СЕКЦИЯ 1.
MATLAB — среда разработки инженерных и научных приложений

Часть 2.

Председатель:
к. т. н., доцент В. Г. Потемкин

Оглавление

УДК 512. 54

К ВОПРОСУ О ВЫЧИСЛЕНИИ ЭЛЕМЕНТОВ В СВОБОДНЫХ БЕРНСАЙДОВСКИХ ГРУППАХ

Кузнецов А. А., Шлёпкин А. К.
Красноярский государственный аграрный университет, Красноярск,
e-mail: alex_kuznetsov80@mail.ru

Слово в алфавите Y — это пустая (обозначается *E*) или конечная последовательность символов из *Y* (*Y* — фиксированное множество символов). Число элементов этой последовательности называется *длиной* слова. Через $\partial(A)$ будем обозначать длину слова *A*, в частности $\partial(E)=0$. Говорят, что слово *X входит* в слово *A*, если можно указать такие слова *R* и *Q*, что $A=RXQ$. Если при этом слово *R* (слово *Q*) пусто, то говорят, что *X* есть *начало* (*конец*) слова *A*.

Слово *A* будем называть *n-апериодическим*, если в него не входит никакое непустое слово вида X^n, т. е. $A \neq R(\underbrace{XX...X}_{n})Q$.

Таким образом, если слово $A=R(\underbrace{XX...X}_{n})Q$, то *A* сокращается на X^n и $A \rightarrow A=RQ$. Далее, если *A* включает в себя какое-либо непустое слово вида X^n, то *A* опять сокращается на X^n. Этот процесс будет продолжаться до тех пор, пока в слове *A* не будет содержатся ни одного вхождения типа X^n, т. е. $A \rightarrow B$.

Построим свободную группу Бернсайда $B(2,3)$ в алфавите $Y=\{0, 1\}$. Для этого сначала опишем вспомогательный алгоритм (см. блок-схему 1) преобразования в алфавите *Y* произвольного слова $A=a_1a_2...a_l$ конечной длины $\partial(A)=l$ $a_i=0, 1$ ($a_i \in A$, $i=1, 2, ..., l$) в 3-апериодическое слово *B* и обозначим его через F_1.

1. Задаем исходное слово *A*.
2. Вычисляем длину *A*: $\partial(A)=l$.

а) Если $l<3$, то слово *A* заведомо является 3-апериодическим, следовательно $A=B$. Алгоритм завершен.

b) Если $l \geq 3$, то тогда задаем тройку последовательно идущих в *A* слов X_1, X_2 и X_3 одинаковой длины ($\partial(X_1)=\partial(X_2)=\partial(X_3)=j$), т. е. $A=R(X_1X_2X_3)Q$. Длину и месторасположение этих слов в слове *A* можно задать исчерпывающим образом посредством двух параметров *i* и *j*, где:

$j=1,2,...,k$ ($k=[l/3]$ — целая часть от отношения $l/3$) — параметр, задающий длину слов $X_{1, 2, 3}$;

$i=1,2,...,n=(l-3j+1)$ — параметр, задающий месторасположение слов $E_{1, 2,3}$ в слове *A*. Таким образом:

$X_1(i, j)=a_i...a_{i+(j-1)}$, $X_2(i, j)=a_{i+(j-1)+1}...a_{i+2(j-1)+1}$, и $X_3(i, j)=a_{i+2(j-1)+2}...a_{i+3(j-1)+2}$.

3. Задаем начальное значение длин слов $X_{1,2,3}$: $j=1$.

4. Задаем начальное месторасположение слов $X_{1,2,3}$: $i=1$ (при $i=1$ слова $X_{1,2,3}$ размещены в начале слова A, т. е. $A=(X_1X_2X_3)P$).

5. Сравниваем слова $X_1(i,j)$, $X_2(i,j)$ и $X_3(i,j)$.

a) Если $X_1=X_2=X_3$ (*), то слово A сокращается и алгоритм начинается заново, при этом в качестве исходного выступает сокращенное слово A (см. пункт 1).

b) Если (*) не выполняется, делаем проверку неравенства $i<n$ (**).

6. a) Если (**) выполняется, то увеличиваем значение параметра i на единицу $i=i+1$, смещая тем самым слова X_1, X_2 и X_3 на один индекс вправо вдоль A. Затем возвращаемся к пункту 5.

b) Если (**) не выполняется, делаем проверку неравенства $j<k$ (***).

7. a) Если (***) выполняется, то увеличиваем значение параметра j на единицу $j=j+1$, увеличивая тем самым длину слов X_1, X_2 и X_3 на единицу. Затем возвращаемся к пункту 4.

b) Если (***) не выполняется, то это означает, что слово A 3-апериодическое, т. е. $A\rightarrow B$. Алгоритм завершен.

Следует отметить, что данный алгоритм не единственный. Покажем это на примере.

Пусть $A_1=11101010$. Тогда, преобразовывая данное слово по алгоритму F_1, описанному выше, получим $A_1\rightarrow B_1=01010$. С другой стороны, сначала в слове A_1 можно сократить слова длины два, а затем перейти к реализации алгоритма F_1 (обозначим этот алгоритм через F_2). Преобразовывая слово A_1 посредством алгоритма F_2, получим: $A_1=11(101010)\rightarrow B_2=11$, т. е. в этом случае получено другое 3-апериодического слово.

Далее, пусть $A_2=1100110011101010$. Тогда преобразовывая данное слово по алгоритму F_1, получим $A_2\rightarrow B_1=0$. Преобразовывая же слово A_2 по алгоритму F_2, получим $A_2=1100110011(101010)\rightarrow B_2=1100110011$.

Пример, приведенный выше, показывает, что вид 3-апериодического слова зависит от выбора алгоритма.

Далее применяем алгоритм F_1 для вычисления элементов свободной группы Бернсайда $B(2,3)$ в алфавите $Y=\{0, 1\}$ [1]. Ниже приведены элементы этой группы (табл. 1), таблица умножения Кэли (табл. 2), а также тождества, получаемые при построении таблицы умножения (табл. 3).

Все результаты были получены в системе компьютерной математики MATLAB 6.5 на компьютере INTEL Pentium IV 2,4GHz(800MHz), ОЗУ 512 Mb.

Из алгоритма F_1, нетрудно получить алгоритм преобразования произвольного слова в 4-апериодическое слово с последующим построением группы $B(2,4)$. В табл. 4 представлена статистика элементов $B(2,4)$ до слов длины 11 включительно. В табл. 5 показаны тождества, получаемые при построении таблицы умножения Кэли. Для дальнейшей реализации метода

требуется более «мощный» компьютер, либо усовершенствование имеющегося алгоритма.

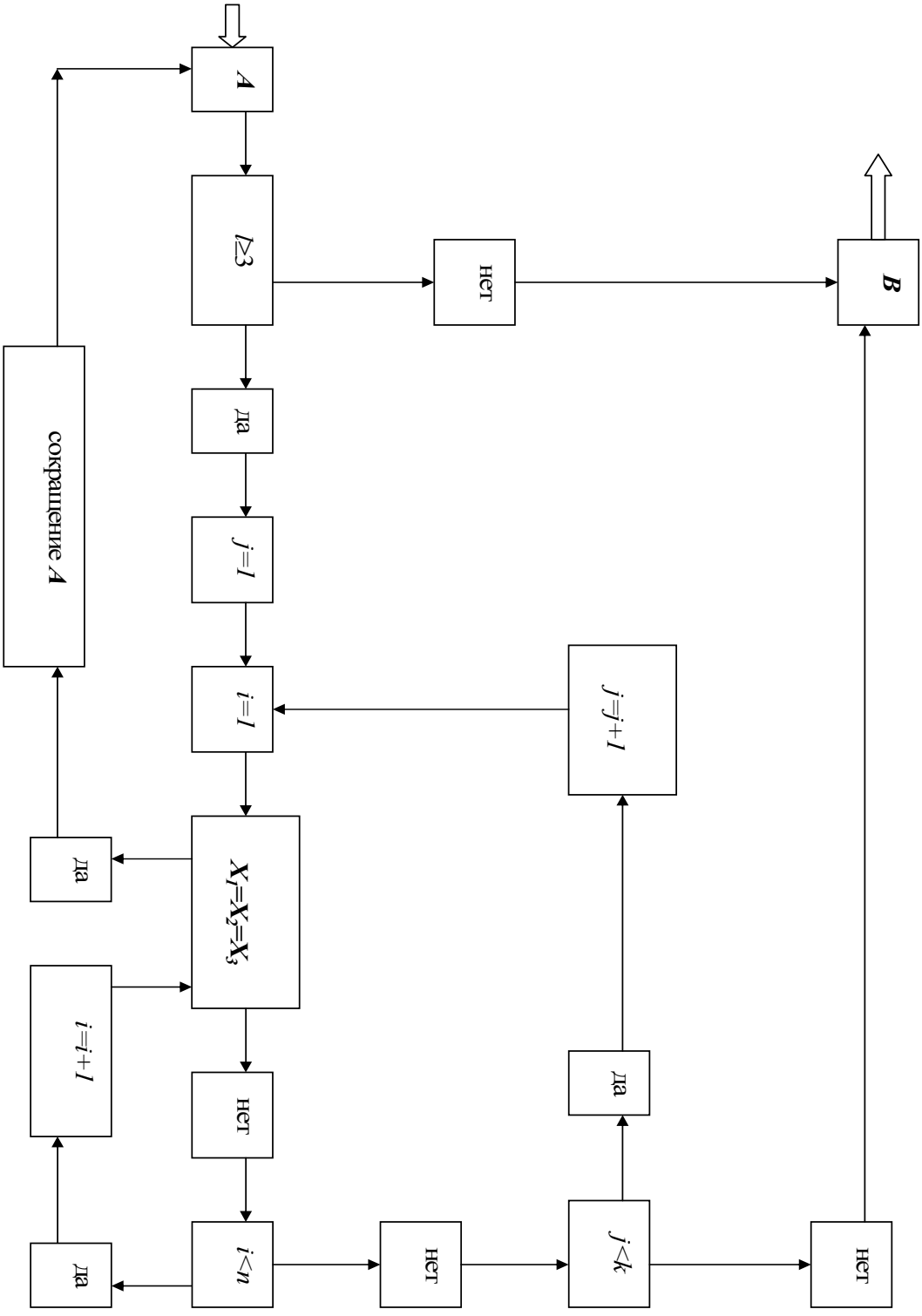


Рис. 1. Блок-схема 1. Алгоритм F_1 преобразования произвольного слова A в 3-апериодическое слово B.

Таблица 1.
Элементы свободной группы Бернсайда $B(2, 3)$ в алфавите Y, расположенные в алфавитном порядке.

№	**Вид элемента в алфавите *Y***	**№**	**Вид элемента в алфавите *Y***	**№**	**Вид элемента в алфавите *Y***
b_1	*E*	b_{10}	*110*	b_{19}	*1101*
b_2	*0*	b_{11}	*001*	b_{20}	*1011*
b_3	*1*	b_{12}	*101*	b_{21}	*00100*
b_4	*00*	b_{13}	*011*	b_{22}	*10100*
b_5	*10*	b_{14}	*0100*	b_{23}	*01100*
b_6	*01*	b_{15}	*1100*	b_{24}	*11010*
b_7	*11*	b_{16}	*0010*	b_{25}	*10110*
b_8	*100*	b_{17}	*1010*	b_{26}	*110100*
b_9	*010*	b_{18}	*0110*	b_{27}	*101100*

Таблица 2.

Таблица умножения Кэли для свободной группы Бернсайда $B(2, 3)$.

	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$	$\boldsymbol{b_3}$	$\boldsymbol{b_4}$	$\boldsymbol{b_5}$	$\boldsymbol{b_6}$	$\boldsymbol{b_7}$	$\boldsymbol{b_8}$	$\boldsymbol{b_9}$	$\boldsymbol{b_{10}}$	$\boldsymbol{b_{11}}$	$\boldsymbol{b_{12}}$	$\boldsymbol{b_{13}}$	$\boldsymbol{b_{14}}$	$\boldsymbol{b_{15}}$	$\boldsymbol{b_{16}}$	$\boldsymbol{b_{17}}$	$\boldsymbol{b_{18}}$	$\boldsymbol{b_{19}}$	$\boldsymbol{b_{20}}$	$\boldsymbol{b_{21}}$	$\boldsymbol{b_{22}}$	$\boldsymbol{b_{23}}$	$\boldsymbol{b_{24}}$	$\boldsymbol{b_{25}}$	$\boldsymbol{b_{26}}$	$\boldsymbol{b_{27}}$
$\boldsymbol{b_1}$	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}
$\boldsymbol{b_2}$	b_2	b_4	b_6	b_1	b_8	b_9	b_{11}	b_{10}	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	b_{29}	b_{30}
$\boldsymbol{b_3}$	b_3	b_5	b_7	b_8	b_{10}	b_{12}	b_{11}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}	b_1	b_{18}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	b_{29}	b_{30}	b_{31}
$\boldsymbol{b_4}$	b_4	b_1	b_{11}	b_2	b_{16}	b_{10}	b_3	b_{17}	b_{21}	b_5	b_{22}	b_{23}	b_4	b_{24}	b_6	b_{25}	b_7	b_{26}	b_8	b_{27}	b_9	b_{28}	b_{29}	b_{30}	b_{31}	b_{32}	b_{33}
$\boldsymbol{b_5}$	b_5	b_8	b_{12}	b_3	b_{17}	b_{18}	b_{15}	b_2	b_{19}	b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	b_{29}	b_{30}	b_{31}	b_{32}	b_{33}
$\boldsymbol{b_6}$	b_6	b_9	b_{13}	b_{14}	b_{18}	b_{17}	b_2	b_{15}	b_{16}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	b_{29}	b_{30}	b_{31}	b_{32}	b_{33}	b_{34}	b_{35}	b_{36}
$\boldsymbol{b_7}$	b_7	b_{10}	b_1	b_{15}	b_2	b_{19}	b_{15}	b_{18}	b_3	b_{20}	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}	b_{28}	b_{29}	b_{30}	b_{31}	b_{32}	b_{33}	b_{34}	b_{35}	b_{36}	b_{37}
$\boldsymbol{b_8}$	b_8	b_3	b_{18}	b_5	b_{23}	b_7	b_{24}	b_6	b_{22}	b_{25}	b_4	b_{26}	b_{27}	b_{28}	b_{29}	b_{30}	b_{31}	b_{32}	b_{33}	b_{34}	b_{35}	b_{36}	b_{37}	b_{38}	b_{39}	b_{40}	b_{41}
$\boldsymbol{b_9}$	b_9	b_{14}	b_5	b_6	b_7	b_{24}	b_{22}	b_{25}	b_4	b_{23}	b_{22}	b_{25}	b_4	b_{23}	b_{22}	b_{25}	b_4	b_{23}	b_{22}	b_{25}	b_4	b_{23}	b_{22}	b_{25}	b_4	b_{23}	b_{22}
$\boldsymbol{b_{10}}$	b_{10}	b_{15}	b_{19}	b_7	b_{24}	b_{25}	b_{21}	b_{26}	b_{27}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}
$\boldsymbol{b_{11}}$	b_{11}	b_{16}	b_{17}	b_{21}	b_{22}	b_{23}	b_4	b_{12}	b_{13}	b_1	b_{10}	b_{26}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}	b_{12}	b_{11}	b_{27}
$\boldsymbol{b_{12}}$	b_{12}	b_{17}	b_{20}	b_{22}	b_{25}	b_4	b_5	b_{27}	b_1	b_{13}	b_1	b_8	b_{26}	b_9	b_{11}	b_2	b_3	b_{16}	b_{20}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}	b_{12}	b_{11}
$\boldsymbol{b_{13}}$	b_{13}	b_{18}	b_2	b_{23}	b_4	b_{24}	b_6	b_1	b_{26}	b_9	b_8	b_{26}	b_9	b_{11}	b_2	b_3	b_{16}	b_{20}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}	b_{12}	b_{11}	b_{27}
$\boldsymbol{b_{14}}$	b_{14}	b_6	b_{22}	b_9	b_{12}	b_{13}	b_{26}	b_{17}	b_{20}	b_2	b_{23}	b_4	b_{24}	b_6	b_1	b_{26}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}	b_{12}	b_{11}	b_{27}	b_1	b_{10}
$\boldsymbol{b_{15}}$	b_{15}	b_7	b_{25}	b_{10}	b_{27}	b_1	b_9	b_{20}	b_2	b_{14}	b_{19}	b_{15}	b_{20}	b_{16}	b_{19}	b_{15}	b_{20}	b_{16}	b_{19}	b_{15}	b_{20}	b_{16}	b_{19}	b_{15}	b_{20}	b_{16}	b_{19}
$\boldsymbol{b_{16}}$	b_{16}	b_{21}	b_{23}	b_{11}	b_{13}	b_{27}	b_{18}	b_2	b_{15}	b_{20}	b_{17}	b_2	b_{19}	b_3	b_4	b_{24}	b_7	b_{25}	b_{22}	b_5	b_6	b_{24}	b_7	b_{25}	b_{22}	b_5	b_6
$\boldsymbol{b_{17}}$	b_{17}	b_{22}	b_4	b_{12}	b_1	b_{26}	b_{11}	b_{27}	b_8	b_{10}	b_{14}	b_{18}	b_{17}	b_2	b_{19}	b_{16}	b_{20}	b_3	b_{14}	b_{19}	b_2	b_{18}	b_{17}	b_2	b_{19}	b_{16}	b_{20}
$\boldsymbol{b_{18}}$	b_{18}	b_{23}	b_{24}	b_{13}	b_{26}	b_{27}	b_8	b_{19}	b_{20}	b_3	b_2	b_{14}	b_{16}	b_{25}	b_5	b_7	b_4	b_6	b_{21}	b_{22}	b_9	b_{11}	b_{10}	b_{13}	b_{17}	b_{18}	b_{12}
$\boldsymbol{b_{19}}$	b_{19}	b_{24}	b_{21}	b_{26}	b_{11}	b_8	b_{10}	b_{16}	b_3	b_{15}	b_{17}	b_{18}	b_5	b_7	b_6	b_{22}	b_4	b_{12}	b_{13}	b_{27}	b_1	b_{11}	b_{26}	b_{12}	b_{27}	b_{13}	b_{14}
$\boldsymbol{b_{20}}$	b_{20}	b_{25}	b_5	b_{27}	b_8	b_9	b_{12}	b_3	b_{14}	b_{24}	b_6	b_{22}	b_9	b_{25}	b_{21}	b_{13}	b_{27}	b_9	b_{11}	b_{26}	b_{12}	b_{10}	b_{15}	b_{17}	b_{18}	b_{19}	b_{16}
$\boldsymbol{b_{21}}$	b_{21}	b_{11}	b_{10}	b_{16}	b_{15}	b_{17}	b_{19}	b_7	b_{22}	b_{21}	b_{13}	b_{27}	b_9	b_1	b_{11}	b_{26}	b_{12}	b_{10}	b_{15}	b_{17}	b_{18}	b_{14}	b_3	b_{16}	b_{20}	b_{23}	b_{24}	b_{25}
$\boldsymbol{b_{22}}$	b_{22}	b_{12}	b_{26}	b_{17}	b_{19}	b_{20}	b_{14}	b_{24}	b_{25}	b_6	b_{23}	b_{27}	b_8	b_{12}	b_{10}	b_{27}	b_1	b_{13}	b_{11}	b_{26}	b_{12}	b_{10}	b_{15}	b_{17}	b_{18}	b_{19}	b_{16}
$\boldsymbol{b_{23}}$	b_{23}	b_{22}	b_{15}	b_6	b_{21}	b_6	b_{23}	b_{21}	b_7	b_{23}	b_{21}	b_7	b_{22}	b_{23}	b_{11}	b_{10}	b_{13}	b_{11}	b_{10}	b_{12}	b_{27}	b_1	b_{26}	b_8	b_9	b_{13}	b_{27}
$\boldsymbol{b_{24}}$	b_{24}	b_{26}	b_8	b_{19}	b_3	b_{14}	b_{18}	b_5	b_6	b_{21}	b_7	b_{22}	b_{12}	b_{27}	b_1	b_{26}	b_{12}	b_{11}	b_{10}	b_{12}	b_{27}	b_1	b_{26}	b_{12}	b_{11}	b_{27}	b_1
$\boldsymbol{b_{25}}$	b_{25}	b_{27}	b_9	b_{20}	b_{14}	b_{16}	b_{15}	b_6	b_{21}	b_5	b_{22}	b_{23}	b_{11}	b_{10}	b_{13}	b_{11}	b_{10}	b_{12}	b_{27}	b_1	b_{26}	b_{12}	b_{11}	b_{27}	b_1	b_{10}	b_{26}
$\boldsymbol{b_{26}}$	b_{26}	b_{19}	b_{144}	b_{24}	b_5	b_{21}	b_{14}	b_3	b_{19}	b_{14}	b_{16}	b_{15}	b_{18}	b_{17}	b_{20}	b_{16}	b_{19}	b_{15}	b_{20}	b_{16}	b_{19}	b_{15}	b_{20}	b_{16}	b_{19}	b_{15}	b_{20}
$\boldsymbol{b_{27}}$	b_{27}	b_{20}	b_{16}	b_{25}	b_{21}	b_{23}	b_{22}	b_5	b_{23}	b_{22}	b_9	b_{15}	b_{17}	b_{16}	b_{18}	b_{17}	b_2	b_{19}	b_{16}	b_{20}	b_{17}	b_2	b_{19}	b_{16}	b_{20}	b_{17}	b_2

Таблица 3.
Тождества, получаемые при построении таблицы умножения Кэли для $B(2,3)$.

T_1	*0011=1010*
T_2	*0101=1100*
T_3	*10010=01100*
T_4	*01001=10100*
T_5	*11011=00100*
T_6	*11001=10110*
T_7	*01101=11010*
T_8	*1001=0110*

Таблица 4.
Статистика элементов $B(2,4)$ в зависимости от длины слова.

Длина слова	**Количество слов данной длины в группе**
0	*1*
1	*2*
2	*4*
3	*8*
4	*14*
5	*26*
6	*46*
7	*80*
8	*133*
9	*218*
10	*354*
11	*566*

Таблица 5.
Тождества, получаемые при построении таблицы умножения Кэли для $B(2,4)$.

T_1	*101010=000111*
T_2	*111000=010101*
T_3	*001110 0=1001001*
T_4	*011011 0=1100011*
T_5	*00011000=1101101*
T_6	*00100100=11100111*
T_7	*11101010=01101101*
T_8	*10001000=0110111*
T_9	*11001100=00110011*
T_{10}	*11101110=00010001*
T_{11}	*101011010=010010111*
T_{12}	*101001010=000101101*
T_{13}	*111010010=010110101*
T_{14}	*101101000=010100101*
T_{15}	*0010001000=1010100111*
T_{16}	*0001101010=1110111011*
T_{17}	*0111001010=1000100111*

T_{18}	*0001101110=1010110001*
T_{19}	*10010111000=00111000101*
T_{20}	*00111010010=10100101001*
T_{21}	*10101100100=0010001011 1*
T_{22}	*10100110110=00011011101*
T_{23}	*11011001010=00010111011*
T_{24}	*11101000100=00100110101*
T_{25}	*01010011100=11100100010*
T_{26}	*01101011010=10110100011*
T_{27}	*11011101000=01010011011*
T_{28}	*10111011000=01101100101*

Литература

1. *Адян С. И.* Проблема Бернсайда в тождествах и группах.— М.: «Наука», 1975.
2. *Ануфриев И. Е.* Самоучитель MATLAB 5.3/6.x.— СПб.: БХВ-Петербург, 2002.

УДК 517.9:518.6

РЕАЛИЗАЦИЯ МЕТОДА ТЕЙЛОРА ДЛЯ РЕШЕНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ПАКЕТЕ MATLAB

Кулинич М. В.
Нижегородский государственный университет им. Н. И. Лобачевского,
Нижний Новгород,
e-mail:kul@hydro.appl.sci-nnov.ru

Создан пакет программ для решения систем обыкновенных дифференциальных уравнений методом Тейлора произвольного порядка, являющийся дополнением для пакета ODE MATLAB. Для вычисления старших производных в символьной форме используется пакет Symbolic системы MATLAB. Особенностью пакета является предварительная генерация программы для заданной системы и заданного порядка метода Тейлора. Для облегчения использования функций пакета в целях демонстрации предоставляется графический интерфейс.

1. Метод Тейлора

Для решения систем обыкновенных дифференциальных уравнений (ОДУ)

$$\frac{d\mathbf{y}(x)}{dx} = \mathbf{f}(x, \mathbf{y}), \tag{1}$$

с начальными условиями

$$\mathbf{y}(x_0) = \mathbf{y}^0, \quad \mathbf{y} \in \mathbb{R}^s, \mathrm{x} \in \mathbb{R} \tag{2}$$

существует множество различных методов [1], особенностью которых является использование информации только о правой части системы уравнений (1).

Одним из старейших одношаговых методов, использующих информацию о производных решения задачи (1)–(2), является метод Тейлора. Вычислительные трудности, связанные с нахождением старших производных этого метода оправданы в случае требования вычислений с высокой степенью точности, например, при расчетах планетарных орбит [2].

Для решения ОДУ метод Тейлора реализован в пакете «TAYLOR» [3–5] и позднее в пакете ATSMCC [6] в коде языка FORTRAN. Решение ОДУ методом Тейлора с использованием символико-численного анализа и подключением NAG Fortran библиотеки, можно найти в работе [7].

Вычисление производных в символьной форме может быть осуществлено точно с помощью пакетов MAPLE, MATHEMATICA и MATLAB,

что позволяет реализовать алгоритм Тейлора для заданного порядка метода N. В пакете MATHEMATICA реализацию алгоритма Тейлора можно найти в работе [8].

Решение $\mathbf{y}(x)$ (в предположении надлежащей дифференцируемости) уравнения (1) в точке $(x+h)$, можно получить при помощи его разложения в ряд Тейлора

$$\mathbf{y}(x+h)=\mathbf{y}(x)+h\mathbf{y}'(x)+\frac{h^2}{2!}\mathbf{y}''(x)+\ldots+\frac{h^N}{N!}\mathbf{y}^{(N)}(x)+\ldots. \tag{3}$$

Если в ряде (3) ограничиться N-членами, то в результате получим одношаговый алгоритм

$$\mathbf{y}(x_0)=\mathbf{y}^0,$$

$$\mathbf{y}(x+h)=\mathbf{y}(x)+h\mathbf{y}'(x)+\frac{h^2}{2!}\mathbf{y}''(x)+\ldots+\frac{h^N}{N!}\mathbf{y}^{(N)}(x) \tag{4}$$

с локальной погрешностью

$$\mathbf{E}=\mathbf{y}^{(N+1)}(\varsigma)\frac{h^{N+1}}{(N+1)!}, \tag{5}$$

$$x\le\xi\le x+h.$$

Используя исходное уравнение (1), алгоритм для нахождения приближенного решения $\mathbf{y}_i$ методом Тейлора порядка N можем представить в форме

$$\mathbf{y}_{i+1}=\mathbf{y}_i+h[\mathbf{f}(x_i,\mathbf{y}_i)+\ldots+\mathbf{f}^{(N-1)}(x_i,\mathbf{y}_i)\frac{h^{N-1}}{N!}], \tag{6}$$

$$x_{i+1}=x_i+h, i=1,\ldots,n-1,$$

$$\mathbf{y}_o=\mathbf{y}^o.$$

Полные производные входящие в (6) могут быть определены рекурсивно:

$$\mathbf{f}^{(1)}(x,\mathbf{y})=\mathbf{f}_x(x,\mathbf{y})+\mathbf{f}_y(x,\mathbf{y})\cdot\mathbf{f}(x,\mathbf{y}), \tag{7}$$

$$\mathbf{f}^{(k)}(x,\mathbf{y})=\mathbf{f}_x{}^{(k-1)}+\mathbf{f}_y{}^{(k-1)}(x,\mathbf{y})\cdot\mathbf{f}(x,\mathbf{y}), k=2,3,\ldots. \tag{8}$$

При N=1 метод Тейлора известен под названием метода Эйлера, при N=2 — под названием метода Хойна.

Современные вычислительные возможности пакета MATLAB позволяют реализовать изложенный алгоритм (6)–(8) символико-численным способом.

2. Контроль погрешности решения обыкновенных дифференциальных уравнений методом Тейлора

Контроль точности, при нахождении решения с переменным шагом интегрирования, в методе Тейлора может быть осуществлен как и в программах решения ОДУ пакета ODE MATLAB [9].

Имея в распоряжении способ вычисления локальной погрешности (5) с оценкой

$$|| \mathbf{E} ||_\infty \approx || \mathbf{Y}_{N+1} ||_\infty \; h^{N+1}, \tag{9}$$
$$\mathbf{Y}_N = 1/N! d^N \mathbf{y}(x_i)/dx^N,$$

которая с точностью до членов более высокого порядка равна главному члену локальной погрешности метода Тейлора, величину шага интегрирования будем выбирать автоматически, исходя из следующих рассуждений.

