AB]:end proc:
```

Пример:

```
> Algebra[MatrSys]([x+2*y-z=5, y-3*z=7,5*x-
3*y+2*z=9]);
```

$$\left[\begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & -3 \\ 5 & -3 & 2 \end{bmatrix},\begin{bmatrix} 1 & 2 & -1 & 5 \\ 0 & 1 & -3 & 7 \\ 5 & -3 & 2 & 9 \end{bmatrix}\right]$$

2.4. Программа нахождения общего решения в стандартном виде

Программная процедура *Algebra[SolLinGen]* находит общее решение системы линейных алгебраических уравнений, причем значению параметра *t=matr* соответствует вывод решения системы в матричном виде, в котором произвольные константы выводятся в формате S_i, где i – номер строки, содержащей только эту константу с коэффициентом 1. В случае если система не совместна, то вместо решения команда выводит сообщение о несовместности. При любом другом значении параметра t, например, 123, – решение выводится в списочном виде: x=x1, y=y1,…:

```
> Algebra[SolLinGen]:=proc(Eqs,t) local
nn,Eq,i,vars,Vars,nnn,k,SB,
EQS,aa,bb,AA,AB,r1,r2,BB,SSS,C,RRR:
nn:=nops(Eqs):Eq:=(i)->Eqs[i]:
vars:=(i)->{op(Algebra[InfoEq](Eq(i))[1])}:
Vars:=Algebra[associate]({seq(vars(i),i=1..nn)}):
nnn:=nops(Vars):SB:={seq(Vars[k]=X[k],k=1..
nnn)}:
EQS:=subs(SB,Eqs):aa:=(i,k)-
>coeff(lhs(EQS[i]),X[k]):
bb:=(i)-
>rhs(EQS[i]):AA:=convert([seq([seq(aa(i,k),
k=1..nnn)], i=1..nn)],Matrix):
AB:=convert([seq([seq(aa(i,k),k=1..nnn),bb(i)],
i=1..nn)],Matrix):BB:=Vector([seq(bb(i),i=1..nn)]):
r1:=linalg[rank](AA):r2:=linalg[rank](AB):
if r1=r2 then
SSS:=LinearAlgebra[LinearSolve](AA,BB,free=
`S`):
RRR:=seq(Vars[i]=SSS[i],i=1..nnn):
else ("Система не совместна") end if:
if t=matr then SB,convert(SSS,Matrix):
else RRR: end if: end proc:
```

Пример 1:

```
> Eq1:=[x+2*y-z=5, x+y-3*z=7,5*x-
3*y+2*z=9,y+2*z=-2]:
> Algebra[SolLinGen](Eq1,a);
```

$$x=\frac{29}{11},\ y=\frac{6}{11},\ z=\frac{-14}{11}.$$

Пример 2:

```
> Eq2:=[x-3*y+10*z-t+u=1,x+y+3*z-t-u=3,x-
2*y+2*t+2*u=1]:
> SSEq2:=Algebra[SolLinGen](Eq2,matr);
```

$$SSEq2 := \{x = X_1, y = X_2, z = X_3, u = X_4, t = X_5\},$$

$$\begin{bmatrix} \frac{7}{3}-2S_3 \\ \frac{1}{3}-S_5+\frac{9}{2}S_3 \\ S_3 \\ -\frac{1}{3}-2S_5+\frac{11}{2}S_3 \\ S_5 \end{bmatrix}. \qquad (5)$$

3. Программные процедуры нахождения фундаментального решения систем линейных алгебраических уравнений

Полученное общее решение в форме (5) дает возможность найти фундаментальное решение СЛАУ. Для того чтобы воспользоваться этим решением, необходимо, во-первых, найти в матрице типа (5) все произвольные константы S_i, во-вторых, определить их номера положения в матрице – столбце, в-третьих, образовать последовательности решений Z_i, полученных из общего решения вида (5) подстановкой: $S_k = \delta_k^i$. Для достижения этого результата в библиотеке *Algebra* имеются две команды: *Finder* и *List_Num*. Команда *Finder(s,a)* отыскивает элемент *a* в списке *s* и определяет его положение; в случае, если элемент *a* не содержится в списке *s*, команда выводит пустой список.

```
> Algebra[Finder]:= proc(s,a) local ss,i,j,aa:
aa:=convert(a,symbol):
ss:=[]:
for i from 1 to LinearAlgebra[RowDimension](s)
do
for j from 1 to LinearAlge-
bra[ColumnDimension](s) do
if(convert(s[i,j],symbol)=aa) then
ss:=[op(ss),[i,j]]:
end if: end do: end do:
ss: end proc:
```

Пример:

```
> Algebra[Finder](SSEq2Matr,S[5]);
[[5,1]]
> Algebra[Finder](SSEq2Matr,S[3]);
[[3,1]]
> Algebra[Finder](SSEq2Matr,S[6]);
[],
```

где *SSEq2Matr* – матрица в правой части (5). Далее, процедура *List_Num* определяет положение элементов в матрице–столбце и записывает это положение в виде упорядоченного списка:

```
> Algebra[List_Num]:=proc(A) local i,ss,nn:
nn:=nops(A):ss:=[]:
```

```
for i from 1 to nn do
if A[i]=1 then ss:=[op(ss),i]:end if:
end do: ss: end proc:
```

Наконец, процедура *Algebra[SolLinBasic]* выводит фундаментальное решение в виде матрицы, каждый столбец которой представляет линейно независимое решение:

```
>Algebra[SolLinBasic]:=proc(Eqs) local
nn,Eq,i,vars,Vars,
nnn,k,SB,EQS,aa,bb,AA,AB,r1,r2,BB,SSS,S,RRR,SSS
M,nFC,NC,mmm,j,ii,FSJ,FS:nn:=nops(Eqs):
Eq:=(i)->Eqs[i]:
vars:=(i)->{op(Algebra[InfoEq](Eq(i))[1])}:
Vars:=Algebra[associate]({seq(vars(i),i=1..nn)}):
nnn:=nops(Vars):
SB:={seq(Vars[k]=X[k],k=1..nnn)}:
EQS:=subs(SB,Eqs): aa:=(i,k)-
>coeff(lhs(EQS[i]),X[k]):
bb:=(i)->rhs(EQS[i]):
AA:=convert([seq([seq(aa(i,k),
k=1..nnn)],i=1..nn)],Matrix):
AB:=convert([seq([seq(aa(i,k),k=1..nnn),bb(i)],i
=1..nn)],Matrix):BB:=Vector([seq(bb(i),i=1..nn)]):
r1:=linalg[rank](AA):r2:=linalg[rank](AB):
if r1=r2 then
SSS:=LinearAlgebra[LinearSolve](AA,BB,free=
`S`):
SSSM:=convert(SSS,Matrix):
nFC:=[seq(nops(Algebra[Finder](SSSM,S[i])),i
=1..nnn)]:
NC:=Algebra[List_Num](nFC):mmm:=nops(N
C): ii:=(j)->NC[j]:
FSJ:=(j)-
>subs({seq(S[ii(k)]=delta(k,j),k=1..mmm)},SSSM):
FS:=convert([seq(FSJ(j),j=1..mmm)],Matrix):
else ("Система не совместна"):end if: FS: end
proc:
```

Пример применения процедуры мы не приводим из-за громоздкости ответа. Для вывода фундаментального решения в стандартном для вузов списочном виде библиотека *Algebra* содержит специальную процедуру *Algebra[SolLinBasics]:*

```
> Algebra[SolLinBasics]:=proc(Eqs)local
mmm,nnn,i,k,SSSS:
SSSS:=Algebra[SolLinBasic](Eqs):
mmm:=LinearAlgebra[RowDimension](SSSS):
nnn:=LinearAlgebra[ColumnDimension](SSSS):
[seq([seq(SSSS[i,k],i=1..mmm)],k=1..nnn)]:
end proc:
```

Пример:

Рассмотрим систему уравнений *Eq3*:

```
>Eq3:=[x+2*y-3*z+u-6*t+8*n=1,x-y+2*z-t-
n=0,2*x+y-z+3*u-t=1]:
```

Ее фундаментальное решение в стандартном, списочном виде дается командой:

```
> Algebra[SolLinBasics](Eq3);
```

$$\left[\left[0,2,1,0,0,0\right],\left[\frac{-17}{6},\frac{-23}{6},0,1,\frac{7}{2},0\right],\left[4,3,0,0,-3,1\right]\right]$$

Заключение

Созданная библиотека процедур приспособлена для студентов нематематических факультетов, удобна и проста в работе. Она может быть использована и преподавателями для проверки работ студентов, а также для генерации заданий.

* * * * * * * * * *

1. *Самигуллина А.Р.* Математическое моделирование объектов линейной алгебры и аналитической геометрии в системе компьютерной математики Maple // Вестник ТГГПУ. – 2010. – №3(21). – С.69-74.

THE LIBRARY OF PROGRAMME PROCEDURES FOR METHODICAL SUPPORT OF LINEAR ALGEBRA COURSE IN A COMPUTER MATHEMATICS SYSTEM "MAPLE"

Yu.G.Ignatyev, A.R.Samigullina

The library of programme procedures for methodical support of Linear Algebra course in a computer Mathematics system "Maple" is described.

Key words: Linear Algebra, computers Mathematic systems, algorithm, libraries of programme procedures.

* * * * * * * * * *

Игнатьев Юрий Геннадиевич – доктор физико-математических наук, профессор, заведующий кафедрой геометрии и математического моделирования Татарского государственного гуманитарно-педагогического университета.

E-mail: ignatev_yu@rambler.ru

Самигуллина Алсу Ринатовна – аспирант кафедры геометрии и математического моделирования Татарского государственного гуманитарно-педагогического университета.

E-mail: alsu_sam@mail.ru

Поступила в редакцию 21.12.2010