Пусть оценка вычисленной погрешности превосходит границу заданной погрешности на шаге вычислений в узле $x_i + h$

$$\left\| \mathbf{E} \right\| \geq \max(retol \, || \mathbf{y} ||_\infty, || \mathbf{abstol} ||_\infty) = \delta, \tag{10}$$

тогда будем считать, что метод не достигает заданной точности, и $\mathbf{y}_{i+1}$ вместе с узлом $x_i + h$ исключается из рассмотрения.

Новый шаг интегрирования выберем в виде

$$h_\delta = \alpha \cdot h \tag{11}$$

и его значение вычислим, исходя из условия

$$\delta = || \mathbf{Y}_{N+1} ||_\infty \; h_\delta^{N+1}. \tag{12}$$

Сравнивая (9) и (12) для нахождения значения α, будем иметь выражение

$$\alpha = \left(\delta \Big/ || \mathbf{E} ||_\infty \right)^{\frac{1}{N+1}}. \tag{13}$$

Шаг интегрирования будем рассчитывать из выражения

$$h = fac \cdot h \left(\delta \Big/ || \mathbf{E} ||_\infty \right)^{\frac{1}{N+1}}, \tag{14}$$

Значение *fac* задается равным 0.9 для того, чтобы избежать тех шагов, где не достигается заданная точность.

Если

$$|| \mathbf{E} ||_\infty \leq \max(retol \, || \mathbf{y} ||_\infty, || \mathbf{abstol} ||_\infty) = \delta,$$

то найденное значение решения $\mathbf{y}_{i+1}$ считается приемлемым, т. е. удовлетворяет требуемой точности и значение $x_i + h$ принимается в качестве следующего узла x_{i+1} интервала интегрирования.

Контроль исчезновения порядка будем проводить аналогично проведению этого контроля пакета ODE MATLAB [9].

$$h \geq h_{\min} = 16\varepsilon \, | x |.$$

Здесь ε — машинная точность, а x текущее значение независимой переменной.

3. Описание функций пакета решения дифференциальных уравнений методом Тейлора

В работе алгоритм Тейлора реализован в виде нескольких функций в системе MATLAB с использованием ODE MATLAB и пакета Symbolic.

Основной функцией пакета является функция ODENNSG (см. текст функции odennsg.m в приложении), предоставляющая возможность сгенерировать программу для нахождения решения заданного уравнения при заданном порядке метода Тейлора, а затем по построенной программе получить решение задачи (1)-(2).

Программа ODENNSG довольно проста в обращении. Для ее использования достаточно, в качестве входных параметров, задать правые части системы обыкновенных дифференциальных уравнений и порядок метода N, т. е. количество членов, используемых в разложении искомого решения, в ряд Тейлора.

По умолчанию ODENNSG генерирует файл с именем ‘odetN.m’. Символ N указывает порядок метода. В дальнейшем, сгенерированную функцию ODETN можно использовать с интерфейсом функций пакета ODE MATLAB, только без указания правой части решаемой системы.

Для функций пакета принят единый стандарт на задание правых частей системы дифференциальных уравнений в виде файла. Например, для системы уравнений

$$\begin{aligned} y_1' &= -\frac{8}{3}\cdot y_1 + y_2 \cdot y_3, \\ y_2' &= -10 \cdot y_2 + 10 \cdot y_3, \\ y_3' &= -y_2 \cdot y_1 + 28 \cdot y_2 - y_3, \end{aligned} \qquad (15)$$

описывающей аттрактор Лоренца, функция задания правой части имеет вид:

```
function r = odefunc(x, y)
r(1) = sym('-8/3*y1 + y2*y3');
r(2) = sym('-10*y2 + 10*y3');
r(3) = sym('-y2*y1 + 28*y2 — y3');
```

Здесь принято, что

$$\begin{aligned} y1 &\Leftrightarrow y_1, \\ y2 &\Leftrightarrow y_2, \\ &\dots \\ yn &\Leftrightarrow y_n. \end{aligned}$$

sym — функция системы MATLAB, представляющая переменные символьными.

Сделав небольшое отступление, опишем синтаксис функции **ODENNSG**:

1) R=ODENNSG(ODEFUN,N).

Здесь **ODEFUN** — это указатель на функцию, определяющую правую часть системы обыкновенных дифференциальных уравнений первого порядка (1). N — порядок метода Тейлора.

В возвращаемом параметре R записывается имя созданного файла в символьной форме.

2) R=ODENNSG(ODEFUN,N,FILE).

Первые два параметра имеют тот же смысл, что и при первом способе обращения, а последний параметр FILE определяет имя выходного файла. Например, вызвав

>> odennsg(@odefunc,3,'ex1.m');

на выходе получим файл ‘ex1.m’, решающий систему с правой частью, записанной в файле ‘odefunc.m’.

Как было сказано ранее, функция **ODENNSG** создает файл, который является функцией решения заданной системы алгоритмом Тейлора. Созданную функцию можно неоднократно использовать, меняя только входные параметры.

Архитектура сгенерированной функции, имеющей по умолчанию имя ‘odetN.m’, аналогична архитектуре функций ODE MATLAB для решения дифференциальных уравнений. Обратиться к сгенерированной функции можно практически так же, как и к стандартным функциям из ODE MATLAB за исключением указателя на функцию, определяющую правую часть системы дифференциальных уравнений, так как она уже заложена внутри сгенерированной функции **ODETN**.

Синтаксис обращения к функции **ODETN**:

1) [T,Y]=ODETN(TSPAN,Y0,U).

Входные параметры: промежуток интегрирования TSPAN и начальные условия Y0 решаемой системы. Выходные параметры: Y — вектор решения в точках вектора T. По умолчанию алгоритм ищет решение с переменным шагом, но если требуется найти решение на выходе в определенных точках T0,T1,…,TFINAL, то промежуток интегрирования следует передавать как вектор, записываемый, например, в виде TSPAN=[T0 T1 … TFINAL].

Последний параметр U(необязательный) служит для передачи значений в правую часть решаемой системы, если ее правая часть представляется в форме $\mathbf{f}(x,\mathbf{y},\mathbf{u}),\mathbf{u}\in\mathbb{R}^m$. Вектор U можно изменять, не генерируя алгоритм для вычисления решения заново.

Обращение к функции **ODETN** для вычисления решения с переменным шагом в системе MATLAB будет выглядеть следующим образом

>> odetn([0 10], [1 1 1]);

Для вычисления решения в заданных точках 0,0.1,…,10

>> odetn([0:0.1:10], [1 1 1]);

2) [T,Y]=ODETN(TSPAN,Y0,U,OPTIONS).

В этой форме обращения к функции **ODETN** добавлен необязательный параметр OPTIONS —- структура, используемая в пакете ODE

MATLAB. Через этот параметр можно изменять такие значения, как абсолютная, относительная погрешность, величина начального шага и некоторые другие. Работа со структурой OPTIONS осуществляется с помощью функции **ODESET**. Узнать все возможные поля структуры OPTIONS можно при помощи вызова функции **ODESET** без параметров.

Например, относительная погрешность в **ODETN** по умолчанию равна 1e-3, чтобы изменить ее на 1e-6, выполняем следующую команду

>> options=odeset('RelTol', 1e-6);

При вызове ODETN с опциями

>> odetn([0 10], [1 1 1], [], options);

решение системы дифференциальных уравнений будет проводиться с относительной точностью равной 1e–6.

При решении дифференциальных уравнений методом Тейлора на высокопроизводительных вычислительных системах процесс генерации алгоритма решения и нахождения решения можно не разделять. Для этого в пакете существует функция **ODENNS**.

Тело функции можно посмотреть в Приложении (файл odenns.m).

Синтаксис обращения к функции **ODENNS**:

1) [S,T,Y]=ODENNS(ODEFUN,TSPAN,Y0,N,U).

ODEFUN — это указатель на функцию, определяющую правую часть системы обыкновенных дифференциальных уравнений первого порядка (1). N — порядок метода Тейлора. TSPAN — промежуток интегрирования, вектор Y0 —- начальные условия решаемой системы. Y — вектор решения в точках вектора T. По умолчанию алгоритм ищет решение с переменным шагом, но если требуется найти решение в определенных точках T0,T1,…,TFINAL, то промежуток интегрирования следует передавать как вектор, записываемый, например, в виде TSPAN=[T0 T1 … TFINAL]. S — символьные формулы для отыскания решения заданной системы с порядком точности N. Последний параметр U(необязательный) — вектор параметров.

2) [S,T,Y]=ODENNS(ODEFUNC,TSPAN,Y0,N,U,OPTIONS).

В этой форме обращения к ODENNS добавлен необязательный параметр OPTIONS. Пример работы с этим параметром представлен при описании функции **ODETN**.

Тело функции можно посмотреть в Приложении (файл odenns.m).

При разработке функции ODENNSG, генерирующей алгоритм Тейлора для нахождения решения системы уравнений (1), написана вспомогательная функция — **ODECONV**, позволяющая построить правую часть системы первого порядка по виду дифференциального уравнения n-ого порядка, разрешенного относительно старшей производной

$$y^{(n)} = f(x, y', ..., y^{(n-1)}), \qquad (7)$$

при его преобразовании к системе обыкновенных дифференциальных уравнений

$$y_1' = y_2$$
$$\dots \qquad . \qquad (8)$$
$$y_n' = f(x, y_1, \dots y_n)$$

Обратиться к ней в системе MATLAB можно следующим образом:

1) R=ODECONV(S,N,T).

Здесь параметр S определяет правую часть уравнения (7). N говорит о старшей производной. Параметр T может быть определен как ‘symbolic’ или ‘numeral’. От его определения зависит форма записи результата исполнения функции.

Например, при преобразовании уравнения

$$y^{(4)} = y + y'' + x$$

к системе дифференциальных уравнений, отклик функции **ODECONV** с параметром T равным ‘symbolic’ будет следующим

```
>> r=odeconv('y(1)+y(3)+x',4,'symbolic')

r =

[    y2,    y3,    y4, y1+y3+x].
```

Здесь ‘symbolic’ указывает на то, что выходные данные будут представлены в символьной форме, принятой в описываемом пакете.

Если же указать T равным ‘numeral’, то результат выполнения **ODECONV** становится таким:

```
>> r=odeconv('y(1)+y(3)+x',4,'numeral')

r =

[    y(2),    y(3),    y(4), y(1)+y(3)+x]
```

2) R=ODECONV(S,N,T,F).

В данном случае функция **ODECONV** выполняет все те же действия, что в предыдущем пункте, но добавляется возможность создания файла с правой частью решаемого уравнения, приведенного к системе (8). При этом, созданный файл может использоваться как с описанной ранее функцией **ODENNSG**, так и со стандартными функциями системы MATLAB, такими как ODE45, ODE23, ….

К примеру, вызвав

```
>> r=odeconv('y(1)+y(3)+x',4,'numeral','func.m')
```

результат остается прежним. Но, кроме этого, создастся файл с именем ‘func.m’. Если его открыть, то мы увидим такую функцию:

```
function r = a(x, y)
r(1, :) = y(2);
r(2, :) = y(3);
r(3, :) = y(4);
r(4, :) = y(1)+y(3)+x;
```

Будучи независимой, функция **ODECONV** может быть использована не только в рамках описываемого пакета, но и самостоятельно. Для этого достаточно установить определенным образом один из параметров, передаваемых в функцию и указать имя файла.

Тело функции можно посмотреть в Приложении (файл odeconv.m).

4. Графический интерфейс, иллюстрирующий работу пакета решения обыкновенных дифференциальных уравнений методом Тейлора

К описываемому пакету прилагается графический интерфейс, написанный в командах языка MATLAB, который в наглядной форме предоставляет доступ ко всем основным возможностям программы ODENNSG. В графическом окне оболочки имеется возможность в естественной форме задавать для решения, как системы обыкновенных дифференциальных уравнений первого порядка, так и обыкновенные уравнения n-ого порядка, разрешенные относительно старшей производной.

Основой оболочки является функция ODEV, вызываемая без параметров, после запуска которой на экране компьютера появляется диалоговое окно (см. рис.1). Тело функции можно посмотреть в Приложении (файл odev.m).

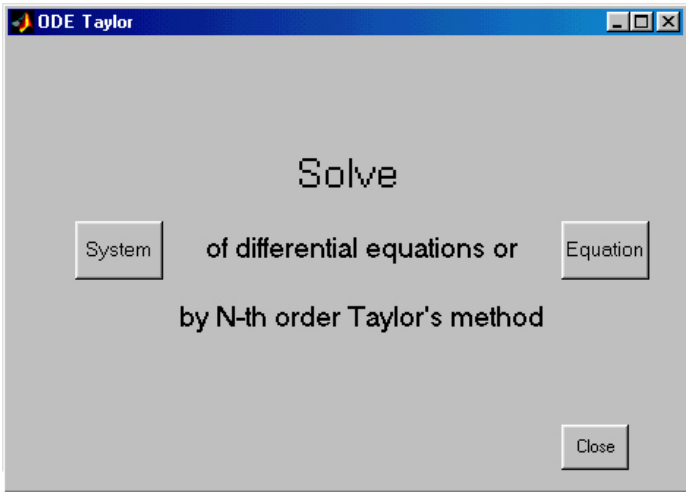


Рис. 1. Главное диалоговое окно функции ODEV.

С помощью кнопок **System** и **Equation** выбирается тип решаемых уравнений. При нажатии кнопки **System** запускается функция VSYSTEM (см. файл vsystem.m приложения) и открывается диалоговое окно (см. рис. 2), где для решения задается система дифференциальных уравнений первого порядка. Если же щелкнуть курсором мыши по кнопке **Equation**, то запускается функция VEQUATION (см. файл vequation.m приложения) и появляется окно для решения дифференциальных уравнений n-ого порядка, разрешенных относительно старшей производной (см. рис. 3).

В диалоговом окне для решения систем существует несколько полей для ввода информации. Во-первых, задается поле для описания правых частей системы обыкновенных дифференциальных уравнений первого порядка

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}, \mathbf{u}),$$

во-вторых, поле для задания параметра N — порядка метода Тейлора.

После ввода этих параметров можно построить программу для решения заданной системы уравнений, нажав кнопку **Gen**.

Для демонстрации решения задачи Коши следует задать обязательные параметры, такие как вектор начальных условий y0 и промежуток интегрирования. После этого можно посмотреть график полученного решения, нажав кнопку **Demo**.

Как видно, на рис. 2 присутствует еще одно поле для ввода, которое не было описано выше. Это поле для задания вектора параметров $\mathbf{u}$, значения которого можно менять, не генерируя программу для решения заданной задачи Коши заново.

Кнопка **Close** закрывает это диалоговое окно.

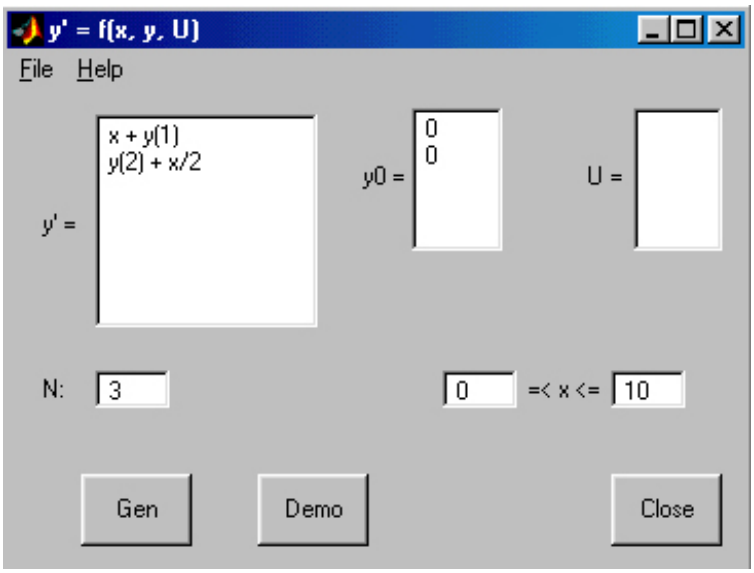


Рис. 2. Окно для решения систем уравнений.

В диалоговом окне (рис.3) для генерации программы решения уравнения n-порядка, разрешенного относительно старшей производной, существуют те же поля для ввода информации. Во-первых, поле задания правой части уравнения

$$y^{(n)} = f(x, y', ..., y^{(n-1)}, \mathbf{u})$$

в форме

$$y(n) = f(x, y(1), \ldots, y(n-1), u(1), \ldots, u(m)).$$

Во-вторых, задается поле для ввода параметра N — порядка метода Тейлора.

После ввода этих параметров можно построить программу для решения заданного уравнения, нажав кнопку **Gen**.

Для демонстрации решения следует задать обязательные параметры, такие как начальные условия $y(x_0), y'(x_0), ..., y^{(n-1)}(x_0)$ и промежуток интегрирования. После этого можно посмотреть график полученного решения, нажав кнопку **Demo**.

На рис. 3 поле U служит для передачи в программу вектора **u**, которые можно менять, не генерируя саму программу для решения заданного уравнения заново.

Кнопка **Close** закрывает это диалоговое окно.

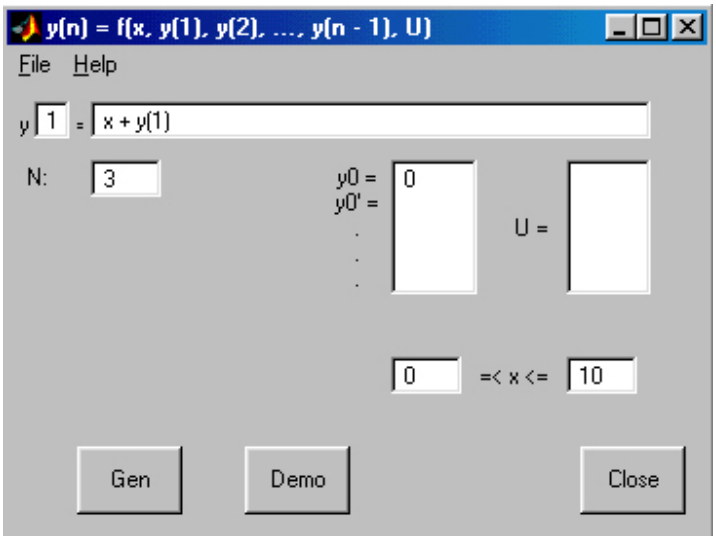


Рис. 3. Окно для решения уравнений n-ого порядка.

Кнопка **File** в окне ‘System’ и окне ‘Equation’ служит для сохранения сценариев и вызова предыдущих сохраненных сценариев.

5. Тестирование пакета для решения обыкновенных дифференциальных уравнений методом Тейлора

В данном разделе представляются результаты тестирования пакета для решения ОДУ методом Тейлора уравнений осциллятора и Лоренца. В качестве параметра тестирования выбрано время решения при различных относительных ошибках *reltol*. Результаты сравниваются с решением аналогичных задач программой ODE45 пакета ODE MATLAB, использующей алгоритм Рунге-Кутта 5-го порядка.

В таблице 1 представлено время решения процессором Pentium IV 1600MHz задачи Коши

$$y_1' = y_2, \qquad y_1(0) = 1,$$
$$y_2' = -y_1 \qquad y_2(0) = 1$$

на интервале [0, 1000], в таблице 2 — время решения задачи Коши

$$y_1' = -\frac{8}{3}\cdot y_1 + y_2 \cdot y_3, \qquad y_1(0) = 1,$$
$$y_2' = -10 \cdot y_2 + 10 \cdot y_3, \qquad y_2(0) = 1,$$
$$y_3' = -y_2 \cdot y_1 + 28 \cdot y_2 - y_3, \; y_3(0) = 1$$

на интервале [0, 1000].

Таблица 1.
Время решения уравнений осциллятора методом Тейлора *N*-го порядка и программой ODE45 системы MATLAB на интервале от 0 до 1000.

reltol	ODE45	ODE Taylor N-th order		
		N=5	N=10	N=15
1e-3	1.98	1.37	1.21	1.2
1e-6	3.96	2.42	1.27	1.2099999999
1e-12	3.95	2.47	1.26	1.21

Таблица 2.
Время решения уравнений Лоренца методом Тейлора N-го порядка и программой ODE45 системы MATLAB на интервале от 0 до 1000.

reltol	ODE45	ODE Taylor N-th order		
		N=5	N=10	N=15
1e-3	11.15	13.18	19.88	44.65
1e-6	31.91	39.22	29.93	54.76
1e-12	44.16	72.94	37.19	63.39

В первом случае старшие производные от правой части достаточно простые и время решения для метода высокого порядка слабо зависит от порядка и точности метода, во втором случае, где производные более громоздки и, следовательно, больше времени тратится на их вычисление, порядок точности сказывается на общем времени решения.

Следует заметить, что время генерации алгоритма достаточно большое, и тем больше, чем выше порядок точности.

Для параллельных процессоров с разрядностью машинного слова более 32 бит, результаты тестирования метода Тейлора высокого порядка для решения ОДУ, можно найти в работе [10].

Литература

1. Современные численные методы решения обыкновенных дифференциальных уравнений / Под ред. Дж. Холла и Дж. Уатт.— М.: Мир, 1979.
2. *Deprit A., Zahar R. M. V.* Numerical Integration of an Orbit and Its Concomitant Variation // Z. Angew. Math. Phys.— N.17.— 1966.— P.425–430.
3. *Norman A. C.* TAYLOR Users Manual. Computing Laboratory.— University of Cambridge, UK, 1973.

4. *Norman A. C.* Computing with formal power series // ASM Trans. Math. Softw.— N.1.— 1975.— P.346–356.
5. *Norman A. C.* Expanding the solutions of implicit sets of ordinary differential equation // Comp. J.— N.19.— 1976, P.63–68.
6. *Corliss G. E., Chang Y. E.* Solving ordinary differential equations using Taylor series // ACM Trans. Math. Softw.— 8,2.— June 1982,— P.114–144.
7. *Dupee B. J, Davenport J. H.* An intelligent interface to numerical routines // DISCO'96 / J. Calmet and L. Limongelly, Eds. V.1128 of Lecture Notes in Computer Science.— Springer Verlag, New York, 1996.
8. *Urintsev A. L.* Solution of ODE System in a Power Series Form (odeteylo.m) (http://library.wolfram.com/infocenter/MathSource/3373/).
9. *Shapine F., Reichelt M.* The MATLAB ode suite // SIAM Journal on Scientific Computing.— N.18–1.— 1997.
10. *Bario R., Blesa F., Lara M.* High-precision of ODE with high-order Taylor methods in parallel. Monograffias de la Real Academia de Zaragosa, 2003.— 22: 67–74.

Приложение

Текст файлов приведен в разделе на CD в разделе «Разработки».

УДК 535.8:681.3

РЕШЕНИЕ ЗАДАЧ ГЕОМЕТРИЧЕСКОЙ И ВОЛНОВОЙ ОПТИКИ В СИСТЕМЕ MATLAB

Левин А. Д.
ВНИИ оптико-физических измерений, Москва,
e-mail:levin@cortec.ru

Расчет современных оптических систем для непараксиальных лучей требует довольно громоздких вычислений. Особенно сложными они становятся при необходимости учитывать распространение так называемых «косых» лучей, которые не лежат ни в меридиональной, ни в сагиттальной плоскостях. Большой объем вычислений требуется и при наличии в системе оптических элементов, в которых происходят явления интерференции или дифракции (зеркала с многослойными покрытиями, интерференционные светофильтры, дифракционные решетки). Специализированные программные продукты для расчета оптических систем ориентированы, как правило, на решение строго ограниченного круга задач (например [1]). Достаточно универсальным является пакет программ Zemax, разработанный фирмой Zemax Development Corporation (США) [2]. Он рассчитан в первую очередь на профессиональных проектировщиков оптических систем, расчетные алгоритмы и формулы «зашиты» внутрь программы и не видны пользователю, что лишает программу необходимой во многих случаях гибкости. В то же время большинство расчетных задач геометрической и волновой оптики могут успешно решаться с помощью системы MATLAB. При этом есть возможность оперативно корректировать расчетные алгоритмы. В данной работе возможности использования пакета MATLAB проиллюстрированы на примере решения двух задач, одна из которых относится к геометрической а другая к волновой оптике.

1. Расчет траектории произвольного луча, преломляемого поверхностью второго порядка

Произвольный луч, падающий на преломляющую поверхность, будем характеризовать матрицей ***Li*** размерностью 3*2. $Li_{11}, Li_{21}, Li_{31}$ — декартовы координаты точки в пространстве предметов, через которую проходит луч, L_{12}, L_{22}, L_{32} — направляющие косинусы этого луча. Тогда уравнение прямой, совпадающей с падающим лучом, принимает вид:

$$\frac{X - L_{11}}{L_{12}} = \frac{Y - L_{21}}{L_{22}} = \frac{Z - L_{31}}{L_{32}}. \qquad (1)$$

Преломляющую поверхность будем описывать вектором 5 компонентным вектором ***S.*** Компонента S_1, S_2 и S_3 этого вектора — коэффициенты канонического уравнения поверхности второго порядка

$$\frac{X^2}{S_1^2}+\frac{Y^2}{S_2^2}+\frac{Z^2}{S_3^2}=1. \qquad (2)$$

Это уравнение описывает сферические и эллиптические поверхности (для сферы $S_1=S_2=S_3=R$, где R — радиус сферы). Компоненты S_4 и S_5 — значения показателей преломления слева и справа от преломляющей поверхности.

Преломленный луч характеризуется матрицей ***Lr***, структура которой аналогична ***Li***.

С учетом общего подхода, изложенного в [3] ,был разработан следующий алгоритм для нахождения преломленного луча.

1. Путем совместного решения уравнений (1) и (2) находились координаты точек пересечения луча с преломляющей поверхностью Lr_{11}, Lr_{21}, Lr_{31}.

2. Находился радиус кривизны преломляющей поверхности в точке ее пересечения с падающим лучом и направляющие косинусы α,β, и γ единичной нормали, проведенной из этой точки.

3. Находился косинус угла падения φ, т. е. угла между падающим лучом и нормалью к поверхности мениска в точке падения

$$\cos\varphi = Lr_{11}*\alpha + Lr_{21}*\beta + Lr_{31}*\gamma. \qquad (3)$$

4. Использовалось уравнение

$$n*\cos\theta = \sqrt{n^2 - 1 + \cos^2\varphi}, \qquad (4)$$

которое следует из закона преломления света (θ — угол преломления).

5. Находятся Lr_{12}, Lr_{22}, Lr_{32}— направляющие косинусы преломленного луча. При этом используется то обстоятельство, что этот луч лежит в плоскости падающего луча и нормали к преломляющей поверхности.

На основании изложенного алгоритма была написана MATLAB функция ***Lr(Li, S),*** позволяющая рассчитать траекторию произвольного луча после его преломления сферической или эллиптической поверхностью. Небольшая модификация (изменение формата вектора ***S***) позволяет использовать эту функцию для расчета преломления другими поверхностями второго порядка (в том числе параболической и цилиндрической).

Кроме того, была написана функция ***Cross(Lr, h)***, которая возвращает координаты точки пересечения преломленного луча с плоскостью, перпендикулярной оптической оси и находящейся на расстоянии h от начала координат. Для расчета хода лучей через следующую преломляющую поверхность эти точки можно считать расположенными в пространстве предметов и вновь использовать функцию ***Lr(Li, S).*** Таким образом, последовательно применяя функции ***Lr(Li, S)*** и ***Cross(Lr,h)*** строится полная траектория луча через оптическую систему. Данный алгоритм позволяет

быстро рассчитывать ход достаточно большого числа произвольных лучей через сложную оптическую систему, не вводя каких-либо упрощающих предположений.

2. Расчет многослойных интерференционных покрытий

С помощью пакета MATLAB удобно моделировать интерференционные процессы в многослойных диэлектрических покрытиях. Такие покрытия применяются при изготовлении зеркал, имеющих заданную зависимость коэффициента отражения от длины волны, интерференционных светофильтров и других оптических элементов.

Многослойные диэлектрические покрытия обычно состоят из плоскопараллельных слоев прозрачных материалов, имеющих различную толщину и различные значения показателя преломления.

При построении и реализации расчетной модели было использовано понятие характеристической матрицы слоистой среды (иногда ее называют интерференционной матрицей). Эта матрица имеет размерность 2×2 и является результатом решения уравнений Максвелла, описывающих распространение электромагнитной волны в слоистой среде (см. например, [3]). Представление решения в матричной форме особенно удобно для численных расчетов с использованием MATLAB.

Выберем систему прямоугольных координат таким образом, чтобы ось Z была направлена по нормали к границам раздела между слоями, а плоскость Z=0 соответствует границе раздела между верхним слоем и воздухом. В этом случае характеристическая матрица связывает X и Y компоненты электрического вектора электромагнитной волны на плоскости Z=0 с этими компонентами на произвольной плоскости Z=const. В [3] показано, что характеристическая матрица слоистой системы является произведением характеристических матриц слоев, из которых эта состоит.

Введем следующие обозначения: θ_0 — угол падения излучения на слоистую структуру; n_0 — показатель преломления воздуха; θ_k— угол между осью Z и направлением распространения луча в k-ом слое; n_k — показатель преломления материала k-го слоя; h_k — толщина k-го слоя; θ_L — угол преломления светового луча, вошедшего в подложку; n_L — показатель преломления материала подложки; λ — длина волны.

Согласно [1], характеристическая матрица k–го слоя имеет вид

$$m_k = \begin{vmatrix} \cos\beta_k & -\dfrac{i}{p_k}\sin\beta_k \\ -ip_k\sin_k & \cos\beta_k \end{vmatrix}, \qquad (5)$$

где

$$\beta_k = \frac{2\pi}{\lambda} n_k h_k \cos\theta_k, \quad p_k = n_k \cos\theta_k; \qquad (6)$$

i — мнимая единица.

Характеристическая матрица M слоистой системы из N слоев может быть вычислена путем перемножения матриц всех слоев, составляющих эту систему.

$$M = \prod_{k=1}^{N} m_k \ . \tag{7}$$

Коэффициент отражения света (по амплитуде) слоистой структурой может быть выражен через элементы матрицы M.

$$r = \frac{(M_{11} + M_{12} p_L) p_0 - (M_{21} + M_{22} p_L) p_0}{(M_{11} + M_{12} p_L) p_0 + (M_{21} + M_{22} p_L) p_0}, \tag{8}$$

где $$p_0 = n_0 \cdot \cos\theta_0, \ p_L = n_L \cdot \cos\theta_L \tag{9}$$

Коэффициент отражения по мощности

$$R = r^2. \tag{10}$$

Для визуальной оценки отражающих свойств поверхности, а также для сравнения с экспериментальными данными важен именно коэффициент отражения по мощности.

Для нахождения углов θ_k достаточно рассмотреть преломление света на границе между k–1 и k слоями. Согласно закону преломления света

$$\frac{\sin\theta_{k-1}}{\sin\theta_k} = \frac{n_k}{n_{k-1}}. \tag{11}$$

Если на границе двух слоев оказывается, что $\sin\theta_k = (n_{k-1}/n_k)\sin\theta_{k-1} > 1$, значит угол θ_k не существует, т. е. на границе слоев произошло полное внутреннее отражение.

В частном случае, для границы раздела между воздухом и первым слоем, эта формула примет вид

$$\frac{\sin\theta_0}{\sin\theta_1} = \frac{n_1}{n_0}. \tag{12}$$

Угол θ_0 — это, как указывалось выше, угол падения излучения на слоистую структуру. Все остальные углы θ_k могут быть вычислены по рекуррентным формулам (11–12).

По алгоритму, основанному на формулах (5–12), были написаны MATLAB функции **Refl**(θ_0,λ) и **Trans**(θ_0 ,λ), которые рассчитывают, соответственно, значения коэффициентов отражения и пропускания для многослойной структуры в зависимости от угла падения θ_0 и длины волны λ. Для пользования этой функцией необходимо ввести в виде параметров рабочей области MATLAB следующие данные о слоистой структуре:

n — вектор показателей преломления слоев слоистой системы;

h — вектор толщин слоев,

a — вектор коэффициентов поглощения слоев.

Размерность каждого из векторов равна числу слоев в системе. Кроме того, вводятся два скалярных параметра — nL — показатель преломле-

ния подложки и *n0*– показатель преломления окружающей среды (чаще всего, это воздух).

В качестве примера приведем результаты расчета для слоистой структуры, состоящей из 8 пар чередующихся слоев ZnS (n=2,3) толщиной 0,5 мкм и криолита (n=1,35) толщиной 0,8 мкм. Эти материалы широко применяются в многослойных диэлектрических покрытиях.

На рис.1 приведена рассчитанная для такой структуры зависимость коэффициента отражения от длины волны.

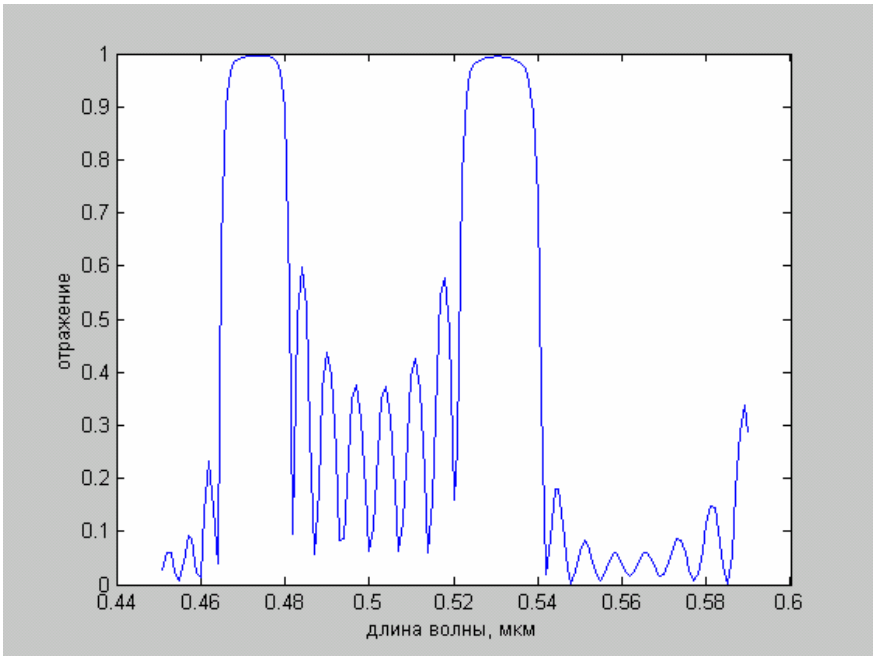


Рис. 1. Спектральная зависимость коэффициента отражения от длины волны для слоистой системы криолит — ZnS.

Данная программа позволяет, варьируя параметры слоистой структуры, подобрать оптимальные значения, необходимые для получения заданной спектральной зависимости коэффициентов отражения или пропускания. Это особенно важно, например, при компьютерном конструировании узкополосных диэлектрических зеркал или интерференционных светофильтров.

Литература

1. *Запрягаева Л. А., Свешникова И. С.* Расчет и проектирование оптических систем.— М., Логос, 2000.
2. ZEMAX. Software for optical design.— Zemax Development Corporation, 2004.
3. *Борн М., Вольф Э.* Основы оптики: пер. с англ.— М.: Наука, 1973.

УДК 004

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ АТМОСФЕРНЫХ ЭМИССИЙ В СРЕДЕ MATLAB С ИСПОЛЬЗОВАНИЕМ ЛИНЕЙНО-КОМБИНИРОВАННЫХ ОЦЕНОК

Линеенко М. Б., Кацюба О. А.
Самарская государственная академия путей сообщения, Самара,
e-mail:mbl@front.ru

Математическое моделирование загрязнения воздушной среды приобретает все большее значение. В связи с этим особую роль приобретают компьютерные методы моделирования эмиссий в атмосферу. Математическое моделирование процессов загрязнения атмосферы позволяет получить зависимость между интенсивностью эмиссии конкретного источника (или источников) и концентрацией атмосферных загрязнений в окружающем воздухе.

Итогом математического моделирования атмосферных эмиссий является определение концентрации загрязняющего вещества $C(x,y,z)$ в любой точке x, расположенной по ветру от источника эмиссии. При определенных допущениях [1] модель концентрации можно записать в виде:

$$C(x,y,z)=\frac{Q}{2\cdot\pi\cdot u\cdot\sigma_y\cdot\sigma_z}e^{-0{,}5\left(\frac{y}{\sigma_y}\right)^2}\cdot\left[e^{-0{,}5\left(\frac{z-H}{\sigma_z}\right)^2}+e^{-0{,}5\left(\frac{z+H}{\sigma_z}\right)^2}\right],$$

где Q — интенсивность эмиссии (кг/с); u — средняя скорость ветра (м/с); H — эффективная высота трубы (м); σ_y,σ_z — стандартные отклонения загрязняющего вещества в направлениях y и z. Для определения рассеивания загрязняющего вещества воспользуемся эмпирическими формулами [1]:

$$\sigma_y(x)=\frac{a_y^{(1)}x}{1+a_y^{(2)}\left(\frac{x}{u}\right)^{0,5}};\ \sigma_z(x)=\frac{a_z^{(1)}x}{1+a_z^{(2)}\left(\frac{x}{u}\right)^{0,5}}.$$

В окончательном виде модель принимает вид:

$$C(x,y,z)=\frac{Q\left[1+a_y^{(1)}\sqrt{x/u}\right]\cdot\left[1+a_z^{(1)}\sqrt{x/u}\right]}{2\pi u a_y^{(1)}a_z^{(1)}x^2}\exp\left[-0{,}5\left(\frac{y(1+a\sqrt{x/u})}{a_y^{(1)}x}\right)^2\right]\cdot$$

$$\cdot\left[\exp\left(-0{,}5\left(\frac{(z-H)(1+a_z^{(2)}\sqrt{x/u})}{a_z^{(1)}x}\right)^2\right)+\exp\left(-0{,}5\left(\frac{(z+H)(1+a_z^{(2)}\sqrt{x/u})}{a_z^{(1)}x}\right)^2\right)\right],(1)$$

при $C(x,y,z)$ измеряемой с помехой.

В этом случае задача определения модели концентрации загрязняющих веществ $C(x_1,x_2,x_3)$ в точках (x_1,x_2,x_3) сводится к задаче нелинейного параметрического оценивания параметров $b_1=Q/2u$, $b_2=a_y^{(1)}$, $b_3=a_y^{(2)}$, $b_4=a_z^{(1)}$ и $b_5=a_z^{(2)}$ при наличии помех наблюдений.

Формально эта задача может быть описана следующим образом. Пусть имеет место задача несмещенного оценивания регрессионных функций; наблюдаемые величины $y_i=C_i(x_1,x_2,x_3)$ представляются в виде $y_i=\eta(x_i,b_0)+\xi(i)$, где $b_0=(b_1,b_2,b_3)$, $\xi(i)$ — случайные независимые величины ($i=\overline{1,N}$), удовлетворяющие следующим условиям

$$E(\xi(i))=0,\ E(\xi^2(i))=\sigma_i^2,$$

где E — оператор математического ожидания.

Требуется определить состоятельные оценки $\hat{b}(N)$, не зная априорно закона распределения шума $f_i(\xi(i))=f_i(y_i,b_0)=f_{\xi_i}(y_i-\eta(x_i,b_0))$, причем эффективность этих оценок должна быть достаточно близка к эффективности оценок максимального правдоподобия.

В условиях априорной неопределенности существуют методы состоятельного оценивания (метод эмпирического риска, М-оценка и т. д.), однако очевидно, что эффективность этих оценок в общем случае далека от эффективности оценки максимального правдоподобия. Наиболее просто вопрос получения множества состоятельных оценок из которых может быть сконструирована достаточно эффективная оценка решается, если воспользоваться методом эмпирического риска, когда эмпирический функционал конструируется на основе некоторых законов распределения, и тогда указанный метод можно трактовать как непосредственное расширение метода максимального правдоподобия на случай априорной неопределенности при модификации некоторых его условий (такие оценки названы квазиправдоподобными), что позволяет получить более конструктивные условия состоятельности и асимптотической нормальности оценок, причем этот метод распространен на случай неоднородных наблюдений, а также на задачу оценивания параметров линейных разностных уравнений при наличии помех в выходных переменных.

Пусть эксперименты $\{x^{(N)},A_u^{(N)},F_b^{(N)}\}$ порождаются наблюдениями $Y=(y_1,...,y_N)^T$ со значениями в $\{x,A_u\}$ и распределениями $\{F_b\}$, пусть семейство $F^{(N)}$ доминируется некоторой мерой υ и существует функция

$$\frac{\partial F_b^{(N)}(y_1,...,y_N;b)}{\partial \upsilon(y_1,...,y_N)} = \frac{\partial F_b^{(N)}(Y^T,b)}{\partial \upsilon} = f_{(N)}(Y^T,b).$$

Определение 1. Функцией квазиправдоподобия параметров b, отвечающей $\left\{x^{(N)}, A_u^{(N)}, F_b^{(N)}\right\}$ и Y, называется неотрицательная вещественная функция

$$\varphi_{(N)}(Y^T,b) \neq f_{(N)}(Y^T,b),\ b \in B,$$

где $\varphi_N(Y^T,b) = \varphi^{(N)}_{\xi_1,...,\xi_N}(y_1 - \eta(x_1,b),...,y_N - \eta(x_N,b))$.

Оценкой максимального квазиправдоподобия (МКП) b_0 для заданной функции квазиправдоподобия $\varphi_{(N)}(Y^T,b)$ по наблюдениям Y называется $\hat{b}(N,\varphi)$, определяемая из

$$\varphi_{(N)}(Y^T,\hat{b}(N,\varphi)) = \sup_{b\in\tilde{B}} \varphi_{(N)}(Y^T,b).$$

В частности, если $\varphi_{(N)}(Y^T,b) = f_{(N)}(Y^T,b)$, то оценка будет правдоподобной.

Пусть имеется возможность получить последовательность состоятельных оценок с разной асимптотической эффективностью, при этом возникает задача повышения эффективности оценок путем комбинации их.

Определение 2. Любая линейная комбинация конечного числа квазиправдоподобных оценок $\hat{b}(N,\varphi^{(l)})$ $(l = \overline{1,k})$ относительно разных функций квазиправдоподобия $\varphi^{(l)}$ с определенными весами называется линейно-комбинированной квазиправдоподобной β_k.

Оказывается, что при определенных условиях, накладываемых на веса линейно-комбинированной оценки, имеет место следующее неравенство:

$$leff\ \beta_{k-1}/b_0 \le leff\ \beta_k/b_0,$$

причем, если в последовательности $\left\{\hat{b}(N,\varphi^{(l)})\right\}$ имеет место $\hat{b}(N,\varphi^{(l)}) \equiv \hat{b}(N,f)$ ($f(y_i,b)$ — закон распределения помех наблюдений), то $\beta_k = \beta_f$ (под значением $leff\ \beta_{k-1}/b_0$ понимается отношение значений обобщенной дисперсии оценки максимального правдоподобия и исследуемой оценки).

Для последовательности квазиправдоподобных оценок имеет место утверждение 1 [2].

Утверждение 1. Пусть имеет место последовательность состоятельных, асимптотически несмещенных оценок $\hat{b}(N,\varphi^{(1)})$ … $\hat{b}(N,\varphi^{(k)})$, при всех $b \in B$ имеет место равенство $\int_{-\infty}^{\infty} \frac{\partial \ln \varphi_i^{(l)}(y_i,b)}{\partial b^{(j)}} f(y_i,b)dy_i = 0$ и выполня-

ется условие дифференцирования по параметру b под знаком интеграла (в частном случае, когда $\varphi_i^{(l)} = f_i$ имеет место условие регулярности в смысле вторых производных по b), то линейно-комбинированная оценка

$$\beta_k = \left[\begin{array}{c|c|c|c} 1^T & 0^T & \cdots & 0^T \\ \hline 0^T & 1^T & \cdots & 0^T \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0^T & 0^T & \cdots & 1^T \end{array}\right] \Gamma^{-1} \left[\begin{array}{c|c|c|c} 1 & 0 & \cdots & 0 \\ \hline 0 & 1 & \cdots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & 1 \end{array}\right]^{-1} \left[\begin{array}{c|c|c|c} 1^T & 0^T & \cdots & 0^T \\ \hline 0^T & 1^T & \cdots & 0^T \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0^T & 0^T & \cdots & 1^T \end{array}\right] \Gamma^{-1} \left[\begin{array}{c} \hat{b}_1[N,\varphi^{(l)}] \\ \hline \vdots \\ \hline \hat{b}_p[N,\varphi^{(l)}] \end{array}\right]$$

будет состоятельной, асимптотически несмещенной с асимптотической эффективностью $leff\ \beta_{k-1}/b_0 \le leff\ \beta_k/b_0$, причем, если имеет место $\hat{b}(N,\varphi^{(S)}) \equiv \hat{b}(N,f)$ $(S = \overline{1,k})$, то $\beta_k \equiv \hat{b}(N,f)$, и $leff\ \beta_k/b_0 = 1$, где T - оператор транспонирования;

$$\hat{b}_j[N,\varphi^{(l)}] = \left|\begin{array}{c} \hat{b}^{(j)}(N,\varphi^{(1)}) \\ \hat{b}^{(j)}(N,\varphi^{(2)}) \\ \vdots \\ \hat{b}^{(j)}(N,\varphi^{(k)}) \end{array}\right|;$$

$$\Gamma = \left|\begin{array}{c|c|c} \operatorname{cov}(\hat{b}^{(1)}(N,\varphi^{(l)}),\operatorname{cov}(\hat{b}^{(1)}(N,\varphi^{(S)}) & \cdots & \operatorname{cov}(\hat{b}^{(1)}(N,\varphi^{(l)}),\operatorname{cov}(\hat{b}^{(p)}(N,\varphi^{(S)}) \\ \hline \operatorname{cov}(\hat{b}^{(2)}(N,\varphi^{(l)}),\operatorname{cov}(\hat{b}^{(1)}(N,\varphi^{(S)}) & \cdots & \operatorname{cov}(\hat{b}^{(2)}(N,\varphi^{(l)}),\operatorname{cov}(\hat{b}^{(p)}(N,\varphi^{(S)}) \\ \hline \vdots & \ddots & \vdots \\ \hline \operatorname{cov}(\hat{b}^{(p)}(N,\varphi^{(l)}),\operatorname{cov}(\hat{b}^{(1)}(N,\varphi^{(S)}) & \cdots & \operatorname{cov}\hat{b}^{(p)}(N,\varphi^{(l)}),\operatorname{cov}(\hat{b}^{(p)}(N,\varphi^{(S)}) \end{array}\right|$$

— ковариационная матрица квазиправдоподобных оценок $\hat{b}(N,\varphi^{(l)})$ (положительно определенная, невырожденная, $pk \times pk$), порядок матрицы

$$1 : k \times 1;\ 0 : k \times 1;\ \hat{b}_j(N,\varphi^{(l)}) : k \times 1.$$

Следствие 1. Ковариационная матрица линейно-комбинированных оценок вычисляется по формуле

$$D[\beta_k] = \left[\left[\begin{array}{c|c|c|c} 1^T & 0^T & \cdots & 0^T \\ \hline 0^T & 1^T & \cdots & 0^T \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0^T & 0^T & \cdots & 1^T \end{array}\right] \Gamma^{-1} \left[\begin{array}{c|c|c|c} 1 & 0 & \cdots & 0 \\ \hline 0^T & 1 & \cdots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & 1 \end{array}\right]\right]^{-1}$$

Достаточно сложным вопросом является выбор функции φ для получения квазиправдоподобных оценок, при этом функции φ должны обладать определенными свойствами. Было бы желательно, чтобы последовательность $\{\varphi^{(l)}\}$ включала в себя наиболее распространенные критерии, соответствующие законам распределения Лапласа, Гаусса, которые обла-

дают определенными экстремальными свойствами [3, с. 129; с. 134] при робастном оценивании параметров.

Одним из возможных примеров такой последовательности $\{\varphi^{(l)}\}$ является последовательность квазиправдоподобных оценок, соответствующая l — обобщенному нормальному закону

$$\varphi(y,\lambda,l)=\frac{1}{2(2\lambda_l^2)_i^{1/l}\Gamma\left(\frac{l+1}{l}\right)}\exp\left(-\frac{|y-\eta(x,b)|^l}{2\lambda_l^2}\right),$$

где $\Gamma\left(\frac{l+1}{l}\right)$ — гамма-функция; $2\lambda_i^2$ — величина, являющаяся функцией дисперсии; l — любое положительное число: при $l=1$ — закон Лапласа, при $l=2$ — закон Гаусса. Тогда в качестве формальных критериев при целых $l>0$ для получения квазиправдоподобных оценок примем последовательно

$$\min\sum_{i=1}^{N}\frac{1}{(2\lambda_1^2)_i}|y_i-\eta(x_i,b)|,\ \ldots,\min\sum_{i=1}^{N}\frac{1}{(2\lambda_l^2)_i}|y_i-\eta(x_i,b)|^l, \qquad (2)$$

что соответствует методу наименьших модулей, квадратов и т. д., так как логарифмическая функция квазиправдоподобия имеет вид

$$\ln\varphi^l(Y^T,\lambda,l)=\ln\frac{1}{\left[2\Gamma\left(\frac{l+1}{l}\right)\right]^N\left|\prod_{i=1}^{N}\left(2\lambda_l^2\right)_i^{1/l}\right|}-\sum_{i=1}^{N}\frac{1}{(2\lambda_l^2)_i}|y_i-\eta(x_i,b)|^l.$$

Следует отметить, что для оценки $\widehat{\Gamma}$ можно применить следующее выражение:

$$\hat{\mathrm{cov}}(\hat{b}(N,\varphi^{(s)}),\hat{b}(N,\varphi^{(l)}))=\frac{1}{N}\cdot\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial^2\ln\varphi_i^{(S)}(y_i,b)}{\partial b^{(m)}\partial b^{(j)}}\right)^{-1}\Bigg/_{b=\hat{b}(N,\varphi^{(S)})}\cdot$$
$$\cdot\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial\ln\varphi_i^{(S)}(y_i,b)}{\partial b^{(j)}}\Bigg/_{b=\hat{b}(N,\varphi^{(l)})}\cdot\frac{1}{N}\sum_{i=1}^{N}\frac{\partial\ln\varphi_i^{(S)}(y_i,b)}{\partial b^{(m)}}\Bigg/_{b=\hat{b}(N,\varphi^{(l)})}\right)\cdot$$
$$\cdot\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial^2\ln\varphi_i^{(l)}(y_i,b)}{\partial b^{(m)}\partial b^{(j)}}\right)^{-1}\Bigg/_{b=\hat{b}(N,\varphi^{(l)})}. \qquad (3)$$

В состав MATLAB входит Toolbox Optimization, предназначенный для решения линейных и нелинейных оптимизационных задач. Функции этого ToolBox реализуют основные алгоритмы оптимизации, причем воз-

можность настройки параметров оптимизации позволяет запрограммировать нужную функцию на эффективное решение поставленной задачи. С применением разработанных в среде MATLAB файл-функций и файл-программ появилась возможность решения достаточно сложных, но высокоточных и надежных методов определения мощности источников выброса вредных веществ и картины распределения концентрации вредных веществ в пространстве. Стохастическая модель дает возможность учесть все особенности конкретного объекта в реальных условиях при наличии помех наблюдений. В качестве последовательности квазиправдоподобных оценок параметров b_1, b_2, b_3, b_4, b_5 модели (1) примем оценки получаемые из минимизации сумм вида (2) при $l = \overline{2,4}$. Для нахождения точки минимума удобно воспользоваться функцией fminsearch, которая позволяет производить поиск локального минимума функции нескольких переменных. Перед применением fminsearch необходимо создать файл-функцию, вычисляющую значения функции (2) для каждого l. Аргументом файл-функции является вектор искомых параметров, данные экспериментов $y_i = C_i(x_1, x_2, x_3)$ в точках (x_1, x_2, x_3) загружаются из текстового файла. Количество используемых экспериментов — 50…100. После получения последовательности квазиправдоподобных оценок возможно определение линейно-комбинированной оценки β_k. Получение линейно-комбинированной оценки средствами MATLAB производилось с использованием выражения (3), в котором частные производные заменялись конечными разностями. На основе предложенных критериев было создано программное обеспечение. Использование этого программного обеспечения позволило определить модель распределения концентрации вредных веществ с погрешностью (остаточной дисперсией) не более 18–20%.

Литература

1. *Cooper C. D., Alley F. C.* Air Pollution Control: A Design Approach.— PWS Publishers, Boston, 1986.
2. *Katsyuba O. A., Lineenko M. B.* Regression nonlinear stochastic models in conditions of a priori indeterminacy // Book of Abstracts The Fourth International Conference Tools For Mathematical Modelling, June 23-28.— St. Petersburg, 2003.— P. 87.
3. *Фомин В. Н.* Рекуррентное оценивание и адаптивная фильтрация.— М.: Наука, 1984.

УДК 581.3

ОБУЧАЮЩАЯ ПОДСИСТЕМА САПР ТЕСТОПРИГОДНОГО ПРОЕКТИРОВАНИЯ АНАЛОГОВЫХ СХЕМ

Мосин С. Г.
Владимирский государственный университет, Владимир,
e-mail: smosin@ieee.org

Тестирование и диагностика являются сложными задачами, которые приходится решать на всех этапах жизненного цикла электронного устройства. Причем тестирование аналоговых схем остается более сложным и дорогостоящим процессом по сравнению с тестированием цифровых схем. Во многом это определяется такими особенностями аналоговых схем как нелинейный и непрерывный характер преобразования входных сигналов, сложная функциональная зависимость между входными и выходными сигналами, высокая чувствительность выходных параметров к вариациям внутрисхемных и внешних параметров, и др. Другой стороной тестирования аналоговых схем является отсутствие эффективных, универсальных методов, одинаково пригодных для выявления неисправностей и причин их возникновения в различных классах электронных устройств.

Предложенные и реализованные к настоящему времени методы тестирования аналоговых схем носят локальный характер и используются на уровне отдельных приложений. Рост требований к надежности и качеству функционирования электронных приборов вынуждает разрабатывать и применять на практике новые методы тестирования и диагностики. Одним из таких решений является подход тестопригодного проектирования электронных устройств, который позволяет уже на ранних стадиях разработки устройства осуществлять выбор условий будущего тестирования, формирование тестов и оценку их полноты и эффективности [1, 2].

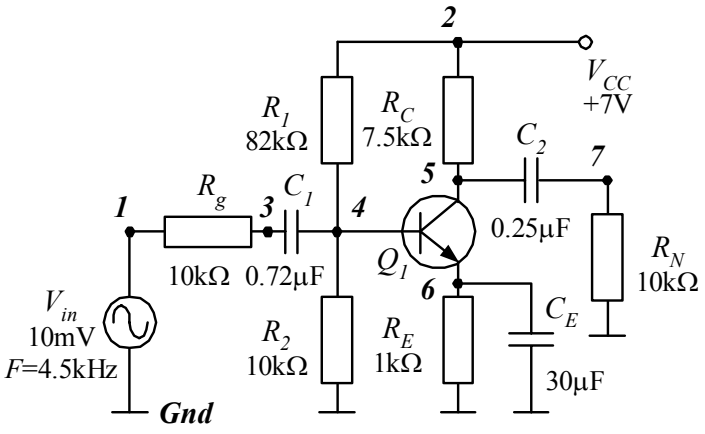
Успех тестопригодного проектирования во многом зависит, во-первых, от опыта инженера-разработчика, который в равной мере должен быть специалистом и в области проектирования, и в области тестирования, а, во-вторых, от используемых автоматизированных средств проектирования электронных устройств и формирования тестов для них. В этой связи важно отметить, что подготовка специалистов в области тестопригодного проектирования имеет большое значение. Хорошо известно, что одной из сторон процесса обучения являются практические занятия, позволяющие закрепить теоретические знания и обладающие зачастую большей наглядностью.

Практические занятия по вопросам тестирования и диагностики аналоговых схем могут быть построены на использовании специализирован-

ных тестовых САПР. Однако такие системы очень скромно представлены на мировом рынке программного обеспечения, а, кроме того, имеющиеся системы [3–5] обладают высокой стоимостью и зачастую являются недоступными для учебных заведений. Исследовательские подсистемы САПР тестирования охватывают лишь некоторые узкие, проблемные вопросы [6-9].

В данной работе предлагается описание подсистемы САПР *TeDiAC — **Te**sting and **Di**agnosis of **A**nalog **C**ircuits*, предназначенной для изучения вопросов тестирования и диагностики аналоговых схем [10]. Предлагаемая подсистема реализована в среде математических расчетов MATLAB и охватывает следующие вопросы, рассматриваемые в рамках тестирования и диагностики аналоговых схем, — моделирование выходных откликов, моделирование неисправностей, анализ тестопригодности устройств, выбор тестовых узлов и тестовых воздействий, а также построение справочников неисправностей.

Моделирование выходных откликов позволяет получить основные характеристики выходных сигналов аналоговой схемы. Исследуемую схему описывают на Spice-подобном языке в виде входного файла, который содержит информацию о внутренних компонентах, их номинальных значениях и допустимой области отклонения, а также узлах включения. Например, для схемы резисторного усилителя, представленного на рис. 1, входное описание будет иметь следующий вид:




```
; amp.cir
Rg  1 3  10k 5%
R1  2 4  82k 5%
R2  4 0  10k 5%
Rc  2 5 7.5k 5%
Re  6 0   1k 5%
Rn  7 0  10k 5%
C1  3 4 0.72u 10%
C2  5 7 0.25u 10%
Ce  0 6   30u 10%
Q1  5 4 6 Q2T316D
Vin 1 0 1V 0.1V 4.5kHz
Vcc 2 0 7V 0V 0Hz
```

Рис. 1. Резисторный усилительный каскад.

В системе *TeDiAC* моделирование аналоговых схем выполняют в трех режимах — статическом, малосигнальном и переходном, каждый из которых обеспечивает получение соответствующих характеристик схемы — положение рабочей точки, амплитудно-частотную и фазо-частотную характеристики, зависимость поведения выходного сигнала от времени. Параметры каждого из трех видов анализа задаются пользователем. Для анализа статического режима выбирают источник тока или напряжения, подключенный к исследуемой схеме, и указывают верхнюю и нижнюю

границы изменения его величины, а также шаг приращения. В результате моделирования для каждого значения варьируемого источника получаем значения постоянного напряжения в узлах схемы и значения постоянного тока, протекающего через компоненты устройства.

Для моделирования схемы в частотной области в режиме малого сигнала также выбирается входной источник тока или напряжения, задается частотный диапазон и определяется количество частотных точек, в которых будет проходить исследование устройства. Результатом моделирования являются амплитудно-частотная и фазо-частотная характеристики, которые отражают зависимость амплитуды и сдвига фазы выходного напряжения в тестовых узлах от частоты входного сигнала выбранного источника.

Для проведения анализа переходного режима пользователь задает конечное время и шаг моделирования аналоговой схемы. В результате получаем значения напряжения в узлах, зависящие от времени протекающих в схеме процессов. Для каждого вида анализа строятся графики полученных зависимостей для напряжения в узлах моделируемого устройства. Результаты моделирования резисторного усилительного каскада (рис. 1) в различных режимах приведены на рис. 2, 3 и 4.

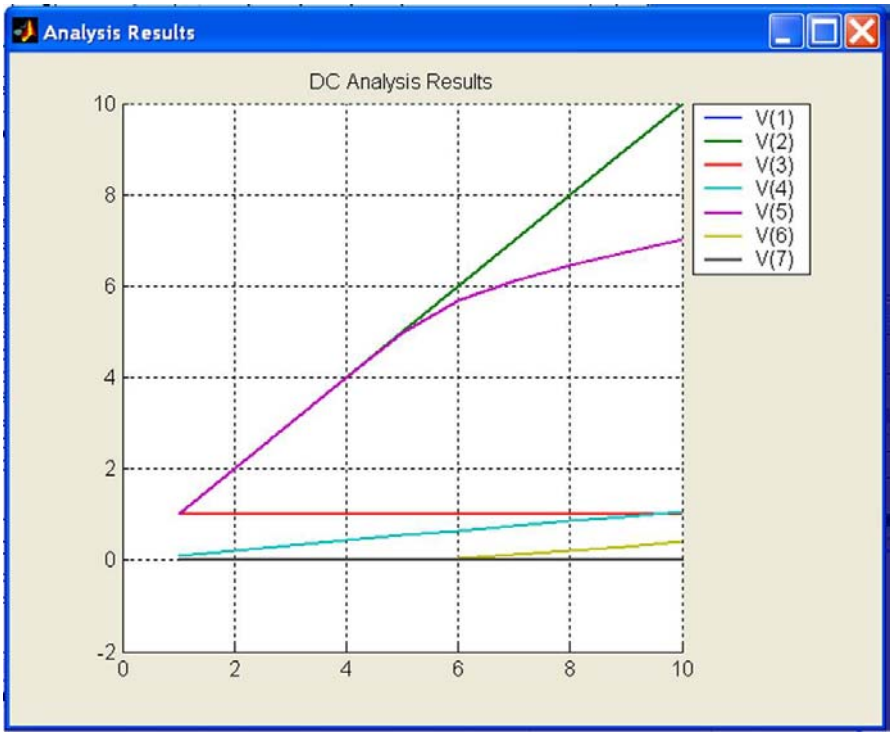


Рис. 2. Результаты моделирования резисторного усилительного каскада в статическом режиме при изменении напряжения V_{CC} в диапазоне от 1 до 10 В с шагом 1 В.

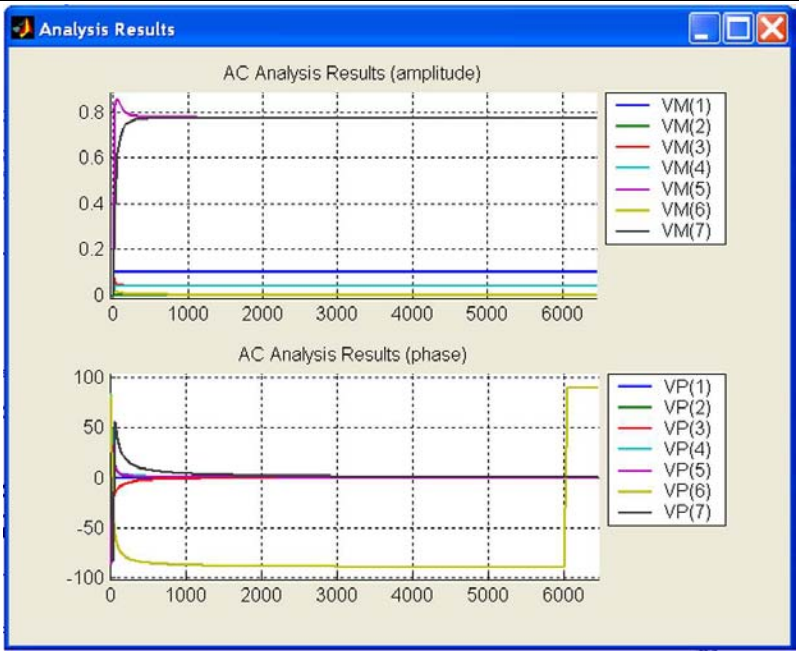


Рис. 3. Результаты моделирования резисторного усилительного каскада в малосигнальном режиме при изменении частоты входного напряжения V_{in} в диапазоне от 1 до 6.5 кГц (количество частотных точек равно 150).

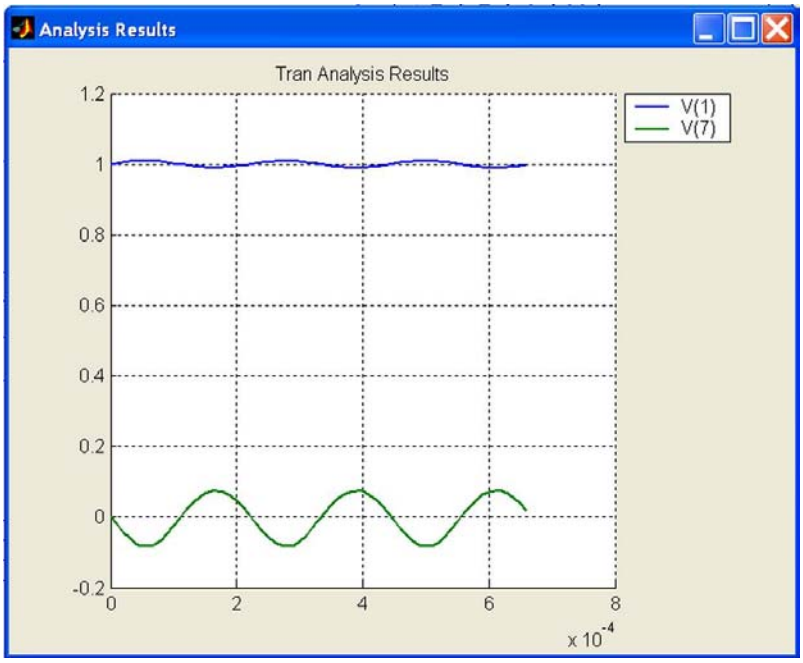


Рис. 4. Результаты моделирования резисторного усилительного каскада в переходном режиме на временном интервале от 0 до 3T.

При исследовании аналоговых схем большое значение имеет моделирование устройства, когда параметры внутренних компонентов принимают не строго номинальные значения, а величины из допустимой области. В ходе моделирования аналоговой схемы методом Монте-Карло, т.е. при многократном повторении эксперимента и назначении параметрам

компонентов случайных значений из допустимой области, удается оценить величину разброса выходной характеристики исправного устройства. Для моделирования исследуемой схемы методом Монте-Карло в системе *TeDiAC* необходимо задать вид проводимого анализа и количество его повторений. В результате расчетов строится семейство графиков выходной характеристики, полученных при случайных значениях параметров внутренних компонентов из допустимой области и график выходной характеристики для номинальных значений параметров.

Анализ результатов позволяет оценить диапазон разброса выходных характеристик исправно функционирующей схемы. На рис. 5 и 6 представлены результаты параметрического исследования «исправной» схемы резисторного усилителя в малосигнальном и переходном режимах соответственно.

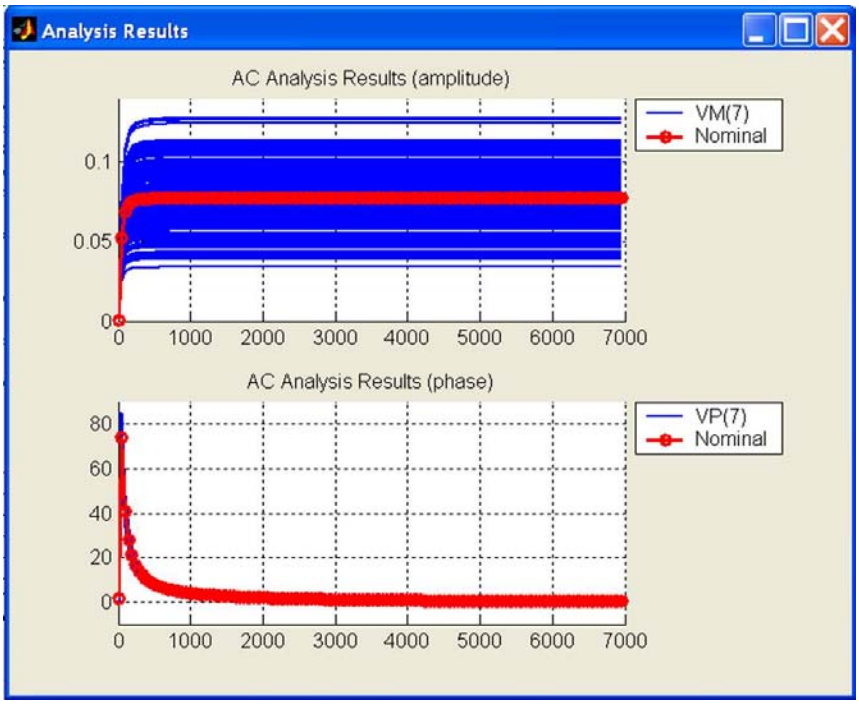


Рис. 5. Результаты параметрического моделирования резисторного усилительного каскада в малосигнальном режиме (число итераций равно 150).

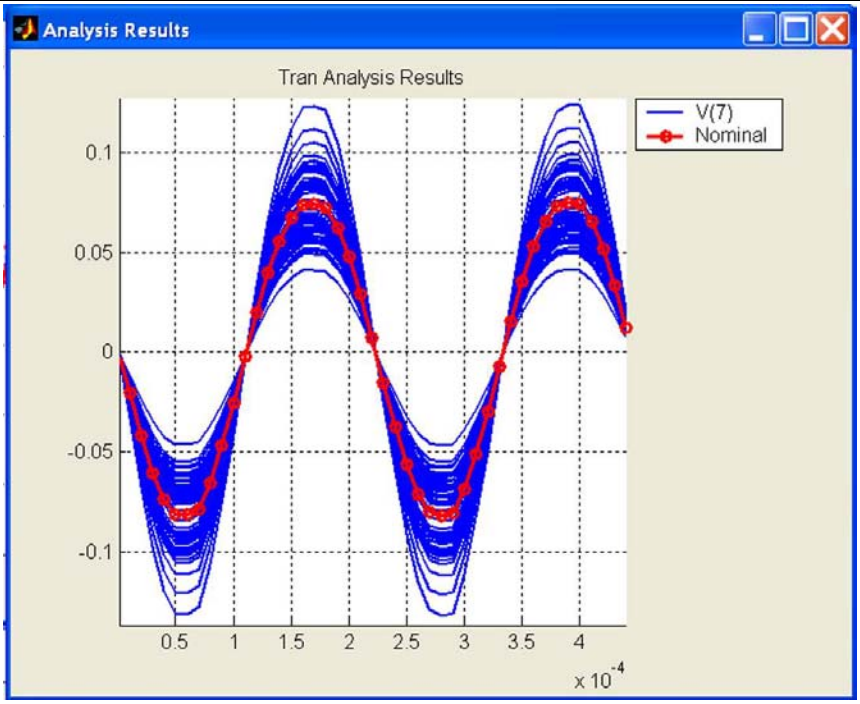


Рис. 6. Результаты параметрического моделирования резисторного усилительного каскада в переходном режиме (число итераций равно 50).

Моделирование неисправностей позволяет изучить виды неисправностей аналоговых схем и их влияние на работу устройства. Неисправности аналоговых схем бывают двух видов — катастрофические и параметрические. Катастрофические, или жесткие неисправности являются причиной появления в схеме короткого замыкания или обрыва цепи. Параметрические, или мягкие неисправности являются следствием отклонения параметра компонентов за границы предельно допустимой области — допуска.

В системе *TeDiAC* моделирование влияния обоих видов неисправностей на работу аналоговых схем выполняется на схемотехническом уровне. В этом случае пользователь выбирает из входного описания схемы те компоненты, в которых предполагается присутствие неисправности, а также указывает определенный вид неисправности. Для параметрических неисправностей пользователь задает величину отклонения параметра выбранного компонента от его номинального значения. Параметрические неисправности формируют путем изменения номинального значения параметра неисправного компонента на величину заданного отклонения с учетом его знака. Катастрофические неисправности моделируют с использованием резистивного элемента с малым значением сопротивления для короткого замыкания и с большим значением сопротивления для обрывов. Резистивный элемент, соответствующий выбранному типу неисправности, помещается на место дефектного компонента. В исследуемой схеме могут рассматриваться как одиночные, так и множественные неисправности.

Пользователь, указывая в схеме неисправности того или иного вида, получает возможность непосредственно на графике проследить их влияние на работу устройства и оценить изменения, которые они вызвали в контролируемых выходных характеристиках. На рис. 7, 8 и 9 представлены результаты моделирования двукратной неисправности (C_1 —20%; R1 +25%) и ее влияние на различные характеристики выходного напряжения.

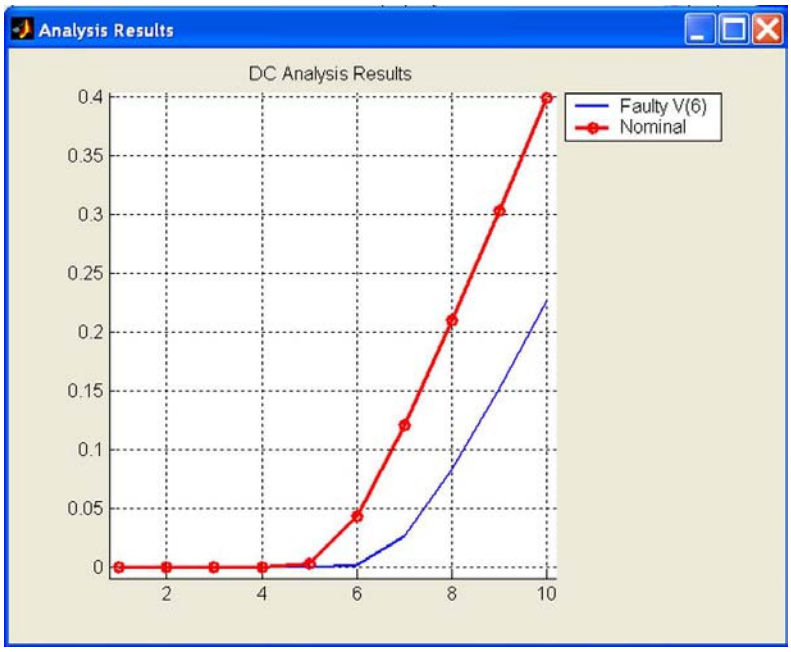


Рис. 7. Результаты моделирования неисправности в статическом режиме.

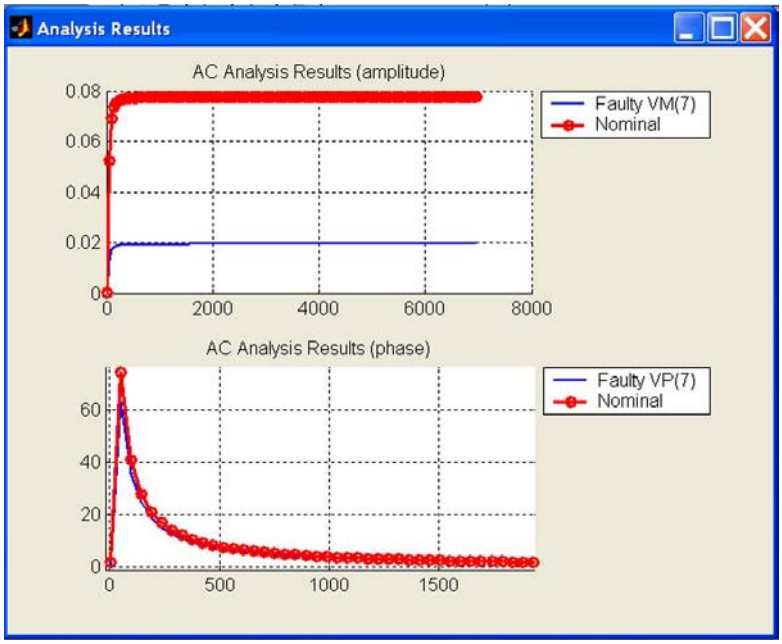


Рис. 8. Результаты моделирования неисправности в малосигнальном режиме.

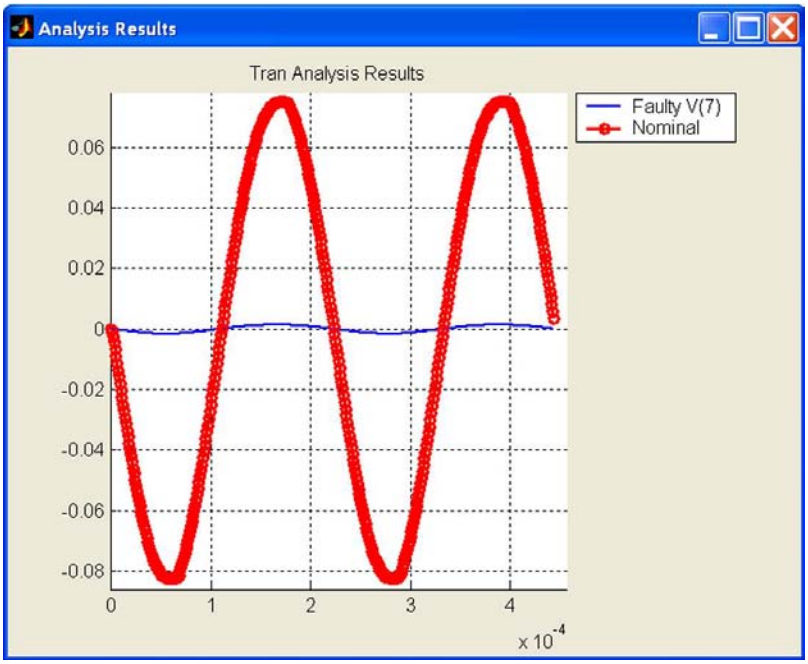


Рис. 9. Результаты моделирования неисправности в переходном режиме.

Анализ тестопригодности. Алгоритмические меры тестопригодности реализуются программно и позволяют получить оценки тестопригодности путем анализа топологического описания схемы. Достоинство этих мер заключается в возможности качественно оценить тестопригодность каждого узла схемы, что позволяет построить сечение схемы по уровню тестопригодности. Сравнение различных узловых значений тестопригодности позволяет легко определить области с невысокой тестопригодностью и оценить эффективность различных методов ее улучшения [2].

При оценке тестопригодности устройства рассматриваются две характеристики внутренних узлов схемы — *управляемость* и *наблюдаемость*, которые обеспечивают количественную оценку сложности управления и наблюдения за величиной сигнала во внутренних узлах схемы, а также характеристика аналоговых компонентов — *коэффициент передачи тестопригодности*.

Под *управляемостью* понимают относительную сложность обеспечения определенного значения сигнала в узле схемы. Данная характеристика, как правило, нормализуется и принимает значение из диапазона от 0 до 1, причем, единица соответствует полной управляемости, а нуль — полной неуправляемости узла. Первичные входы устройства всегда рассматриваются, как полностью управляемые.

Под *наблюдаемостью* понимают относительную сложность распространения ошибки от внутренних узлов схемы до ее первичных выходов. Данная характеристика также нормализуется и принимает значение от 0 до 1, причем, единица соответствует полной наблюдаемости, а нуль

полному отсутствию наблюдаемости узла. Первичные выходы устройства всегда рассматриваются, как полностью наблюдаемые.

Коэффициент передачи тестопригодности (КПТ) позволяет определять, как управляемость и наблюдаемость влияют на маршрут прохождения тестовой информации, которая передается от одних компонентов к другим или к первичным выходам. Коэффициент передачи тестопригодности компонента отражает, во-первых, легкость прохождения произвольного сигнала к выходам компонента при управлении его входами, и, во-вторых, легкость определения сигнала, присутствующего на входах компонента, по результатам исследования значений сигнала на его выходах.

В этом случае компоненты рассматриваются, как самостоятельные элементы, которые проектировщик может использовать, не имея доступа к их внутренней структуре. В качестве примера таких элементов можно отметить резисторы, транзисторы, операционные усилители, компараторы, преобразователи и др.

Метод анализа тестопригодности в подсистеме *TeDiAC* основан на рассмотрении аналоговой схемы в виде совокупности проводящих путей, образованных внутренними компонентами и обеспечивающих распространение сигнала от входа устройства к его выходам [11]. Каждый проводящий путь характеризуется в зависимости от режима функционирования схемы определенным значением сопротивления R, или, в общем случае, $Z(\omega)$, где Z — полное сопротивление, ω — частота входного сигнала. Величина сопротивления влияет на значения наблюдаемости и управляемости. Так, наблюдаемость и управляемость узлов *а* и *в*, между которыми подключен элемент с сопротивлением $Z(\omega)$, увеличиваются, когда величина этого сопротивления приближается к нулю (предельный случай — наличие короткого замыкания), и уменьшаются, когда $Z(\omega)$ принимает предельно большое значение (предельный случай — наличие разрыва). Характер данной зависимости описывается следующим выражением:

$$T_f(Z) = 1 - \frac{Z(\omega)}{OC}, \qquad (1)$$

где $Z(\omega)$ — реальное значение сопротивления компонента, OC — величина сопротивления, обеспечивающего условие обрыва цепи (обычно принимается равной 10 МОм). Выражение (1) позволяет рассчитывать коэффициент передачи тестопригодности T_f (КПТ) пассивных компонентов (сопротивлений, емкостей и индуктивностей).

Расчет КПТ активных компонентов выполняется с использованием эквивалентных схем замещения данных устройств, применяемых в аналоговой схемотехнике. Математическая модель активного компонента представляется совокупностью пассивных компонентов и управляемых источников тока или напряжения.

Управляемость любого *i*-го узла

$$C_i = \frac{1}{F_{in}} \sum_{m=1}^{F_{in}} C_m \left(T_f\right)_m , \tag{2}$$

где C_i — управляемость узла i; F_{in} — коэффициент объединения по входу в узле i; C_m — входная управляемость m-го соединения из числа объединенных в узле i; $\left(T_f\right)_m$ — КПТ компонента m-го соединения из числа объединенных в узле i.

Наблюдаемость произвольного i-го узла

$$O_i = \frac{1}{F_{out}} \sum_{m=1}^{F_{out}} O_m \left(T_f\right)_m , \tag{3}$$

где O_i — наблюдаемость узла i; F_{out} — коэффициент разветвления по выходу в узле i; O_m — входная наблюдаемость m-го соединения из числа разветвляющихся в узле i; $\left(T_f\right)_m$ — КПТ компонента m-го соединения из числа разветвляющихся в узле i.

Поскольку малые значения либо управляемости, либо наблюдаемости приводят к ухудшению тестопригодности устройства, то для оценки тестопригодности предлагается использовать следующую функцию от этих двух характеристик:

$$T_i = \sqrt{C_i \cdot O_i} , \tag{4}$$

где T_i — тестопригодность узла i; C_i — управляемость узла i; O_i — наблюдаемость узла i. Внутренние узлы схемы, обладающие высокими значениями тестопригодности, рекомендуется использовать в качестве дополнительных тестовых узлов устройства. Контроль выходных характеристик схемы относительно этих узлов повышает вероятность обнаружения возможных неисправностей.

Общий показатель тестопригодности всей схемы может быть представлен как среднее арифметическое значение тестопригодности всех узлов схемы, т. е.

$$T = \frac{1}{N} \sum_{i=1}^{N} T_i , \tag{5}$$

где T — тестопригодность всей схемы; T_i — тестопригодность узла i; N — число узлов схемы.

В качестве первичных входов схемы рассматривают узлы, к которым подключены источники тока или напряжения, а также узел «земля». Выбор непосредственно управляемых узлов в системе *TeDiAC* осуществляется автоматически. Наблюдаемые узлы задает пользователь.

Анализ тестопригодности в системе *TeDiAC* может быть выполнен в двух альтернативных режимах. В ходе первого на одной фиксированной частоте входного сигнала происходит расчет тестопригодности всех выбранных пользователем узлов исследуемой схемы. В результате формируются гистограммы, отражающие значения управляемости, наблюдаемости

и тестопригодности данных узлов. На рис. 10 представлены результаты анализа тестопригодности резисторного усилительного каскада, проведенного для всех внутренних узлов при частоте входного сигнала 4.5 кГц.

В ходе второго режима исследуется зависимость уровня тестопригодности определенного узла от изменения частоты входного сигнала. Пользователь выбирает узел исследуемой схемы и задает диапазон изменения частоты входного сигнала и число частотных точек внутри него. В результате формируются графики зависимости управляемости, наблюдаемости и тестопригодности заданного узла от частоты сигнала, подаваемого на вход схемы. На рис. 11 представлены результаты анализа тестопригодности резисторного усилительного каскада, проведенного для внутреннего узла номер 5 при изменении частоты входного сигнала в диапазоне от 1 Гц до 5000 Гц, число рассмотренных частотных точек равно 50.

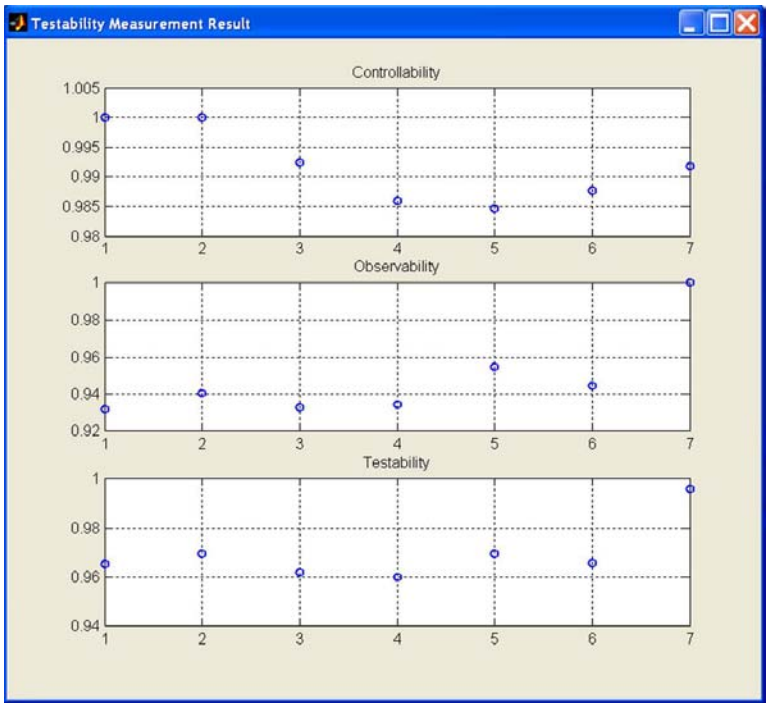


Рис. 10. Результаты анализа тестопригодности внутренних узлов усилительного каскада при частоте входного сигнала 4.5 кГц

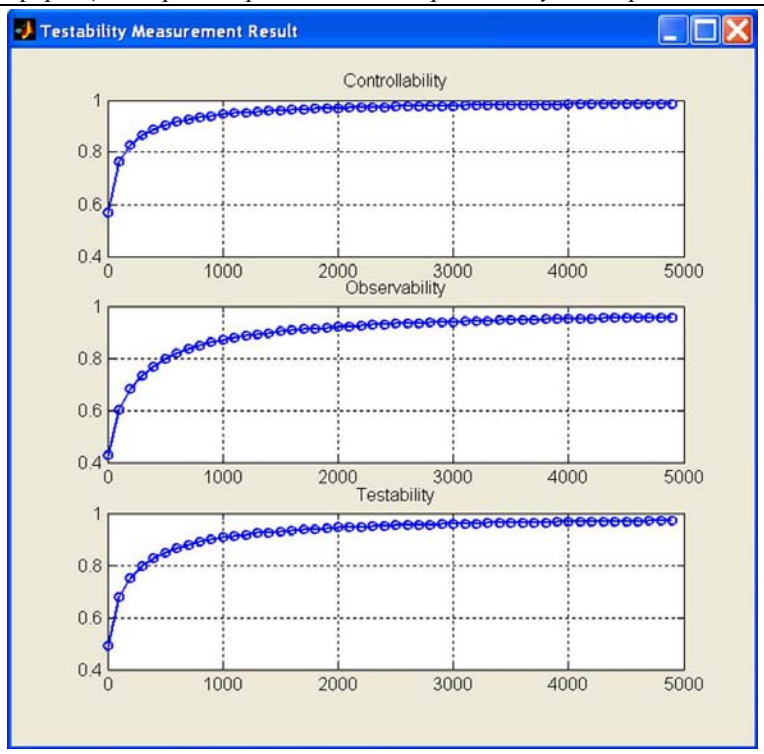


Рис. 11. Результаты анализа тестопригодности усилительного каскада в узле 5 при частоте входного сигнала из диапазона 1 Гц до 5 кГц

Анализ тестопригодности позволяет количественно оценить как изменения схемного проекта исследуемого устройства или использование дополнительного числа тестовых узлов влияют на легкость и качество будущего тестирования проектируемой схемы.

Выбор тестовых узлов может быть выполнен двумя способами, во-первых, по результатам анализа тестопригодности и, во-вторых, на основе анализа чувствительности.

Анализ тестопригодности позволяет определить набор узлов с высоким уровнем тестопригодности. Данные узлы можно использовать в качестве тестовых, поскольку они являются легко управляемыми и/или легко наблюдаемыми, а значит, обеспечивают простоту измерения относительно них контролируемых характеристик выходного напряжения. Так, для резисторного усилительного каскада (рис. 1) в качестве тестового узла можно использовать узел номер 5, поскольку он обладает максимальным значением тестопригодности среди внутренних узлов схемы, т.е. таких, которые не являются первичными входами или выходами.

При анализе схем под *чувствительностью* понимают влияние изменения параметров компонентов и внешних условий работы схемы на ее выходные реакции или передаточную функцию. Количественная оценка влияния некоторого компонента или характеристики схемы (сопротивление, крутизна, температура, геометрические размеры и т.п.) на изменение выходной функции схемы (напряжение или ток, входной или выходной

импеданс, коэффициент передачи и т.п.) определяется следующим выражением [12]

$$S_x^y = \frac{\partial y}{\partial x}, \tag{6}$$

где S_x^y — функция чувствительности, y — дифференцируемая функция схемы (выходной параметр), x — внутренний параметр. Другими словами под чувствительностью понимается производная выходной функции схемы по ее параметрам.

Проведение анализа чувствительности на ранних стадиях проектирования аналоговой схемы позволяет получить информацию, необходимую для выбора тестовых узлов. В ходе моделирования проектируемой схемы вычисляются коэффициенты чувствительности выходных параметров схемы к изменениям параметров внутренних компонентов. Данные коэффициенты рассчитываются для всех компонентов и всех внутренних узлов схемы. Для сложных устройств расчет может проводиться лишь для некоторого набора внутренних узлов, определяемого проектировщиком в соответствии с особенностями каждого конкретного приложения [13, 14].

В результате моделирования для каждого контролируемого параметра формируется матрица $\mathbf{S} \subset \Re(m, n)$, где m — число внутренних компонентов, n — число рассмотренных внутренних узлов. Элементами данной матрицы ($S_{i,j}$) являются коэффициенты чувствительности выходного параметра схемы, контролируемого относительно узла j, к отклонению параметра компонента i. Процесс выбора тестовых узлов сводится к поиску таких столбцов матрицы **S**, которые включают наибольшее число максимальных значений коэффициентов чувствительности каждой строки. То есть, к поиску минимального покрытия внутренних узлов максимальными значениями коэффициентов чувствительности всех компонентов. Внутренние узлы схемы, вошедшие в данное минимальное покрытие, определяются в качестве тестовых узлов. Данный алгоритм обеспечивает поиск тестовых узлов, выходные отклики относительно которых предоставляют полную информацию о работоспособности схемы. При этом считается, что на вход схемы подается определенное входное воздействие.

На рис. 12 представлен график в виде трехмерной поверхности, отображающий матрицу **S**, полученную в системе *TeDiAC* для резисторного усилительного каскада при частоте входного сигнала 4.5 кГц. Из графика видно, что наибольшее число максимальных значений коэффициентов чувствительности каждой строки (для каждого компонента) располагается в двух узлах схемы — 5 и 7. Однако узел 7 — это первичный выход усилителя, который уже является тестовым узлом, поэтому узел номер 5 может быть включен во множество тестовых узлов схемы усилительного каскада. Таким образом, выбор тестовых узлов на основе анализа тестопригодности и анализа чувствительности позволяет получить одинаковые результаты.

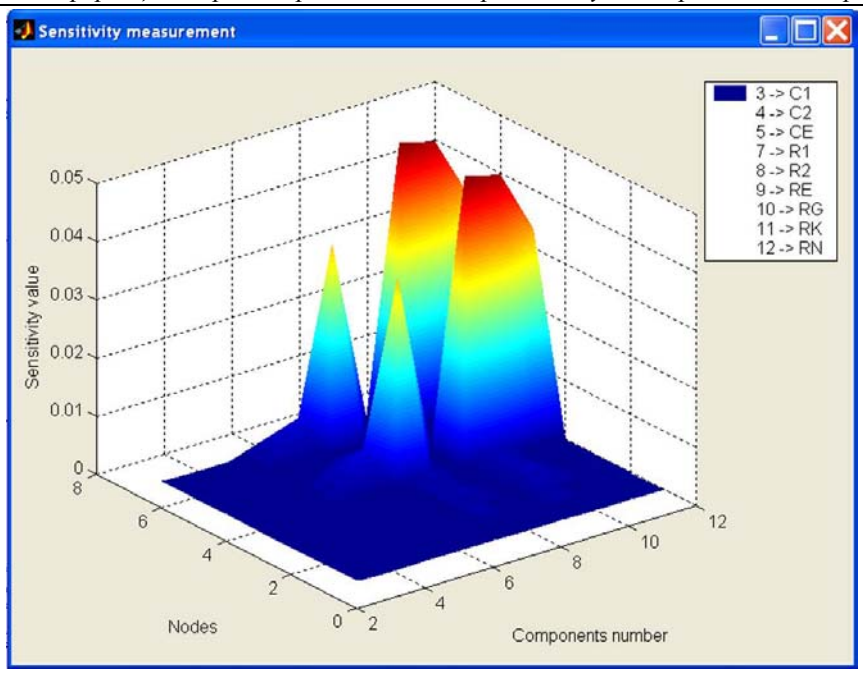


Рис. 12. Результаты анализа чувствительности усилительного каскада при частоте входного сигнала 4.5 кГц

Выбор тестовых частот может быть выполнен двумя способами, во-первых, по результатам анализа тестопригодности и, во-вторых, на основе анализа чувствительности.

В первом случае происходит поиск таких частот входного сигнала, при которых тестопригодность схемы в тестовом узле принимает максимальные значения. При тестировании устройства можно использовать весь набор частот или его подмножество для формирования входных тестовых сигналов.

Второй способ основан на применении функции чувствительности, которая, как и выходные характеристики устройства, зависит от параметров входного сигнала (например, от амплитуды, постоянной составляющей или частоты, и др.). С изменением параметров входного сигнала значение коэффициентов чувствительности (наблюдаемости) может либо увеличиваться, либо уменьшаться. Данная особенность широко используется при выборе оптимального тестового вектора, т. е. набора входных тестовых сигналов, обеспечивающих однозначное выявление максимального числа возможных в схеме неисправностей. Использование вместо одного тестового воздействия набора входных тестовых сигналов, обеспечивающих функционирование устройства в различных режимах, позволяет повысить вероятность обнаружения неисправностей. Это становится возможным за счет увеличения количества информации о работоспособности схемы, получаемой при измерении контролируемых параметров. В этом случае появляется возможность сократить набор тестовых узлов без потери качества последующего тестирования. При использовании набора входных тесто-

вых воздействий $\mathbf{T} \subset \Re(p)$ формируется совокупность $\mathbf{S}$ матриц $S_{i,j}^{k}$, каждая из которых получена для определенного сигнала $T_k \in \mathbf{T}$, $1 \le k \le p$. Минимизация набора тестовых узлов осуществляется путем поиска минимального покрытия заявленных узлов схемы максимальными значениями коэффициентов чувствительности всех компонентов, рассчитанными при различных входных сигналах из тестового вектора [14].

Для резисторного усилительного каскада в качестве результата выбора тестовых частот получаем набор из пяти частот — 14 Гц, 40 Гц, 53 Гц, 79 Гц и 5864 Гц, которые обеспечивают максимальные значения коэффициентов чувствительности в тестовом узле для каждого компонента. В качестве тестового узла был рассмотрен узел номер 5. Результаты проведенного расчета коэффициентов чувствительности на частотах из рабочего диапазона резисторного усилительного каскада приведены на рис. 13.

Несколько слов о средстве разработки подсистемы *TeDiAC*. Использование системы MATLAB открывает широкие возможности по решению задач различных проблемных областей. Обладая развитым математическим аппаратом, система позволяет пользователю решать свои задачи, отрабатывать новые идеи и подходы, не отвлекаясь на реализацию отдельных математических преобразований как элементарных, так и специфических. Обширная библиотека инструментальных средств (*toolboxes*) предоставляет широкий набор специализированных математических функций, необходимых для решения разнообразных технических задач.

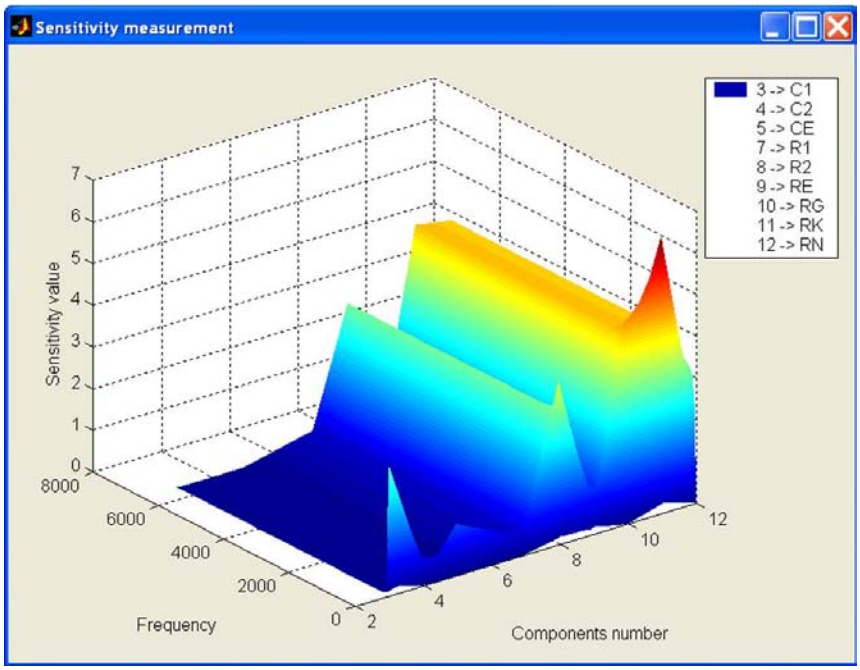


Рис. 13. Результаты выбора тестовых частот для усилительного каскада с использованием анализа чувствительности

На личном опыте доказано, что легкость использования данного инструментария и многообразие предоставляемых возможностей позволяют реализовывать различные приложения, существенно сокращая общее время их разработки, не уступая, а зачастую даже выигрывая в качественных характеристиках получаемых подсистем, по сравнению с их реализациями на языках программирования высокого уровня. Данные обстоятельства обуславливают систему *MATLAB* как незаменимое средство, используемое для реализации некоммерческих подсистем САПР РЭА, выполнения исследовательских работ и обучения.

Литература

1. *Mosin S. G.* Introduction to analog circuits testing and diagnosis // Автоматизированные системы управления и приборы автоматики. Выпуск 122, Харьков, 2003.— С.104–119.
2. *Мосин С. Г.* Подходы тестопригодного проектирования аналоговых интегральных схем // Радиоэлектроника и информатика.— 2003.— №1.— С.49–59.
3. Test Designer. A software program from Intusoft.— Intusoft Newsletter, 1998.
4. DesignMaxx, FaultMaxx, FaultMaxx. Opmaxx Datasheet.— Copyright Opmaxx Inc. 8209, 1998.
5. *Saab K., Marche D., Hamida N.B., Kaminska B.*, LIMSoft: automated tool for sensitivity analysis and test vector generation // IEE Proc. Circuits Devises Syst.— V.143.— N.6.— 1996.— P.386–392.
6. *Hoffmann C.* A New Design Flow and Testability Measure for the Generation of a Structural Test and BIST for Analogue and Mixed-Signal Circuits // Proc. of IEEE Design, Automation and Test in Europe, Munich, 2002.— P.197–204.
7. *You Z., Sanchez S. E.* Analog system-level fault diagnosis based on a symbolic method in the frequency domain // IEEE Trans. on Instrumentation and Measurement.— V.44.— N.1.— Feb.1995.— P.28–35.
8. *Мосин С. Г., Рудаков О. В., Лобачев Г. А.* Подсистема тестирования аналоговых и смешанных интегральных схем // Конверсия, приборостроение, медицинская техника: Материалы междунар. науч.-техн. конф.— Владимир: ВлГУ, 1999.— С.133–135.
9. *Мосин С. Г., Морозов М. А.* Подсистема получения тестовых воздействий для линейных аналоговых схем // Современные информ. технологии в образовательном процессе и науч. исслед.: Сб. ст. конф.— Шуя: «Весть», 2000.— С.34–35.
10. *Mosin S. G.* Educational purpose CAD tool for testing and diagnosis of analog circuits: fault simulation // Proc. of East-West Design and Test Conference.— Crimea, Ukraine, 2003.— P.87–90.

11. *Huyngh S.D., Kim S., Soma M.* Automatic Analog Test Signal Generation Using Multifrequency Analysis // IEEE Trans. on Circuit and Systems–II: Analog and Digital Signal Processing.— V.46.— N.5.— 1999.— P.565–576.
12. *Гехер К.* Теория чувствительности и допусков электронных цепей. Пер. с англ. Под ред. Ю. Л. Хотунцева.— М.: Сов. радио, 1973.— 200 с.
13. *Мосин С. Г., Ланцов В. Н.* Методика мультичастотной функциональной диагностики аналоговых и смешанных интегральных микросхем // Компьютерные технологии обработки и анализа данных / Под ред. С. С. Садыкова, Р. С. Садуллаева.— Ташкент: НПО «Кибернетика» АН РУз, 2000.— С.160–168.
14. *Mosin S., Lantsov V.* A New opportunity of using sensitivity function for functional testing // Proc. of 2nd Electronic Circuits and Systems Conference (ECS'01).— Slovakia: Bratislava, 2001.— 4 p.

УДК 510+519.24/27:62-50

ИССЛЕДОВАНИЕ ХАРАКТЕРИСТИК КЛАССИЧЕСКИХ ПЛАНОВ ЭКСПЕРИМЕНТОВ МЕТОДАМИ МОДЕЛИРОВАНИЯ В СРЕДЕ MATLAB

Наумов А. А., Сенич В. В.
Новосибирский государственный технический университет, Новосибирск,
e-mail: naumov@fb.nstu.ru, senich@fb.nstu.ru;

Постановка задачи

Цель настоящего исследования заключается в том, чтобы показать насколько эффективны и устойчивы классические процедуры планирования экспериментов по отношению к априорным предположениям (гипотезам) относительно свойств случайных ошибок регрессионных моделей. В частности, исследуется влияние таких предположений на оптимальность планов экспериментов. Показано, что фактическая (апостериорная) дисперсия оценки функции отклика и дисперсия оценки вектора параметров (и соответствующие значения критериев оптимальности планов экспериментов) могут сильно отличаться от априорных дисперсий.

Рассмотрим уравнение наблюдения за объектом (или системой) вида:

$$y(x) = \sum_{i=1}^{k} \theta_i f_i(x) + \varepsilon(x), \qquad (1)$$

где $E\left(y(x)\middle|x\right) = \sum_{i=1}^{k} \theta_i f_i(x) = \theta^T f(x)$ — уравнение регрессии (регрессионная модель, функция отклика), $\varepsilon(x)$ — случайная переменная со свойствами $E\left(\varepsilon(x)\right) = 0, E\left(\varepsilon^2(x)\right) = \sigma^2, \sigma > 0$ (здесь E — символ математического ожидания), $f^T(x) = \left(f_1(x), f_2(x), .., f_k(x)\right)$ — базисный вектор регрессии (вектор непрерывных на области экспериментирования X функций, $X \in R^p$), x — вектор размерности p входных контролируемых переменных изучаемого объекта, $y(x)$ — выходная наблюдаемая переменная объекта, $\theta^T = \left(\theta_1, \theta_2, .., \theta_k\right)$ — вектор неизвестных параметров модели. Пусть наблюдения за объектом производятся в точках $x_1, x_2, ..., x_k$ (в точках так называемого спектра плана экспериментов, множества $X_{Sp} = \left\{x_1, x_2, ..., x_m\right\}$) в количестве $n_1, n_2, ..., n_m$ наблюдений в каждой из точек спектра плана со-

ответственно. Положим $N = \sum_{i=1}^{m} n_i$, $M(\xi) = \sum_{i=1}^{m} n_i f(x_i) f^T(x_i)$,

$Y = \sum_{i=1}^{m} n_i f(x_i) \bar{y}_i$, $\bar{y}_i = \frac{1}{n_i}(\sum_{j=1}^{n_i} y_{ij})$. Тогда в соответствии с обычным методом наименьших квадратов, оценки параметров θ можно найти в соответствии с формулой:

$$\hat{\theta} = M^{-1}(\xi) Y . \qquad (2)$$

Планом эксперимента (ПЭ) называется набор, состоящий из точек спектра плана и так называемых весов, в соответствии с которыми распределяется общее число экспериментов N по этим точкам. Вид плана экспериментов следующий:

$$\xi = \begin{Bmatrix} x_1, & x_2, & \dots & , x_m \\ p_1, & p_2, & \dots & , p_m \end{Bmatrix},$$

где

$$\xi \in \Xi = \left\{ \xi = \begin{Bmatrix} x_1, & x_2, & \dots & , x_m \\ p_1, & p_2, & \dots & , p_m \end{Bmatrix} \middle| \; x_i \in X, p_i = (n_i / N) \geq 0, i = 1,2,...,m, \sum_{i=1}^{m} p_i = 1 \right\} .$$

Суть задачи построения наилучшего (оптимального) плана экспериментов состоит в нахождении на множестве допустимых планов экспериментов Ξ такого, который максимизировал бы (или минимизировал) некоторый функционал от точностной характеристики оценок параметров модели (или оценок самой модели). Например, если план эксперимента получается в соответствии с решением задачи максимизации некоторого критерия от матрицы $M(\xi)$ (так называемой информационной матрицы Фишера), т.е. максимизируется некоторый критерий от нее $\Phi(M(\xi))$, то соответствующий план называется *Ф*-оптимальным. Более точно, *Ф*- оптимальный план получается как решение следующей оптимизационной задачи (см., например, [1],[2],[3] и др.):

$$\Phi(M(\xi^*)) = \max_{\xi \in \Xi} \Phi(M(\xi)) . \qquad (3)$$

Часто на практике вместо задачи (3) решают эквивалентную ей задачу:

$$\Phi(M^{-1}(\xi^*)) = \min_{\xi \in \Xi} \Phi(M^{-1}(\xi)) . \qquad (4)$$

Здесь матрица $M^{-1}(\xi)$ является обратной для матрицы $M(\xi)$ и представляет собой с точностью до некоторой постоянной величины дисперсионную матрицу оценок параметров модели, $\Phi(M^{-1}(\xi))$ - выпуклый вниз на множестве матриц $M^{-1}(\xi), \xi \in \Xi$, функционал. Этот функционал называют также критерием оптимальности (эффективности) планов экспериментов.

Рассмотрим в качестве примера частный случай критерия $\Phi(M^{-1}(\xi))$.

***D*-оптимальность.**

В этом случае для нахождения *D*-оптимального плана решается задача:

$$\det(M^{-1}(\xi^*)) = \min_{\xi \in \Xi} (\det(M^{-1}(\xi))). \quad (5)$$

Здесь $\det(M^{-1}(\xi))$ — определитель матрицы $M^{-1}(\xi)$, которая, как было отмечено выше, с точностью до множителя σ^2 равна дисперсионной матрице оценок параметров модели регрессии.

Теорема 1 (эквивалентности) [2].

Оптимальный план эксперимента ξ^*, полученный как результат решения задачи (5) и план эксперимента — решение задачи

$$\xi^* = Arg\left(\min_{\xi} \max_{x} \hat{d}_{\eta}(x,\xi)\right), \quad (6)$$

где $\hat{d}_{\eta}(x,\xi) = \sigma^2 f^T(x) M^{-1}(\xi) f(x)$, совпадают. При этом для оптимального плана ξ^* выполняется равенство:

$$\max_{x} \hat{d}_{\eta}(x,\xi^*) = \sigma^2 k / N. \quad (7)$$

Сделаем важное замечание. Функция $\hat{d}_{\eta}(x,\xi)$ представляет собой дисперсию оценки модели регрессии. В то же время, эта функция выступает в роли фундаментальной (основной) функции, которая используется в алгоритмах синтеза *D*-оптимальных классических ПЭ. Аналогично, в общем случае для задач построения классических Φ-оптимальных планов экспериментов в качестве такой функции выступает функция $\hat{d}(x,\xi)$, имеющая один из следующих видов (см., например, [3], [10], [13] и др.):

в случае *D*-оптимального планирования экспериментов, $\det(M^{-1}(\xi))$ (или $\det(\sigma^2 M^{-1}(\xi))$) — вид критерия оптимальности, $\hat{d}(x,\xi) = f^T(x) M^{-1}(\xi) f(x)$ (т.е. $\hat{d}(x,\xi) = \frac{1}{\sigma^2} \hat{d}_{\eta}(x,\xi)$);

в случае *A*-оптимального планирования, $SpM^{-1}(\xi)$ (или $Sp(\sigma^2 M^{-1}(\xi))$) — критерий оптимальности, $\hat{d}(x,\xi) = f^T(x) M^{-2}(\xi) f(x)$;

в случае *I*-оптимального планирования экспериментов, $\int_{X_0} f^T(x) M^{-1}(\xi) f(x) p(x) dx$ (или $\sigma^2 \int_{X_0} f^T(x) M^{-1}(\xi) f(x) p(x) dx$)

$(p(x) \geq 0, x \in X_0 \subseteq X, \int_{X_0} p(x) dx = 1)$ –

критерий оптимальности, $\hat{d}(x,\xi) = \int_{X_0} \left(f^T(x) M^{-1}(\xi) f(\tilde{x})\right)^2 p(\tilde{x}) d\tilde{x}$;

в случае *C*-оптимального планирования экспериментов, $C^T M^{-1}(\xi)C$ ($C \in \mathrm{M}_{(1\times k)}$ – множество $k \times k$ матриц) (или $\sigma^2 C^T M^{-1}(\xi)C$)— критерий оптимальности, $\hat{d}(x,\xi) = f^T(x)M^{-1}(\xi)\cdot C \cdot C^T \cdot M^{-1}(\xi)f(x)$;

в случае *G*-оптимального планирования, $\max\limits_{x\in X} f^T(x)M^{-1}(\xi)f(x)$ (или $\max\limits_{x\in X}\sigma^2 f^T(x)M^{-1}(\xi)f(x)$) - критерий оптимальности, $\hat{d}(x,\xi) = f^T(x)M^{-1}(\xi)\cdot f(x)$ и т.д. Везде ниже такие функции для задачи классического планирования экспериментов обозначены как $\hat{d}(x,\xi)$. Таким образом, для задач *D*- и *G*-оптимального классического ПЭ выполняется равенство:

$$\hat{d}_\eta(x,\xi) = \sigma^2 \hat{d}(x,\xi).$$

Отметим, что для других (вышеперечисленных) критериев оптимальности (кроме *D*- и *G*- критериев) это равенство не выполняется и $\hat{d}_\eta(x,\xi)$ и $\hat{d}(x,\xi)$ — это две различные (не связанные таким простым соотношением) функции. Заметим, что аналогичным образом везде ниже через $\tilde{d}_\eta(x,\xi,y)$ и $\tilde{d}(x,\xi,y)$ будем обозначать дисперсию оценки регрессионной модели и функцию, используемую при синтезе эффективных стратегий управления экспериментом, соответственно. Соответствующие различным критериям эффективности *Ф* функции $\tilde{d}(x,\xi,y)$ (в задачах эффективного управления экспериментом) будут введены в рассмотрение и проанализированы ниже. Следует отметить, что функции $\hat{d}(x,\xi)$ (в задачах ПЭ при использовании классического подхода) и соответствующие им функции $\tilde{d}(x,\xi,y)$ (в задачах эффективного управления экспериментом) играют важную роль в алгоритмах синтеза классических оптимальных планов и эффективных стратегий управления экспериментами соответственно.

1. Исследование дисперсии оценок регрессии и апостериорная информация

Проверим робастность (устойчивость) классических *D*- *(G*-) оптимальных планов, для которых выполняется равенство (7), по отношению к некоторым исходным (априорным) предположениям задачи синтеза этих планов. Здесь мы исследуем зависимость характеристик *D*-оптимальных планов от свойств случайной составляющей модели $\varepsilon(x)$. Так, в частности, ответим на вопрос: насколько сильно влияет на оптимальность (или неоптимальность) плана использование апостериорных оценок дисперсий

ошибок наблюдений в каждой из точек спектра плана. Заметим, что при этом, по-прежнему, предполагается выполнение условий $E\left(\varepsilon(x)\right)=0$ и $E\left(\varepsilon^2(x)\right)=\sigma^2$.

Рассмотрим подробнее одну из возможных практических ситуаций. Предположим, что обрабатываются данные, полученные в соответствии с *D*-оптимальным планом ξ^* (выбранным, например, из [4]). В этом случае, поскольку выполняются условия $E\left(\varepsilon(x)\right)=0$ и $E\left(\varepsilon^2(x)\right)=\sigma^2$, экспериментатор вправе оценивать параметры регрессионной модели в соответствии с обычным (не взвешенным) методом наименьших квадратов по формуле :

$$\hat{\theta}=M^{-1}(\xi^*)Y. \tag{8}$$

Затем он может оценить качество построенной модели $\hat{y}(x)=\eta(x,\hat{\theta})=\sum_{i=1}^{k}\hat{\theta}_i f_i(x)=\hat{\theta}^T f(x)$ (оценки функции отклика), вычислив, например, дисперсию этой модели:

$$\hat{d}_\eta(x,\xi^*)=\sigma^2\hat{d}(x,\xi^*)=\sigma^2 f^T(x)M^{-1}(\xi^*)f(x). \tag{9}$$

Причем, согласно теореме эквивалентности, для этой дисперсии должно выполниться равенство:

$$max_x\hat{d}_\eta(x,\xi^*)=\sigma^2 max_x\hat{d}(x,\xi^*)=\sigma^2 k/N. \tag{10}$$

Необходимо отметить, что, обрабатывая экспериментальные данные, полученные в соответствии с *D*-оптимальным планом ξ^* в соответствии с формулами (8), (9) экспериментатор мог использовать и апостериорную информацию, поскольку эти оценки находятся после проведения экспериментов на объекте. Итак, он мог учесть фактические (апостериорные) значения оценок дисперсии ошибок наблюдений в каждой из точек спектра плана, во-первых, при оценивании параметров. А, во-вторых, учесть апостериорную информацию также и при нахождении дисперсии оценки модели.

Если воспользоваться взвешенным методом наименьших квадратов, то вышеприведенные формулы следует изменить следующим образом (см. [3],[13]):

$$\hat{\theta}=(F_1^T D_2^{-1}F_1)^{-1}F_1^T D_2^{-1}\overline{y}, \tag{11}$$

$$\tilde{d}(x,\xi,y)=f^T(x)(F_1^T D_2^{-1}F_1)^{-1}f(x)=f^T(x)M^{-1}(\xi,y)f(x), \tag{12}$$

где

$$\overline{y}=\begin{pmatrix}\overline{y}_1\\ \vdots\\ \overline{y}_m\end{pmatrix},\ \overline{y}_i=\frac{1}{n_i}\sum_{j=1}^{n_i}y_{ij};$$

$$y_{ij}=y_j(x_i)=f^T(x_i)\theta+\varepsilon_j(x_i),\ j=1,2,...,n_i;i=1,2,...,m;$$

$$F_1 = \begin{pmatrix} f^T(x_1) \\ \dots \\ f^T(x_m) \end{pmatrix},$$

$$D_2 = diag\left(\frac{1}{n_i(n_i-1)} \sum_{j=1}^{n_i} (\overline{y}_i - y_{ij})^2 \right) = diag\left(\hat{\sigma}^2_{\overline{y}}(x_i) \right), n_i \geq 2, i = 1,2,...,m.$$

Как сильно влияет на оптимальность плана переход от формул (8) и (9) к формулам (10) и (11)? На много ли ошибся экспериментатор, воспользовавшись классическим оптимальным планом и обработав полученные в результате экспериментирования с системой данные по формулам (8) и (9)? Ниже мы покажем, что разница в значениях дисперсий $\hat{d}_\eta(x,\xi^*) = \sigma^2 \hat{d}(x,\xi^*)$ и $\tilde{d}_\eta(x,\xi^*,y)$ (для *G*- оптимальных планов) и функций $\hat{d}(x,\xi^*)$ и их аналогов — функций $\tilde{d}(x,\xi^*,y)$, определяющих оптимальность планов для любых критериев оптимальности или эффективности, может быть весьма существенной и составлять несколько десятков процентов от значений $\hat{d}_\eta(x,\xi^*)$ или $\hat{d}(x,\xi^*)$.

Сведем в таблицу (см. табл. 1) вышеприведенные и одну новую расчетные схемы, которыми может воспользоваться экспериментатор в соответствии с рекомендациями классического подхода к планированию экспериментов (ПЭ) или с учетом апостериорной информации.

Таблица 1.

N п/п	Особенность расчетной схемы	Основные формулы расчетной схемы
1.	Экспериментатор использует обычный метод наименьших квадратов с учетом выполнения свойств для случайной ошибки модели $E(\varepsilon(x)) = 0$ и $E(\varepsilon^2(x)) = \sigma^2$.	$\hat{\theta} = M^{-1}(\xi)Y$, $\hat{d}_\eta(x,\xi) = \sigma^2 f^T(x) M^{-1}(\xi) f(x)$, $D(\hat{\theta}) = \sigma^2 (M^{-1}(\xi))$, $N = \sum_{i=1}^{m} n_i$, $M(\xi) = \sum_{i=1}^{m} n_i f(x_i) f^T(x_i)$, $Y = \sum_{i=1}^{m} n_i f(x_i) \overline{y}_i$, $\overline{y}_i = \frac{1}{n_i}(\sum_{j=1}^{n_i} y_{ij})$.
2.	Экспериментатор использует взвешенный метод наименьших квадратов с учетом апостериорных значений дисперсий случайной ошибки модели. Выполняются равенства $E(\varepsilon(x)) = 0$ и $E(\varepsilon^2(x)) = \sigma^2$.	$\hat{\theta} = (F_1^T D_2^{-1} F_1)^{-1} F_1^T D_2^{-1} \overline{y}$, $\tilde{d}_\eta(x,\xi) = f^T(x)(F_1^T D_2^{-1} F_1)^{-1} f(x) =$ $= f^T(x) M^{-1}(\xi, y) f(x)$, $D(\hat{\theta}) = (F_1^T D_2^{-1} F_1)^{-1}$,

		$\overline{y} = \begin{pmatrix} \overline{y}_1 \\ \vdots \\ \overline{y}_m \end{pmatrix}$, $\overline{y}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij}$, $y_{ij} = y_j(x_i) = f^T(x_i)\theta + \varepsilon_j(x_i)$, $j = 1,2,...,n_i; i = 1,2,...,m;$ $F_1 = \begin{pmatrix} f^T(x_1) \\ ... \\ f^T(x_m) \end{pmatrix}$, $D_2 = diag\left(\frac{1}{n_i(n_i-1)}\sum_{j=1}^{n_i}(\overline{y}_i - y_{ij})^2\right) =$ $= diag\left(\hat{\sigma}^2_{\overline{y}}(x_i)\right), n_i \geq 2, i = 1,2,...,m$
3.	Экспериментатор использует обычный метод наименьших квадратов для оценивания параметров с учетом выполнения свойств для случайной ошибки модели $E(\varepsilon(x)) = 0$ и $E(\varepsilon^2(x)) = \sigma^2$. Однако, дисперсия оценки модели находится с учетом апостериорных оценок дисперсий ошибок наблюдений в каждой точке спектра плана.	$\hat{\theta} = \left(F_1^T F_1\right)^{-1} F_1^T \cdot \overline{y}$, $\hat{d}_\eta(x,\xi) = f^T(x)M^{-1}(\xi)F_1^T D_2 F_1 \times$ $\times M^{-1}(\xi) f(x)$, $D(\hat{\theta}) = M^{-1}(\xi)F_1^T D_2 F_1 M^{-1}(\xi)$, $F_1 = \begin{pmatrix} f^T(x_1) \\ ... \\ f^T(x_m) \end{pmatrix}$, $\overline{y} = \begin{pmatrix} \overline{y}_1 \\ \vdots \\ \overline{y}_m \end{pmatrix}$, $\overline{y}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij}$, $M(\xi) = F_1^T F_1 = \sum_{i=1}^{m} f(x_i) f^T(x_i)$, $D_2 = diag\left(\hat{\sigma}^2_{\overline{y}}(x_i)\right), i = 1,2,...,m$.

Заметим, что третья расчетная схема часто используется экспериментаторами, которые плохо представляют, каким образом следует обрабатывать результаты экспериментов, хотя сами данные получены ими на основе оптимального плана экспериментов. Конечно, вторая схема является обобщением первой. Но, почему же тогда классические *D*- оптимальные планы в предположении, что дисперсия ошибок наблюдений является постоянной во всех точках области экспериментирования *X* величиной (или известна с точностью до некоторой константы), синтезируются (строятся) для первой схемы? Ответ очень простой. Он заключается в том, что для

второй схемы характерен очень неприятный с точки зрения построения алгоритмов и процедур синтеза оптимальных планов момент: дисперсионная матрица $D(\hat{\theta}) = (F_1^T D_2^{-1} F_1)^{-1}$, а, значит, и критерий оптимальности $\Phi(D(\hat{\theta})) = \Phi((F_1^T D_2^{-1} F_1)^{-1})$, зависят от фактических значений наблюдений за выходной переменной исследуемого объекта (или системы). То есть для второй схемы характерна зависимость синтезируемого (еще только строящегося) плана от будущих результатов наблюдений. Классическая наука оптимального планирования экспериментов (КОПЭ) при синтезе оптимальных планов использует априорное значение дисперсии случайной ошибки измерений (см. [1],[2],[3],[10],[13] и многие другие работы). И, таким образом, проблема была снята. В целом, такое упрощение сыграло положительную роль на этапе изучения асимптотических свойств классических оптимальных планов, построения соответствующих алгоритмов, создания программ для ЭВМ, разработки математического аппарата. Но, на наш взгляд, наступает новый этап развития науки управления экспериментом, когда нужно осмыслить тот факт, что классические оптимальные планы экспериментов на самом деле, в общем случае, таковыми могут не являться. Ниже на многочисленных примерах иллюстрируется неоптимальность (мы будем говорить — неэффективность) классических планов.

Рассмотрим пример.

Пусть базисный вектор модели имеет вид $f^T(x) = (1, x, x^2, x^3)$, а вектор истинных значений параметров — $\theta^T = (2.0, 5.0, -3.0, 4.0)$ (т.е. количество параметров модели *k=4*). Для области экспериментирования $X = [-1.0; 9.0]$ подберем соответствующий виду модели *D*- оптимальный (в соответствии с теоремой эквивалентности он же и *G*- оптимальный) план ξ^* [4]:

$$\xi^* = \begin{Bmatrix} -1.0 & 1.765 & 6.235 & 9 \\ n_1 & n_2 & n_3 & n_4 \end{Bmatrix}$$ ($n_1 = n_2 = n_3 = n_4$, число точек спектра плана равно *m=4*).

Число измерений в каждой точке спектра плана должно быть одинаковым. И пусть, например, оно равно четырем, т.е. $n_i = 4, i = 1,2,3,4$. Напомним еще раз основные расчетные формулы, которые используются при моделировании. В предположении, что $E(\varepsilon(x)) = 0$ и $E(\varepsilon^2(x)) = \sigma^2$, экспериментатор оценивает параметры модели по методу наименьших квадратов (МНК) следующим образом:

$$\hat{\theta} = (\overline{F}_1^T \overline{F}_1)^{-1} \overline{F}_1^T \overline{y},$$

при этом

$$\hat{d}_\eta(x,\xi)=\sigma^2 f^T(x)M^{-1}(\xi)f(x);\ D(\hat{\theta})=\sigma^2 M^{-1}(\xi),$$

$$M(\xi)=\sum_{i=1}^{m} n_i f(x_i)f^T(x_i)=\overline{F}_1^T\overline{F}_1;\ \overline{F}_1^T\overline{y}=\sum_{i=1}^{m} n_i f(x_i)\overline{y}_i.$$

Если принять во внимание апостериорные оценки дисперсий ошибок наблюдений, то расчеты следует проводить в соответствии с следующими формулами:

$$\hat{\theta}=\left(F_1^T D_2^{-1}F_1\right)^{-1}F_1^T D_2^{-1}\overline{y};\ \ D(\hat{\theta})=M^{-1}(\xi,y);$$

$$\tilde{d}_\eta(x,\xi,y)=f^T(x)M^{-1}(\xi,y)f(x),$$

$$M(\xi,y)=F_1^T D_2^{-1}F_1=\sum_{i=1}^{m}\sigma_{\overline{y}}^{-2}(x_i)f(x_i)f^T(x_i).$$

Для простоты обозначений и в тех случаях, когда это не является принципиальным и не вызывает путаницу в понимании сути проблемы, наряду с обозначением $\tilde{d}_\eta(x,\xi,y)$ мы будем использовать также обозначение $\tilde{d}_\eta(x,\xi)$. Конечно, обозначение $\tilde{d}_\eta(x,\xi,y)$ более точно характеризует особенность нового подхода, связанного с использованием апостериорной информации об ошибках наблюдений. Следует заметить ещё раз, что здесь не предполагается выполнение условия различия дисперсий ошибок наблюдений в различных точках области X (т. е. условия гетероскедастичности).

С помощью моделирующих вычислений покажем, как сильно могут отличаться значения функций $\hat{d}_\eta(x,\xi^*)$ и $\tilde{d}_\eta(x,\xi^*)$. На рис. 1 представлены априорная функция дисперсии оценки отклика $\hat{d}_\eta(x,\xi^*)$ (пунктирная линия) и одна из возможных реализаций апостериорной функции дисперсии оценки отклика $\tilde{d}_\eta(x,\xi^*)$ (сплошная линия) для $\sigma=2.5$. Случайные ошибки модели $\varepsilon(x)$ генерировались датчиком случайных чисел, распределенных по нормальному закону со средним значением равным нулю и средним квадратическим отклонением $\sigma=2.5$. На рис. 2 показан вид априорной дисперсии $\hat{d}_\eta(x,\xi^*)$ и полученных в результате моделирования на компьютере 50-ти апостериорных дисперсий вида $\tilde{d}_\eta(x,\xi^*)$. Как и следовало ожидать, функция $\tilde{d}_\eta(x,\xi^*)$ является случайной и разброс в ее поведении зависит не только от характеристик случайной величины $\varepsilon(x)$, но и от всех факторов, определяющих аналитический вид этой функции (плана экспериментов, базисного вектора модели и некоторых других). Еще раз заметим, что все множество реализаций случайной функции $\tilde{d}_\eta(x,\xi^*)$ (в количестве 50-ти функций) получено при неизменном (зафиксированном) *D*- (*G*-) оптимальном плане экспериментов, вид которого приведен выше.

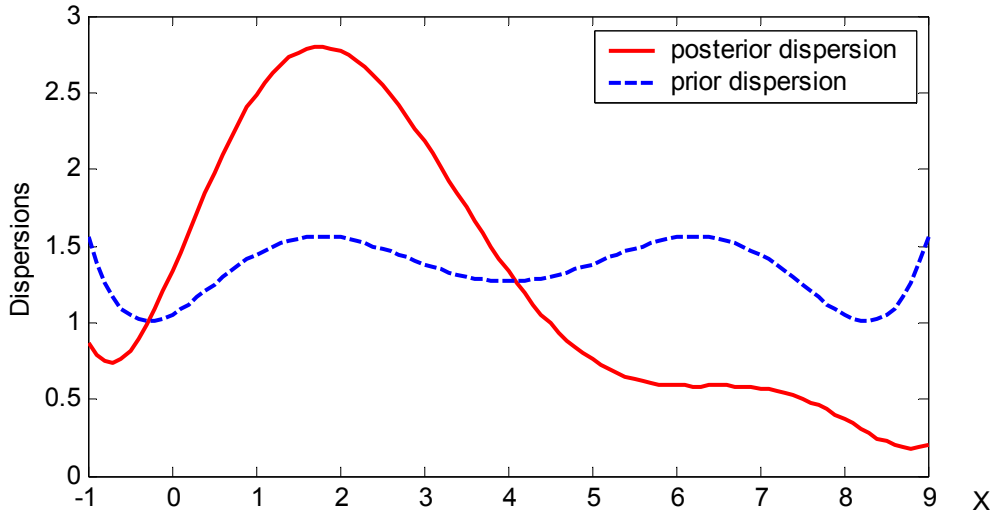


Рис. 1.Вид функций $\hat{d}_\eta(x,\xi^*)$ (пунктирная линия) и $\tilde{d}_\eta(x,\xi^*)$ (сплошная линия).

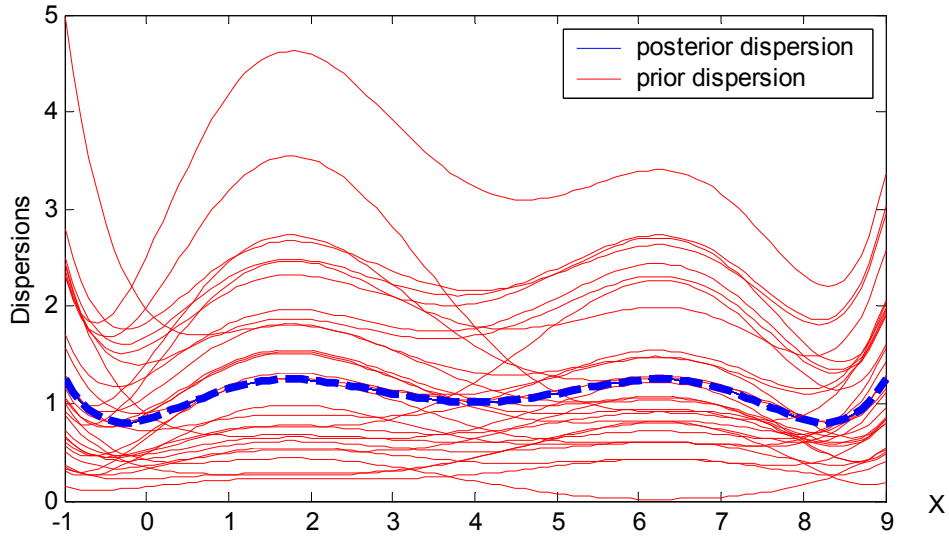


Рис. 2.Функции $\hat{d}_\eta(x,\xi^*)$ (пунктирная линия, жирная) и $\tilde{d}_\eta(x,\xi^*)$ (непрерывные линии) для 50 модельных вычислений.

Вывод: апостериорная дисперсия оценки функции отклика может быть существенно больше априорного значения, соответствующего оптимальному плану.

Продолжим исследование характеристик оптимального плана. Пусть, как и выше, априорная функция дисперсии оценки отклика обозначена через $\hat{d}_\eta(x,\xi^*)$ и апостериорная функция дисперсии оценки отклика — $\tilde{d}_\eta(x,\xi^*)$. Введем обозначение для разности между этими функциями $\Delta d(x,\xi^*) = \hat{d}_\eta(x,\xi^*) - \tilde{d}_\eta(x,\xi^*)$. Пусть количество измерений в каждой точ-

ке спектра $n_i=5$; количество повторных вычислений (модельных просчетов) равно $NM=100$ при каждом значении σ; среднее квадратическое отклонение принимает значения $\sigma \in [0;5]$.

Введем обозначения:

- $\max_{\varepsilon} \max_{x} \Delta d(x,\xi^*)$ — наибольшее на X значение $\Delta d(x,\xi^*)$ за 100 модельных вычислений при определённом значении σ (штрих- пунктирная линия);

- $E_{\varepsilon}(\max_{x} \Delta d(x,\xi^*))$ — среднее значение наибольших на X разностей $\Delta d(x,\xi^*)$ за 100 модельных просчётов при определённом значении σ (сплошная линия);

- $E_{\varepsilon}(\min_{x} \Delta d(x,\xi^*))$ — среднее значение среди наименьших на X значений $\Delta d(x,\xi^*)$ за 100 модельных просчётов при определённом значении σ (пунктирная линия);

- $\min_{\varepsilon} \min_{x} \Delta d(x,\xi^*)$ — наименьшее значение среди наименьших на X разностей $\Delta d(x,\xi^*)$ за 100 модельных просчётов при определённом значении σ (точечная линия).

На рис. 3 приведен вид этих характеристик различия априорной и апостериорных дисперсий оценок функций отклика (для D- (G-) оптимального плана), введенных в рассмотрение выше.

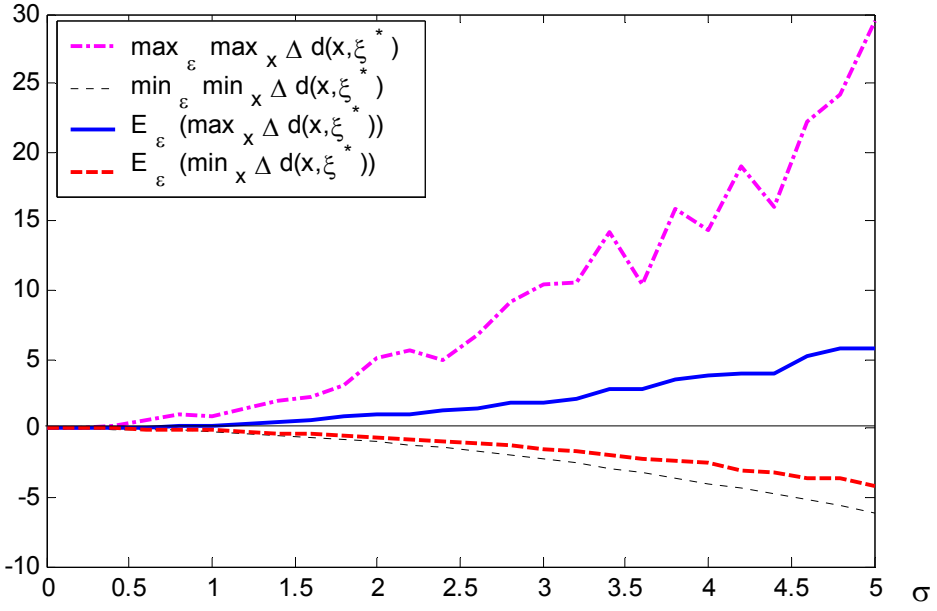


Рис. 3. Характеристики различия дисперсий как функции σ:

$\max_{\varepsilon} \max_{x} \Delta d(x,\xi^*)$ — штрих-пунктирная линия, $E_{\varepsilon}(\max_{x} \Delta d(x,\xi^*))$ — сплошная,

$E_{\varepsilon}(\min_{x} \Delta d(x,\xi^*))$ — пунктирная линия, $\min_{\varepsilon} \min_{x} \Delta d(x,\xi^*)$ — точечная линия.

На рис. 4 представлены только две из этих характеристик: средние значения $E_{\varepsilon}(\max_x \Delta d(x,\xi^*))$ и $E_{\varepsilon}(\min_x \Delta d(x,\xi^*))$.

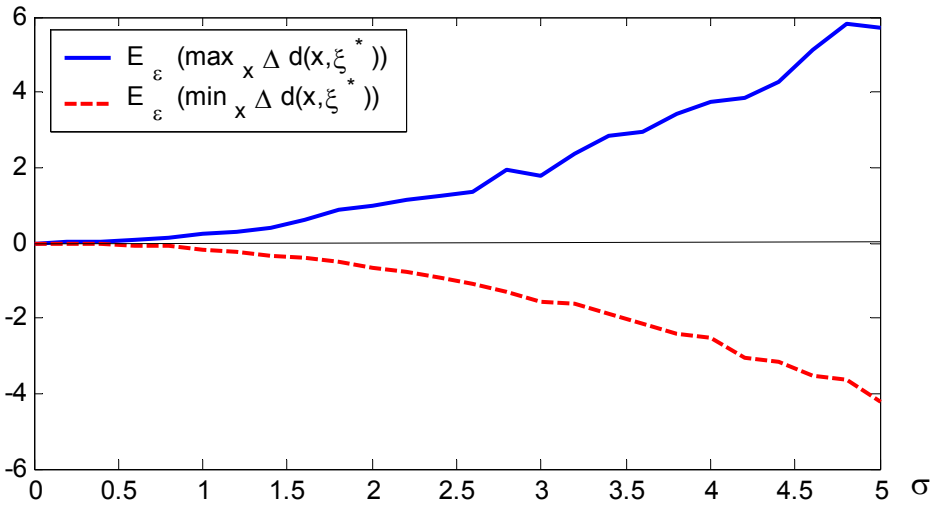


Рис. 4. Средние значения $E_{\varepsilon}(\max_x \Delta d(x,\xi^*))$ — сплошная, $E_{\varepsilon}(\min_x \Delta d(x,\xi^*))$ — пунктирная линии как функции среднеквадратического отклонения σ.

Обозначим относительное отклонение (различие) между этими функциями через $\Delta d_o(x,\xi^*) = (\hat{d}_\eta(x,\xi^*) - \tilde{d}_\eta(x,\xi^*))/\hat{d}_\eta(x,\xi^*)$. Пусть количество измерений в каждой точке спектра n_i*=5;* количество модельных просчетов равно *NM=100* при каждом значении $\sigma, \sigma \in [0;5]$. Аналогично тому, как это было сделано выше, введем в рассмотрение следующие характеристики для относительных разностей $\Delta d_o(x,\xi^*)$:

- $E_{\varepsilon}(\max_x \Delta d_o(x,\xi^*))$ — среднее значение наибольших на X значений $\Delta d_0(x,\xi^*)$ за 100 модельных просчётов при определённом значении σ (сплошная линия);

- $E_{\varepsilon}(\min_x \Delta d_o(x,\xi^*))$ — среднее значение наименьших на X значений $\Delta d_0(x,\xi^*)$ за 100 модельных просчётов при определённом значении σ (пунктирная линия).

На рис. 5 показано поведение средних относительных характеристик различия функций дисперсий в зависимости от значений σ.

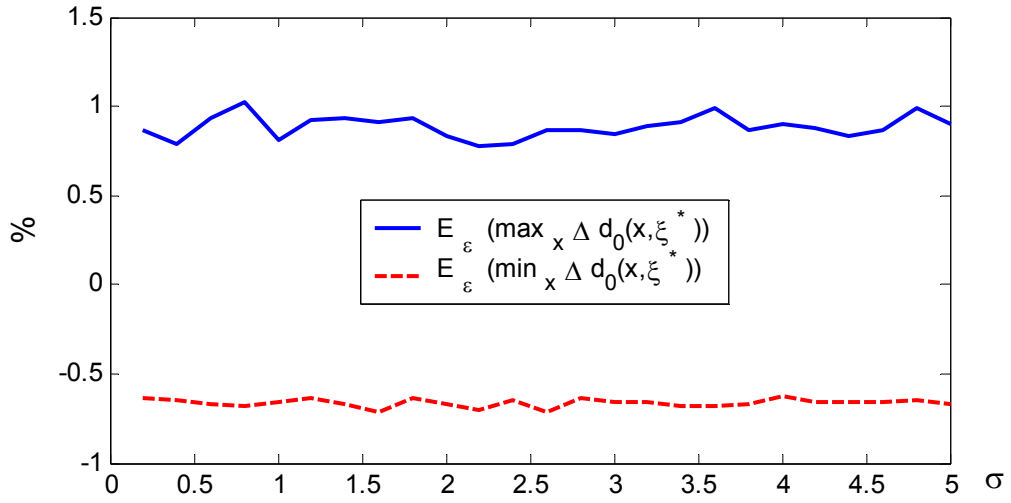


Рис. 5. Значения относительных разностей в зависимости от σ:
$E_\varepsilon(\max_x \Delta d_o(x,\xi^*))$ — сплошная, $E_\varepsilon(\min_x \Delta d_o(x,\xi^*))$ — пунктирная линии.

Исследуем поведение вышерассмотренных характеристик в зависимости от числа повторных наблюдений в точках спектра классического D- (G-) оптимального плана. Исходные данные, касающиеся модели и оптимального плана оставим неизменными, а n_i (количество измерений в каждой точке спектра) пусть принимают целочисленные значения из интервала [2;15]. При этом будем полагать σ=2.5, NM=100. На рис. 6 и рис. 7 приведены графики зависимостей $\max_\varepsilon \max_x \Delta d(x,\xi^*)$, $E_\varepsilon(\max_x \Delta d(x,\xi^*))$, $E_\varepsilon(\min_x \Delta d(x,\xi^*))$ и $\min_\varepsilon \min_x \Delta d(x,\xi^*)$ как функций n_i.

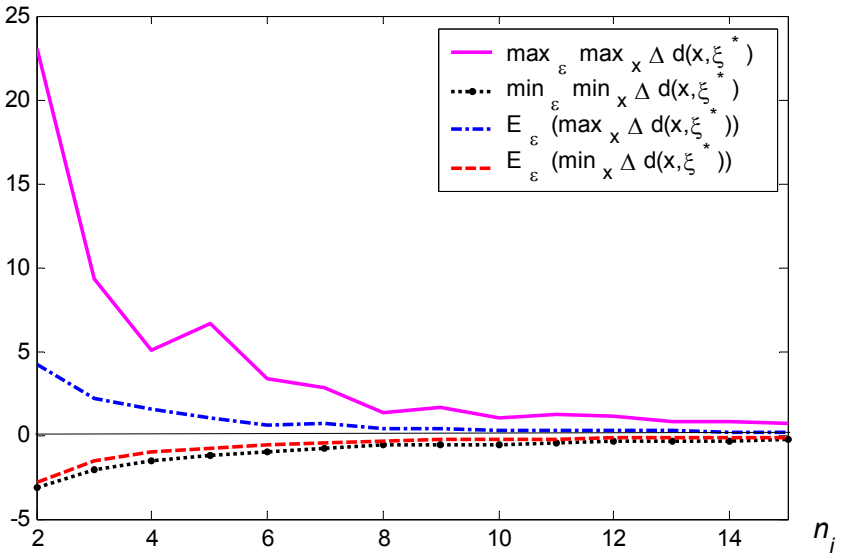


Рис. 6. Значения разностей дисперсий в зависимости от n_i.

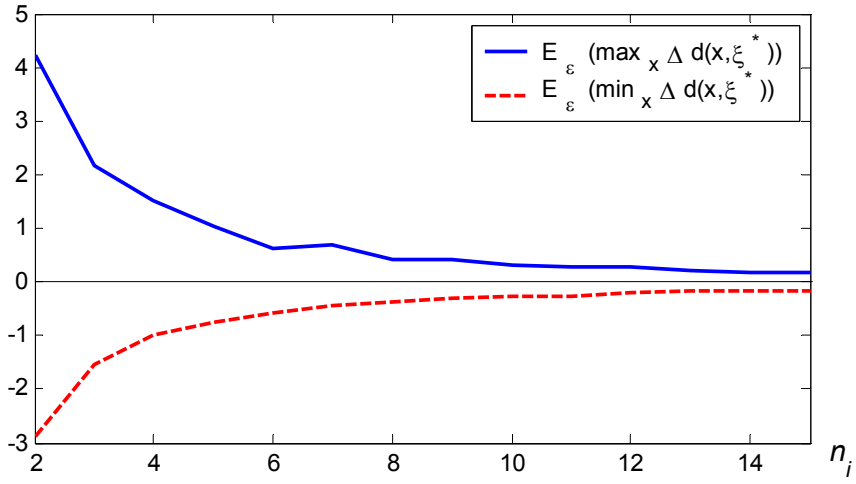


Рис. 7. Значения разностей дисперсий в зависимости от n_i :
$E_\varepsilon(\max_x \Delta d(x,\xi^*))$ — сплошная, $E_\varepsilon(\min_x \Delta d(x,\xi^*))$ — пунктирная линии.

Для вышеприведенной модели и классического D- (G-) оптимального плана исследуем вопрос: как часто априорная функция $\hat{d}_\eta(x,\xi^*)$ превышает значения апостериорной функции $\tilde{d}_\eta(x,\xi^*)$? Для этого введем в рассмотрение величину $P_{\geq 0}^{(1)} = \dfrac{NM_{\geq 0}^{(1)}}{NM} \cdot 100\%$, где NM — общее число модельных вычислений, а $NM_{\geq 0}^{(1)} = \left\| \left\{ \Delta d\left(x,\xi^*\right) \middle| \forall x \in X, \Delta d\left(x,\xi^*\right) \geq 0 \right\} \right\|$ (здесь через $\left\| \{\bullet\} \right\|$ обозначена мощность множества $\{\bullet\}$).

Таким образом, $P_{\geq 0}^{(1)}$ характеризует долю (в процентах) случаев, когда выполняется неравенство $\Delta d\left(x,\xi^*\right) \geq 0, \forall x \in X$. Т. е. величина $P_{\geq 0}^{(1)}$ оценивает долю от общего количества модельных просчётов, в которых значения функции апостериорной дисперсии оказываются меньше (на всей области X) , чем соответствующие значения априорной дисперсии. На рис. 8 показана зависимость $P_{\geq 0}^{(1)}$ от значений n_i.

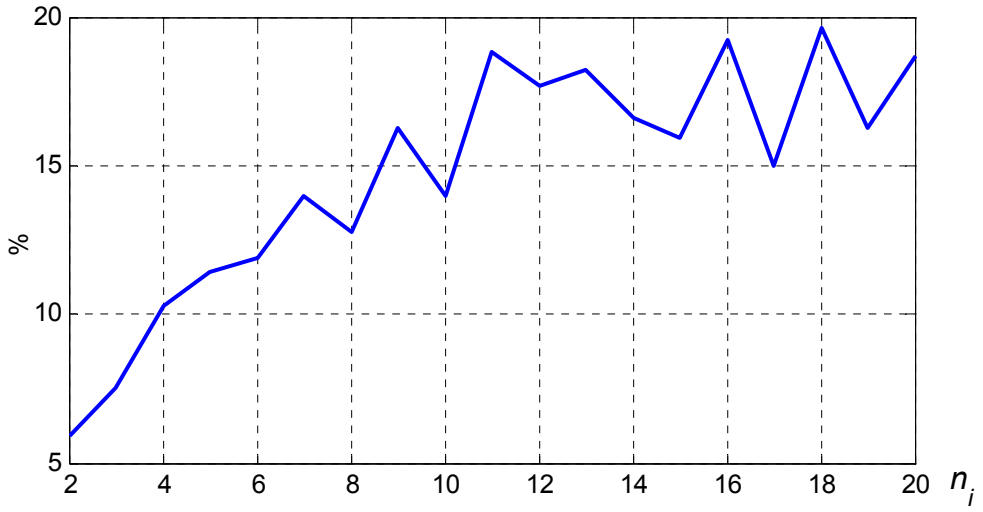


Рис 8. Изменение доли $P_{\geq 0}^{(1)} = \dfrac{NM_{\geq 0}^{(1)}}{NM} \cdot 100\%$ в зависимости от n_i.

Построим графики долей $P_{\geq 0}^{(1)} = \dfrac{NM_{\geq 0}^{(1)}}{NM} \cdot 100\%$ и $P_{\leq 0}^{(1)} = \dfrac{NM_{\leq 0}^{(1)}}{NM} \cdot 100\%$ ($NM_{\leq 0}^{(1)} = \left\| \left\{ \Delta d\left(x, \xi^*\right) \middle| \forall x \in X, \Delta d\left(x, \xi^*\right) \leq 0 \right\} \right\|$) в зависимости от значений n_i (см. рис. 9).

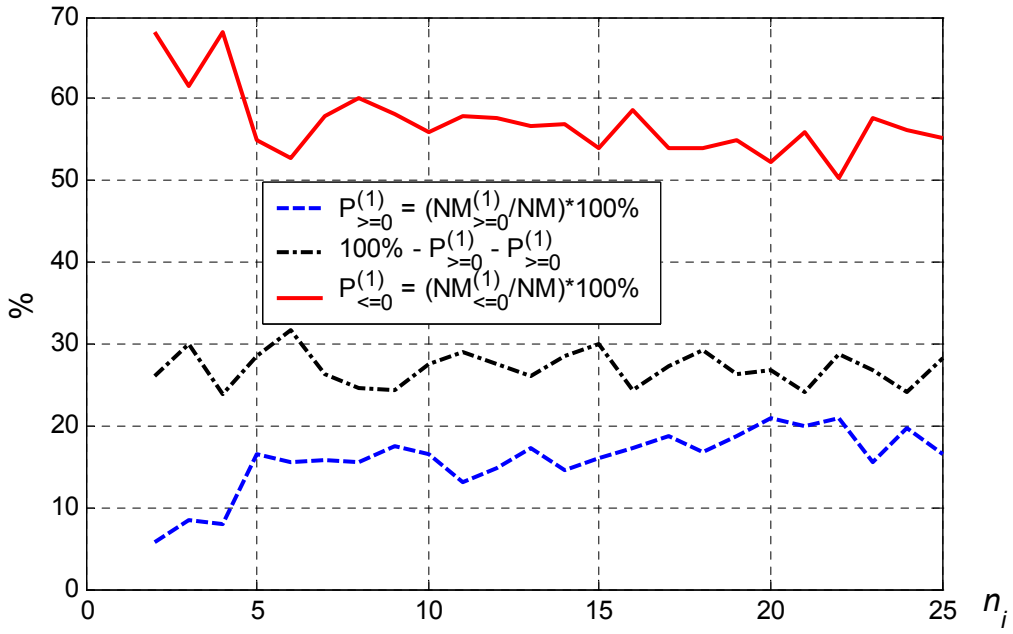


Рис. 9. Изменение $P_{\geq 0}^{(1)} = \dfrac{NM_{\geq 0}^{(1)}}{NM} \cdot 100\%$ (пунктирная линия), $P_{\leq 0}^{(1)} = \dfrac{NM_{\leq 0}^{(1)}}{NM} \cdot 100\%$ (непрерывная линия) и $(100\% - P_{\geq 0}^{(1)} - P_{\leq 0}^{(1)})$ (остальные случаи, штрих-пунктирная линия) в зависимости от n_i.

Итак, выше на модельных примерах было показано, что для классических *D*- (*G*-) оптимальных планов отличие между априорной и апостериорной дисперсиями оценок моделей может быть существенным. Интересно найти ответ на вопрос: не поведет ли себя (в смысле значений этих дисперсий) неоптимальный план «лучше», чем оптимальный? Покажем, что для неоптимального плана апостериорная дисперсия может быть для всех значений x (из области X) меньше априорной (выше было показано, что для оптимального плана апостериорная дисперсия может быть существенно выше априорной). Для модели, рассмотренной выше ($f^T(x)=(1,x,x^2,x^3)$, $\theta^T=(2.0,5.0,-3.0,4.0)$, $X=[-1.0;9.0]$), и для плана, который не является *D*- оптимальным и имеет вид:

$$\xi^0=\begin{Bmatrix}-0.5 & 2.0 & 5.0 & 8.5\\ n_1 & n_2 & n_3 & n_4\end{Bmatrix}, n_1=n_2=n_3=n_4,\ n_i=4, i=1,2,3,4,$$

найдем функции $\hat{d}_\eta(x,\xi^*)$, $\tilde{d}_\eta(x,\xi^0)$ и построим их графики (см. рис. 10 и рис. 11).

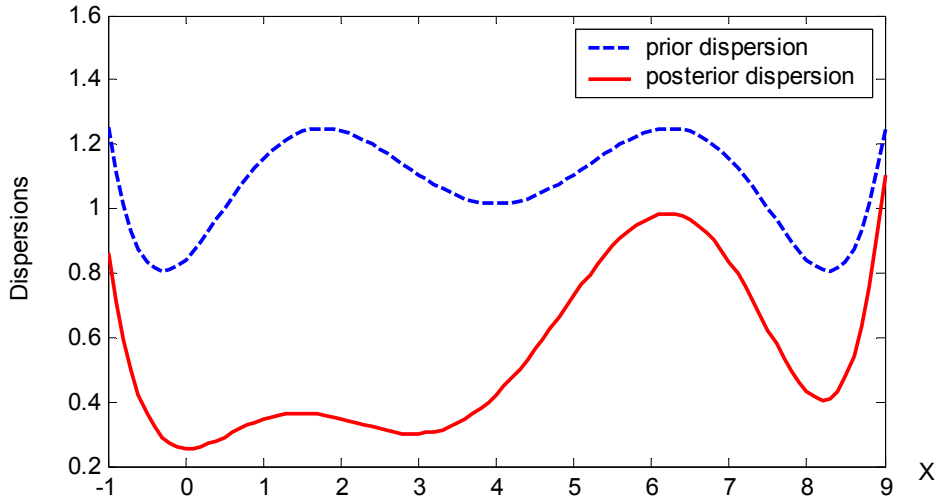


Рис. 10. Функции дисперсии $\hat{d}_\eta(x,\xi^*)$ (пунктирная линия) и $\tilde{d}_\eta(x,\xi^0)$ (сплошная линия) для неоптимального плана.

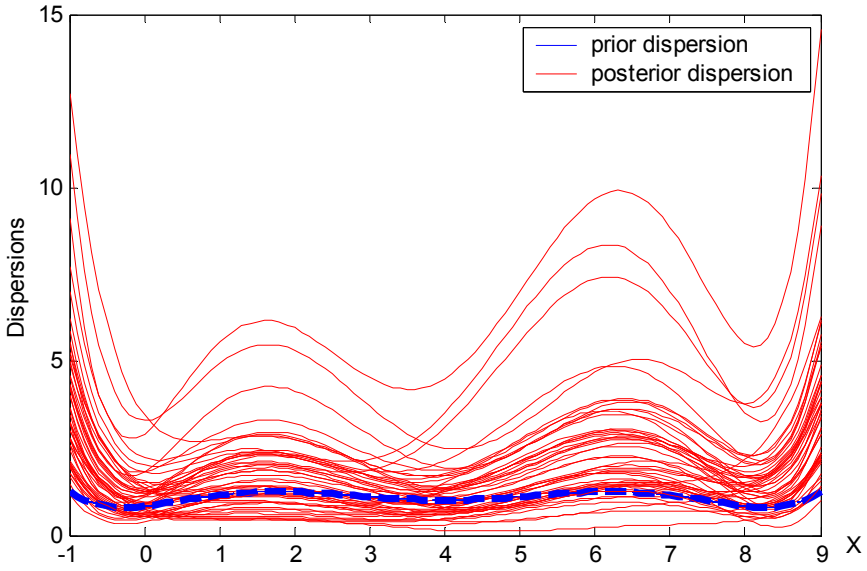


Рис. 11. Функции дисперсии $\hat{d}_\eta(x,\xi^*)$ (пунктирная линия, жирная) и $\tilde{d}_\eta(x,\xi^0)$ — для не *D*-оптимального плана (сплошная линия) (30 модельных просчётов).

Рассмотрим пример регрессионной модели от двух переменных (для ситуации $\boldsymbol{M_{II}}$). Пусть базисный вектор модели и вектор параметров имеют вид: $f^T(x)=(1,x_1,x_2,x_1^2)$, $\theta^T=(4.0,5.0,3.0,9.0)$. Подберем соответствующий данной модели классический *D*- (*G*-) оптимальный план (см. [4]) и пересчитаем его на область $X1\times X2$, *X1=[1.0;3.0], X2=[0.0;6.0]* . Получим план $\xi^*=\begin{Bmatrix}(1,6)&(2,6)&(3,6)&(1,0)&(2,0)&(3,0)\\ n_1&n_2&n_3&n_4&n_5&n_6\end{Bmatrix}$. Количество измерений в каждой точке спектра выберем равным $n_i=5, i=1,2,3,4,5,6;$ (*N=30*) при этом, количество параметров модели равно *k=4* и количество точек спектра плана равно *m=6*. На Рис. 12 показан вид априорной функции дисперсии $\hat{d}_\eta(x,\xi^*)$ и реальной (фактической, апостериорной) функции дисперсии $\tilde{d}_\eta(x,\xi^*)$. Случайная ошибка наблюдений при этом генерировалась датчиком случайных чисел, распределенных по нормальному закону с нулевым средним и $\sigma=3.5$.

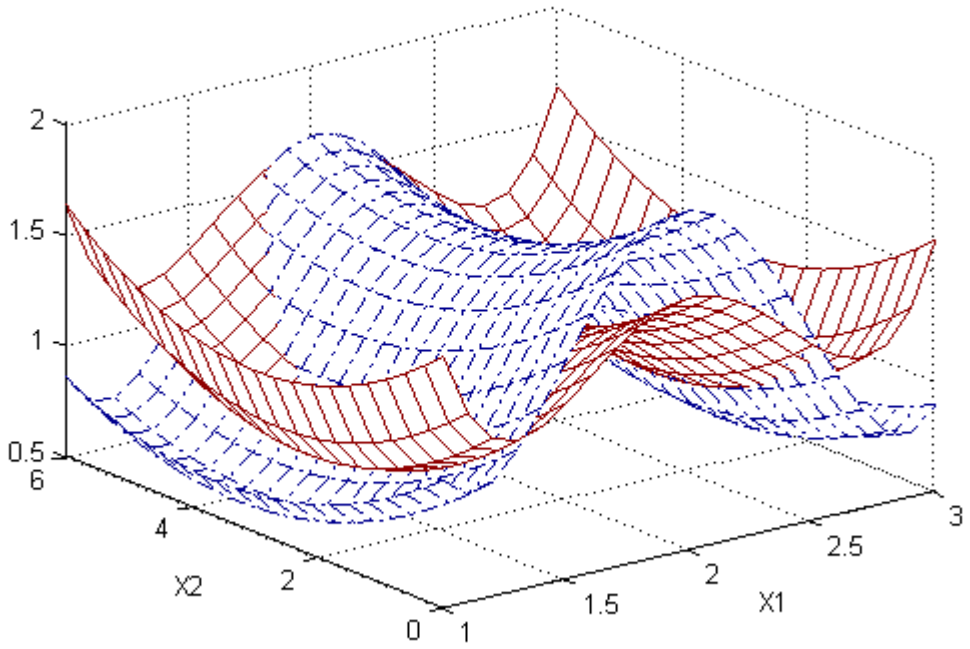


Рис.12. Сравнение априорной функции дисперсии $\hat{d}_\eta(x,\xi^*)$ (сплошные линии сетки) и реальной (апостериорной) функции дисперсии $\tilde{d}_\eta(x,\xi^*)$ (пунктирные линии сетки).

2. Сравнительный анализ априорных и апостериорных оценок параметров модели и их характеристик для классических оптимальных планов экспериментов

2.1. Анализ критериев эффективности

Исследуем поведение *D*- критерия эффективности. С этой целью для *NM*=1000 (количество модельных просчётов) вычислим значения $\det(M^{-1}(\xi^*,y))$ и построим гистограмму закона распределения. Вдоль вертикальной оси будем откладывать количество ситуаций, соответствующих тем значениям $\det(M^{-1}(\xi^*,y))$, которые попали в данный интервал (откладываются вдоль горизонтальной оси). При этом были вычислены также среднее значение критерия эффективности $E_\varepsilon\left(\det(M^{-1}(\xi,y))\right)$ и значение классического функционала $\det(\sigma^2 M^{-1}(\xi^*))$ (постоянная величина).

D- критерий.

Пример.

Базисный вектор модели имеет вид $f^T(x)=(1,x,x^2,x^3)$, область экспериментирования — *X*= [-1;1], вектор параметров модели — $\theta^T=(2,5,-3,2)$, ошибка наблюдений распределена по нормальному закону

$\varepsilon \sim N(0;\sigma^2)$. Классический *D*- оптимальный план имеет вид $\xi^* = \begin{Bmatrix} -1.0 & -0.447 & 0.447 & 1 \\ 4 & 4 & 4 & 4 \end{Bmatrix}$, *NM*=1000, $\sigma = 2$. На рис. 13 показан закон распределения критерия $\det(M^{-1}(\xi^*, y))$

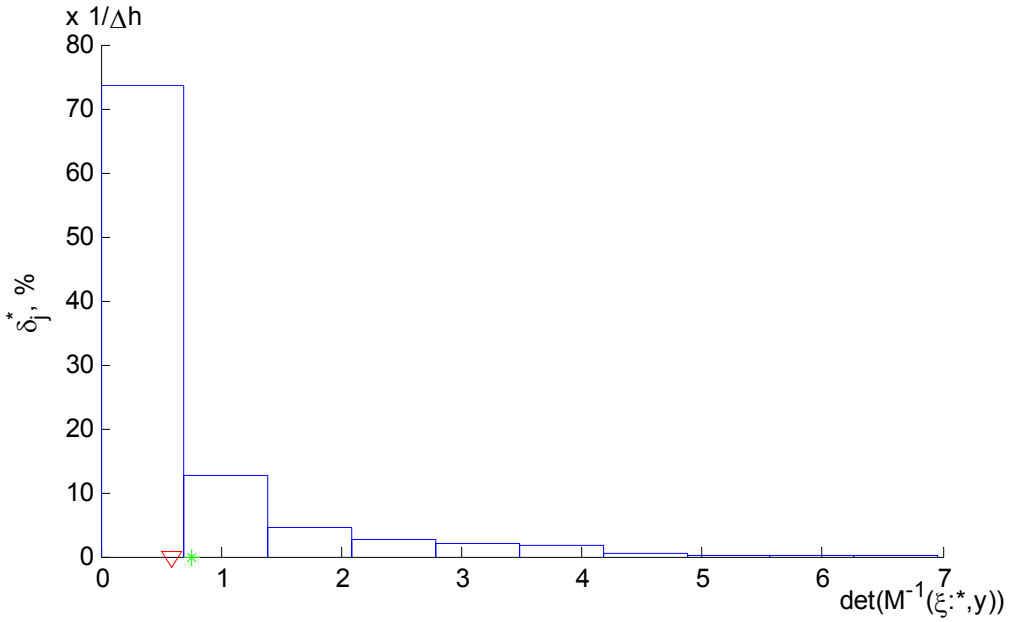


Рис. 13. Закон распределения критерия $\det(M^{-1}(\xi^*, y))$.

При этом $\Delta h = 0{,}696$ (ширина интервалов), $I_{NM} = 10$ (количество интервалов разбиения значений $\det(M^{-1}(\xi^*, y))$), $\delta_j^* = P_j^* \cdot 100\% = \frac{N_j}{NM} 100\%$, $j = 1,2,...,I_{NM}$, δ_j^* — эмпирические доли, N_j — количество реализаций выборки, попавших в j-ый интервал, $j = 1,2,...,I_{NM}$, $\sum_{j=1}^{I_{NM}} N_j = NM$, P_j^* — эмпирическая частость (доля). На рис. 13 также показаны значения: $\det(\sigma^2 M^{-1}(\xi^*))$=0.7629 (звёздочка), $E_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$=0.5879 (треугольник). При этом наименьшее и наибольшее значения критерия эффективности составили: $\min_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$=2.4028e-004 и $\max_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$=6.9609 соответственно. На рис. 14 показан этот же закон распределения, когда число интервалов равно $I_{NM} = 100$.

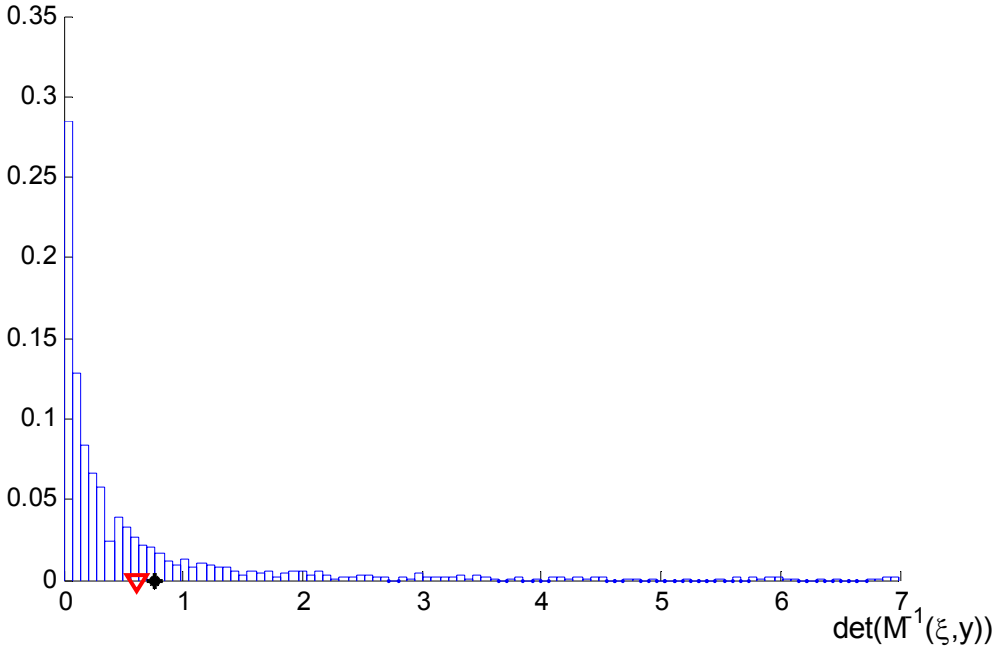


Рис. 14. Закон распределения $\det(M^{-1}(\xi^*, y))$ при $I_{NM} = 100$, $\Delta h = 0{,}0696$.

Здесь, как и выше показаны значения $\det(\sigma^2 M^{-1}(\xi^*))$=0.7629 (звёздочка) и $E_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$=0.6098 (треугольник). При этом $\min_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$=1.3674e-004 и $\max_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$=6.9976.

Анализ поведения случайной величины $\det(M^{-1}(\xi^*, y))$ показал, что в 21.6% случаев всех модельных просчётов значение $\det(M^{-1}(\xi^*, y))$ было больше значения функционала $\det(\sigma^2 M^{-1}(\xi^*))$ (используемого в классической теории планирования экспериментов).

A- критерий.

Пример.

Пусть базисный вектор модели имеет вид $f^T(x) = (1, x, x^2)$, X= [-1;1], $\theta^T = (2, 5, -3)$, как и выше, $\varepsilon \sim N(0; \sigma^2)$. Классический A-оптимальный план в этом случае имеет вид $\xi^* = \begin{Bmatrix} -1 & 0 & 1 \\ 4 & 8 & 4 \end{Bmatrix}$. Положим NM=1000 и $\sigma = 2$. На рис. 15 показана гистограмма выборочного закона распределения критерия эффективности $SpM^{-1}(\xi^*, y)$.

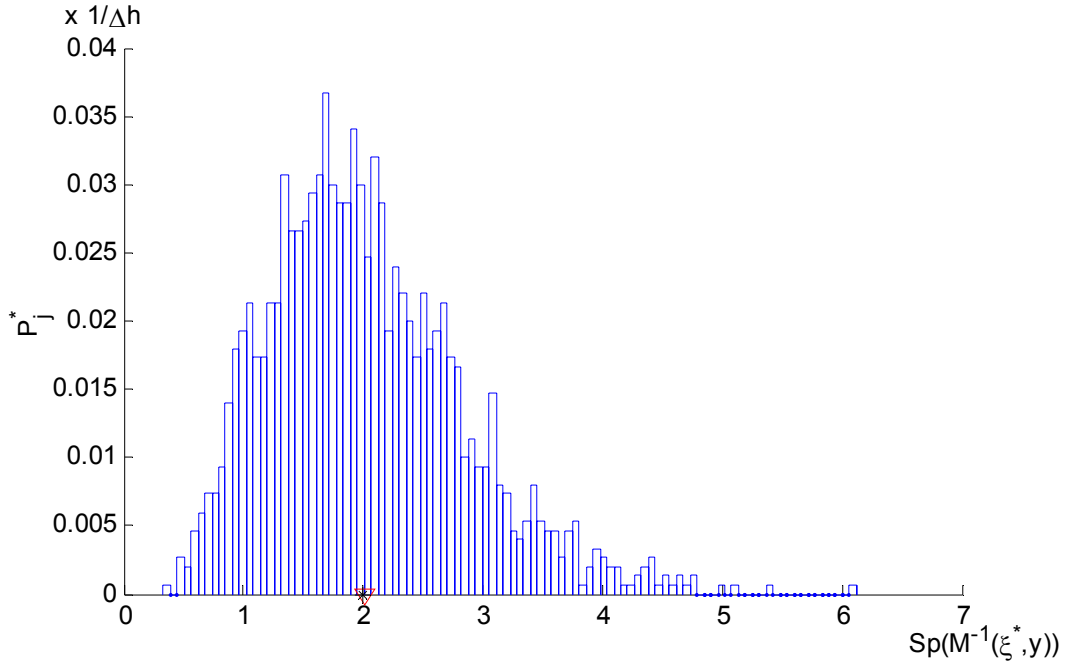


Рис. 15. Гистограмма выборочного закона распределения критерия эффективности $SpM^{-1}(\xi^*, y)$.

На этом рисунке также показаны значения $Sp(\sigma^2 M^{-1}(\xi^*))=2$ (звёздочка), $E_\varepsilon\left(Sp(M^{-1}(\xi^*, y))\right)=2.0102$ (треугольник). При этом $\min_\varepsilon\left(Sp(M^{-1}(\xi^*, y))\right)=0.2448$ и $\max_\varepsilon\left(Sp(M^{-1}(\xi^*, y))\right)=6.1229$.

Следует отметить, что в 45.7% модельных просчётов значение критерия эффективности $Sp(M^{-1}(\xi^*, y))$ было больше, чем $Sp(\sigma^2 M^{-1}(\xi^*))$ (соответствующего классическому подходу), т.е. $P\left\{Sp(M^{-1}(\xi^*, y)) > Sp\left(\sigma^2 M^{-1}(\xi^*)\right)\right\} \approx 0.457$.

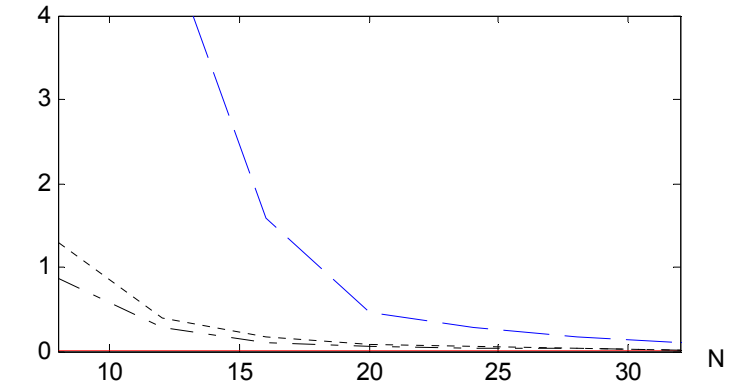
Отметим также, что аналогичная вероятность, оцененная для критерия *D*- эффективности при том же общем числе экспериментов (равным 16) и совпадающих характеристиках закона распределения ошибки наблюдений составила 0.216. Это говорит о том, что если вместо апостериорных характеристик плана эксперимента использовать априорные, то в случае *D*- критерия вероятность получить значение большее, чем гарантируется классической теорией планирования экспериментов меньше, чем при использовании *A*- критерия. В этом смысле *D*- оптимальные планы экспериментов являются более устойчивыми, чем *A*- оптимальные.

Исследуем поведение критериев оптимальности в зависимости от значений *N* (общего числа экспериментов). Этим самым будут исследованы асимптотические свойства критерия эффективности в случаях использования априорного и апостериорного подходов.

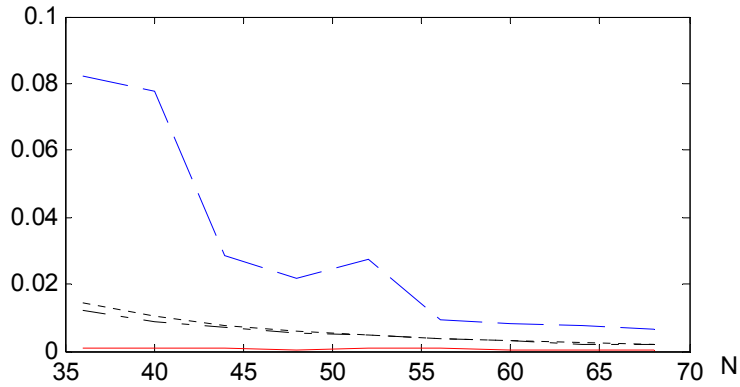
D- критерий.

Пример.

Пусть, как и в примере выше, базисный вектор модели имеет вид $f^T(x)=(1,x,x^2,x^3)$ и X=[-1;1], $\theta^T=(2,5,-3,2)$, $\varepsilon\sim N(0;\sigma^2)$. Для классического D- оптимального плана $\xi^*=\begin{Bmatrix}-1.0 & -0.447 & 0.447 & 1\\ n_1 & n_2 & n_3 & n_4\end{Bmatrix}$ при $n_1=n_2=n_3=n_4$ и для $N=\sum_{i=1}^{4}n_i$ проведем исследование зависимости значения критерия эффективности $\det(M^{-1}(\xi^*,y))$ от N при изменении N в диапазоне от 8 до 68 с шагом равным 4 (т.е. при N=8(4)68). Количество модельных просчетов положим равным NM=100 при каждом значении N. При этом ошибку измерений будем генерировать датчиком случайных чисел, распределенных по нормальному закону $\varepsilon\sim N(0;\sigma^2)$, $\sigma=2$. На Рис. 16 (a) и b)) показаны значения $\max\limits_{\varepsilon}\det(M^{-1}(\xi^*,y))$, $\min\limits_{\varepsilon}\det(M^{-1}(\xi^*,y))$ и $E_\varepsilon\left(\det(M^{-1}(\xi^*,y))\right)$.



a)



b)

Рис. 16. Зависимости $\max\limits_{\varepsilon}\det(M^{-1}(\xi^*,y))$ (пунктирная линия), $\min\limits_{\varepsilon}\det(M^{-1}(\xi^*,y))$ (непрерывная линия), $\det(\sigma^2M^{-1}(\xi^*))$ (точечная линия) и $E_\varepsilon\left(\det(M^{-1}(\xi^*,y))\right)$ (штрих-пунктирная линия) для различных диапазонов N.

Выше было исследовано поведение абсолютного значения критерия эффективности $\det(M^{-1}(\xi^*, y))$. Для исследования относительных характеристик возможных отличий в значениях критерия эффективности для априорного и апостериорного подходов введем следующие обозначения:

$$\Delta_N = \det(\sigma^2 M^{-1}(\xi^*)) - E_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right),$$

$$\Delta_{N,\max} = \det(\sigma^2 M^{-1}(\xi^*)) - \max_\varepsilon \det(M^{-1}(\xi^*, y))$$

$$\Delta_{N,\min} = \det(\sigma^2 M^{-1}(\xi^*)) - \min_\varepsilon \det(M^{-1}(\xi^*, y)).$$

Результаты модельных просчетов этих разностей для *NM*=100 (количество модельных просчётов) и при разных значениях N приведены на рис. 17.

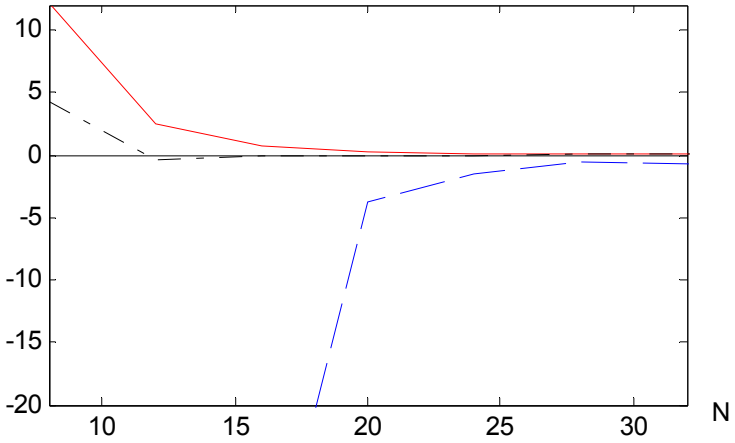


Рис. 17. Зависимости Δ_N (штрих- пунктирная линия), $\Delta_{N,\max}$ (пунктирная линия), $\Delta_{N,\min}$ (непрерывная линия) от значения *N*.

Аналогично исследуем относительные величины отклонений $\Delta_N / \det(\sigma^2 M^{-1}(\xi^*))$, $\Delta_{N,\max} / \det(\sigma^2 M^{-1}(\xi^*))$ и $\Delta_{N,\min} / \det(\sigma^2 M^{-1}(\xi^*))$. На рис. 18 показаны соответствующие зависимости для относительных отклонений.

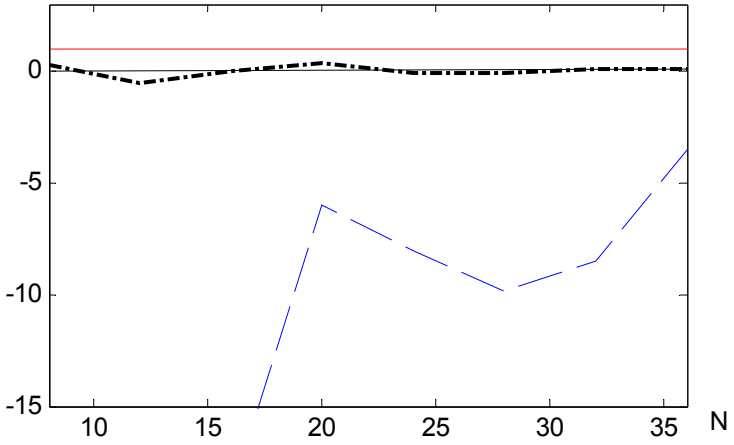


Рис. 18. Относительные величины $\Delta_N / \det(\sigma^2 M^{-1}(\xi^*))$ — штрих-пунктирная линия, $\Delta_{N,\max} / \det(\sigma^2 M^{-1}(\xi^*))$ — пунктирная линия и $\Delta_{N,\min} / \det(\sigma^2 M^{-1}(\xi^*))$ — непрерывная линия.

Так, из рис. 18 видно, что, например, при N=20 величина $\Delta_{N,\max} = \det(\sigma^2 M^{-1}(\xi^*)) - \max\limits_{\varepsilon} \det(M^{-1}(\xi^*, y))$ превосходит $\det(\sigma^2 M^{-1}(\xi^*))$ в 6 раз (или $\max\limits_{\varepsilon} \det(M^{-1}(\xi^*, y))$ больше $\det(\sigma^2 M^{-1}(\xi^*))$ в 7 раз).

Исследуем вопрос: как часто апостериорное значение критерия эффективности превосходит значение этого критерия, найденного по априорным данным? С этой целью введем обозначение $P_{\le 0}^{(1)} = \dfrac{NM_{\le 0}^{(1)}}{NM} \cdot 100\%$, где $NM_{\le 0}^{(1)} = \left\| \left\{ \Delta \middle| \Delta = \det(\sigma^2 M^{-1}(\xi^*)) - \det(M^{-1}(\xi^*, y)) \le 0 \right\} \right\|$. Т.е. $P_{\le 0}^{(1)}$ - это доля (в процентах) модельных просчётов, для которых значение $\det(M^{-1}(\xi^*, y))$ было не меньше значения $\det(\sigma^2 M^{-1}(\xi^*))$. На Рис. 19 показано изменение доли $P_{\le 0}^{(1)} = \dfrac{NM_{\le 0}^{(1)}}{NM} \cdot 100\%$ в зависимости от N. Общая тенденция изменения этой доли — это ее рост с замедлением роста при больших значениях N. При этом NM было выбрано равным 100.

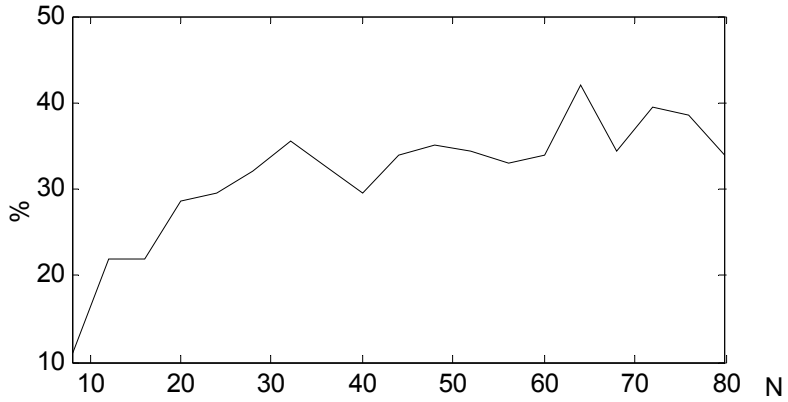


Рис. 19 . Изменение доли $P_{\le 0}^{(1)} = \dfrac{NM_{\le 0}^{(1)}}{NM} \cdot 100\%$ в зависимости от N.

Проведем исследование поведения критерия эффективности в зависимости от значения среднеквадратического отклонения ошибки измерения σ. Для тех же исходных данных и модели регрессии построим зависимость $\det(M^{-1}(\xi^*, y))$ от σ при $\sigma \in [0.5, 2.0]$. На рис. 20 показаны графики зависимостей $\det(\sigma^2 M^{-1}(\xi^*))$, $\max\limits_{\varepsilon} \det(M^{-1}(\xi^*, y))$, $\min\limits_{\varepsilon} \det(M^{-1}(\xi^*, y))$ и $E_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$ от σ. При этом NM=100.

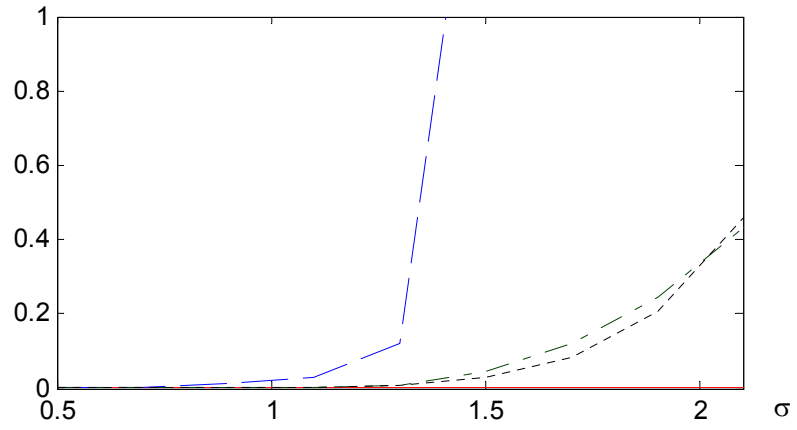


Рис. 20 . Зависимости $\max\limits_{\varepsilon} \det(M^{-1}(\xi^*, y))$ - пунктирная линия, $\min\limits_{\varepsilon} \det(M^{-1}(\xi^*, y))$ — сплошная линия, $E_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$ — штрих- пунктирная линия, $\det(\sigma^2 M^{-1}(\xi^*))$ — точечная линия.

Аналогично тому, как это было сделано выше, после введения обозначений $\Delta_\sigma = \det(\sigma^2 M^{-1}(\xi^*)) - E_\varepsilon\left(\det(M^{-1}(\xi^*, y))\right)$,
$\Delta_{\sigma,\max} = \det(\sigma^2 M^{-1}(\xi^*)) - \max\limits_{\varepsilon} \det(M^{-1}(\xi^*, y))$ и

$\Delta_{\sigma,\min} = \det(\sigma^2 M^{-1}(\xi^*)) - \min_{\varepsilon} \det(M^{-1}(\xi^*, y))$, были проведены расчеты и по ним построены графики для разностей Δ_σ, $\Delta_{\sigma,\max}$, и $\Delta_{\sigma,\min}$ (см. Рис. 21). А затем и для относительных величин: $\Delta_\sigma / \det(\sigma^2 M^{-1}(\xi^*))$ (штрих- пунктирная линия), $\Delta_{\sigma,\max} / \det(\sigma^2 M^{-1}(\xi^*))$ (пунктирная линия) и $\Delta_{\sigma,\min} / \det(\sigma^2 M^{-1}(\xi^*))$ (сплошная линия) (см. Рис. 22).

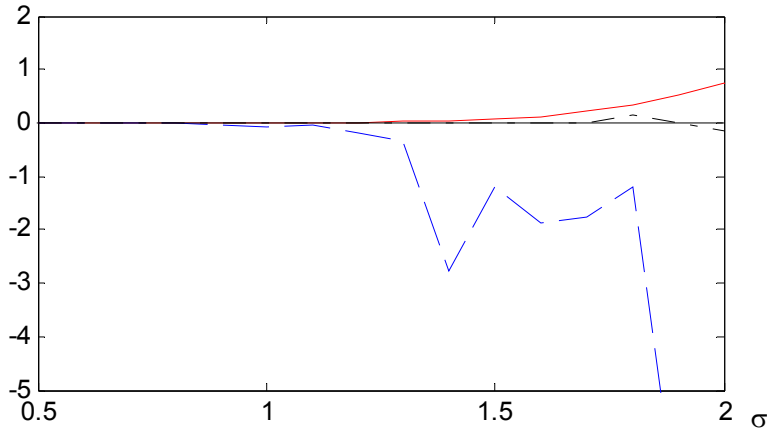


Рис. 21. Разности Δ_σ — штрих- пунктирная линия, $\Delta_{\sigma,\max}$ — пунктирная линия, $\Delta_{\sigma,\min}$ — сплошная линия.

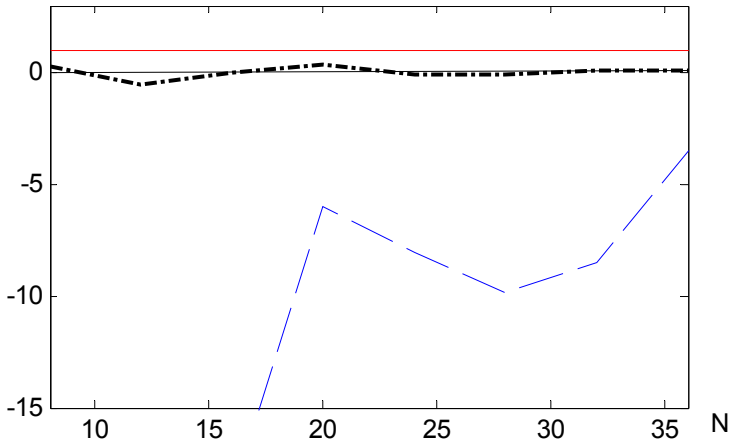


Рис. 22. Относительные величины $\Delta_\sigma / \det(\sigma^2 M^{-1}(\xi^*))$ — штрих- пунктирная линия, $\Delta_{\sigma,\max} / \det(\sigma^2 M^{-1}(\xi^*))$ — пунктирная линия, $\Delta_{\sigma,\min} / \det(\sigma^2 M^{-1}(\xi^*))$ — сплошная линия.

Заметим, что, в отличии от аналогичных зависимостей относительно общего числа экспериментов *N*, в данном случае не наблюдается какой-либо четко выраженной тенденции в изменении относительных величин. Можно предположить, что они не зависят от значения σ.

И, наконец, исследуем поведение доли $P_{\le 0}^{(1)} = \frac{NM_{\le 0}^{(1)}}{NM} \cdot 100\%$, где $NM_{\le 0}^{(1)} = \left\| \left\{ \Delta \middle| \Delta = \det(\sigma^2 M^{-1}(\xi^*)) - \det(M^{-1}(\xi^*, y)) \le 0 \right\} \right\|$, в зависимости от значений σ. На Рис. 23 показан вид зависимости этой доли $P_{\le 0}^{(1)}$ как функции среднеквадратического отклонения.

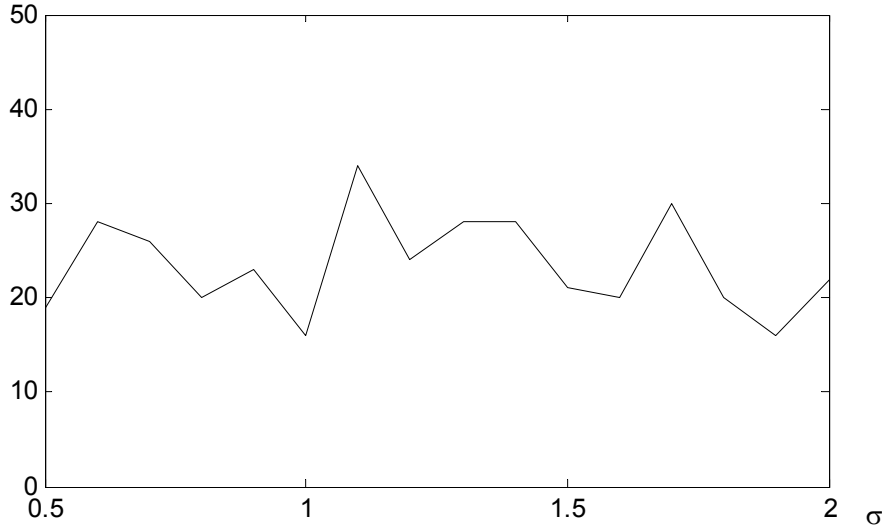


Рис. 23. Изменение доли $P_{\le 0}^{(1)} = \frac{NM_{\le 0}^{(1)}}{NM} \cdot 100\%$ в зависимости от σ.

Можно сделать вывод, что изменение доли (как и относительные характеристики, исследованные выше) не имеет ярко выраженного (отличного от постоянного значения) тренда.

Сравним поведение критериев эффективности для случая, когда план экспериментов является *D*- оптимальным и не является таковым. Это сравнение представляется важным, поскольку позволит ответить на вопрос: можно ли воспользоваться неоптимальным планом вместо оптимального, если учитывается апостериорная информация.

Пусть базисный вектор модели — $f^T(x) = (1, x, x^2, x^3)$ и X=[-1;1], $\theta^T = (2,5,-3,2)$, $\varepsilon \sim N(0;\sigma^2)$, $\sigma = 2$. Для этой модели классический *D*- оптимальный план эксперимента имеет вид: $\xi^* = \begin{Bmatrix} -1.0 & -0.447 & 0.447 & 1 \\ 4 & 4 & 4 & 4 \end{Bmatrix}$. При этом $\det(\sigma^2 M^{-1}(\xi^*))$=0.7629. В качестве альтернативы рассмотрим план эксперимента с «равномерным» спектром $\xi^R = \begin{Bmatrix} -1.0 & -0.3333 & 0.3333 & 1 \\ 4 & 4 & 4 & 4 \end{Bmatrix}$. Было выполнено *NM*=1000 модельных

просчётов. В 24.5% из них значение $\det(M^{-1}(\xi^R, y))$ было меньше, чем $\det(\sigma^2 M^{-1}(\xi^*))$ (равного 0.7629).

Таким образом, можно оценить вероятность $P\left\{\det(M^{-1}(\xi^R, y)) < \det(\sigma^2 M^{-1}(\xi^*))\right\} = 0.245$, которая говорит о том, что только в 24.5% процентах случаев план с «равномерным» спектром обработанный в соответствии с апостериорной информацией будет иметь значение критерия эффективности меньшее, чем оптимальный план, построенный для этой же модели.

2.2. Анализ схем оценивания параметров

Сравним три схемы оценивания параметров (см. табл. 1). Для этого выберем в качестве сравнительных характеристик следующие:

1) $\Delta_{\theta,\hat{\theta}_{(s)}} = \left\|\theta - \hat{\theta}_{(s)}\right\|$, s — номер расчётной схемы ($s=1,2,3$, соответствует пп. 1,2 и 3 Табл. 1 соответственно); $\left\|\theta - \hat{\theta}_{(s)}\right\| = \sqrt{\sum_{i=1}^{k}\left(\theta_i - \hat{\theta}_{i(s)}\right)^2}$; $\theta_i, i = 1,2,...,k$ — истинные значения параметров модели; $\hat{\theta}_{i(s)}, i = 1,2,...,k$ - оценки параметров модели, полученные в соответствии с расчётными формулами s-ой схемы, $s=1,2,3$.

2) аналогично, для сравнения значений критериев эффективности вводим в рассмотрение следующие разности:

$\Delta_{\Phi,\Phi_{(s)}} = \Phi\left(\sigma^2 M^{-1}\left(\xi^*\right)\right) - \Phi\left(M_{(s)}^{-1}\left(\xi^*, y\right)\right)$, $s=2,3$ (номер расчетной схемы), где

$\sigma^2 M^{-1}\left(\xi^*\right) = \sigma^2\left(F^T F\right)^{-1} = \sigma^2 M_{(1)}^{-1}\left(\xi\right)$— дисперсионная матрица оценок параметров $\hat{\theta}_{(1)}$, полученных в соответствии с первой расчётной схемой (для $s=1$).

Случай *m=k*, *D*-оптимальный план.

Предположим, что дисперсия (σ^2) известна точно, тогда для модели с параметрами $\theta^T = (2{,}5, -3, 4)$, базисным вектором $f^T(x) = (1, x, x^2, x^3)$, $x \in X = [-1,1]$ воспользуемся D-оптимальным планом экспериментов

$$\xi^* = \begin{Bmatrix} -1.0 & -0.447 & 0.447 & 1 \\ 4 & 4 & 4 & 4 \end{Bmatrix}.$$

Поскольку величины $\Delta_{\theta,\hat{\theta}_{(s)}}$ и $\Delta_{\Phi,\Phi_{(s)}}$ являются случайными, то построим для них (для s = 2) законы распределения (см. рис. 24 а, б). При этом выберем $I_{NM} = 40$ (количество интервалов гистограммы), $\Delta h =$ 0.469

(ширина интервала гистограммы). На этих рисунках треугольником показано значение $E_\varepsilon \Delta_{\Phi,\Phi_{(s)}}$ — выборочное среднее значение разностей $\Delta_{\Phi,\Phi_{(s)}}$.

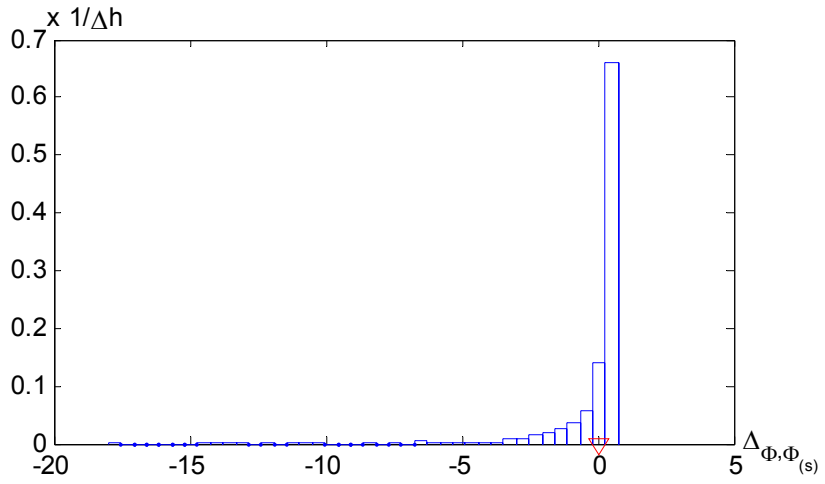


Рис. 24, а. Эмпирический закон распределения случайной величины $\Delta_{\Phi,\Phi_{(s)}}$.

На рис. 24, б, количество интервалов $I_{NM} = 40$ и $\Delta h = 0{,}31$, треугольником показано значение $E_\varepsilon \Delta_{\theta,\hat{\theta}_{(s)}}$.

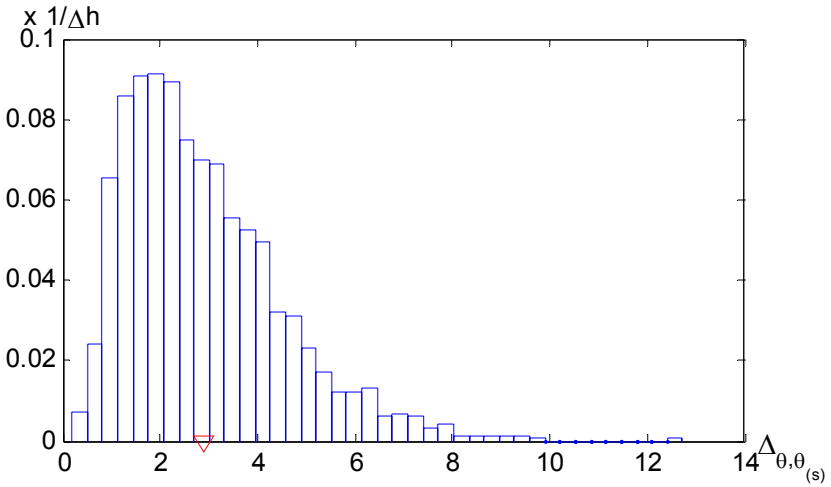


Рис. 24 б). Эмпирический закон распределения случайной величины $\Delta_{\theta,\hat{\theta}_{(s)}}$.

Проведем модельные расчеты этих характеристик ($\Delta_{\theta,\hat{\theta}_{(s)}}$ и $\Delta_{\Phi,\Phi_{(s)}}$) для различных значений *s* и сведем полученные результаты в табл. 2.

Таблица 2.

	s=1	*s=2*	*s=3*
$\max\limits_{\varepsilon} \Delta_{\theta,\hat{\theta}_{(s)}}$	12.7252	12.7252	12.7252
$\min\limits_{\varepsilon} \Delta_{\theta,\hat{\theta}_{(s)}}$	0.1880	0.1880	0.1880
$E_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$	2.8925	2.8925	2.8925
$\max\limits_{\varepsilon} \Delta_{\Phi,\Phi_{(s)}}$	0	0.7627	0.7627
$\min\limits_{\varepsilon} \Delta_{\Phi,\Phi_{(s)}}$	0	-17.9954	-17.9954
$E_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$	0	0.0045	0.0045

Можно заметить, что значения характеристик $\Delta_{\theta,\hat{\theta}_{(s)}}$ и $\Delta_{\Phi,\Phi_{(s)}}$ не зависят от схемы обработки данных. Это можно объяснить тем, что классический план эксперимента является насыщенным, т.е. для него выполняется равенство $k = m$ (т.е. количество параметров модели равно числу точек спектра оптимального плана). Такая особенность проявляет себя в том, что матрица F, фигурирующая в расчетах и входящая в выражения для дисперсионных матриц $\sigma^2 M^{-1}\left(\xi^*\right) = \sigma^2\left(F^T F\right)^{-1} = \sigma^2 M_{(1)}^{-1}\left(\xi\right)$ и $M_{(s)}^{-1}\left(\xi^*, y\right)$, является квадратной и невырожденной.

Случай *m>k, D*- оптимальный план.

Пусть вектор истинных значений параметров имеет вид $\theta^T = \left(2,-5,4\right)$, базисный вектор — $f^T(x) = (1, x, x^3)$, $x \in X = [-1,1]$, D-оптимальный план экспериментов в этом случае имеет вид $\xi^* = \begin{Bmatrix} -1.0 & -0.575 & 0.575 & 1 \\ 4 & 4 & 4 & 4 \end{Bmatrix}$. В Табл. 3 приведены значения характеристик для различных схем.

Таблица 3.

	s=1	*s=2*	*s=3*
$\max\limits_{\varepsilon} \Delta_{\theta,\hat{\theta}_{(s)}}$	9.8094	10.6659	9.8094
$\min\limits_{\varepsilon} \Delta_{\theta,\hat{\theta}_{(s)}}$	0.1188	0.0808	0.1188
$E_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$	2.4840	2.6296	2.6296
$\max\limits_{\varepsilon} \Delta_{\Phi,\Phi_{(s)}}$	0	0.4218	0.4214
$\min\limits_{\varepsilon} \Delta_{\Phi,\Phi_{(s)}}$	0	-6.9067	-8.0408
$E_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$	0	0.1491	0.0049

Реализации двумерной случайной величины $(\Delta_{\theta,\hat{\theta}_{(1)}}, \Phi_{(s)})$ для различных значений s представлены на рис. 25.

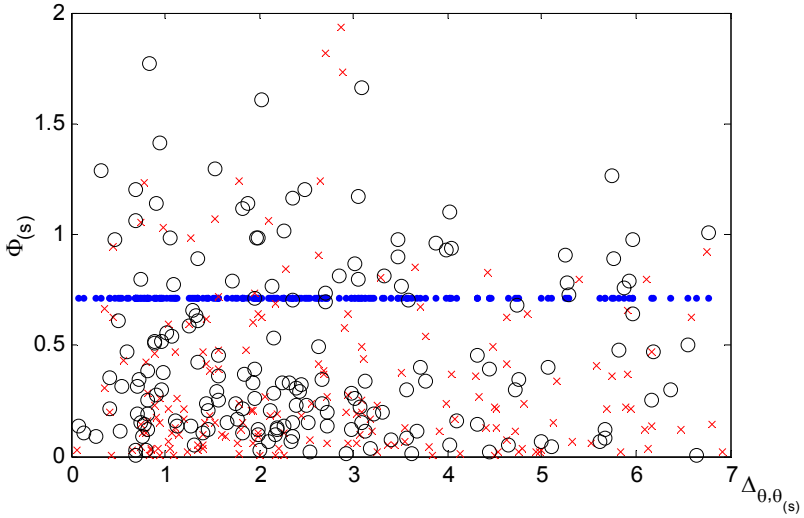


Рис. 25. Реализации случайной величины $(\Delta_{\theta,\hat{\theta}_{(1)}}, \Phi_{(s)})$, $(\Delta_{\theta,\hat{\theta}_{(1)}}, \Phi_{(1)})$ — точки, $(\Delta_{\theta,\hat{\theta}_{(2)}}, \Phi_{(2)})$ — крестики, $(\Delta_{\theta,\hat{\theta}_{(3)}}, \Phi_{(3)})$ — кружки.

Гистограмма закона распределения двумерной случайной величины $(\Delta_{\theta,\hat{\theta}_{(1)}}, \Phi_{(s)})$ для s=3 показана на рис. 26. Здесь $I_{NM,1} = 30$, $I_{NM,2} = 60$, $\Delta h_1 = 0.39,\ \Delta h_2 = 0.19.$

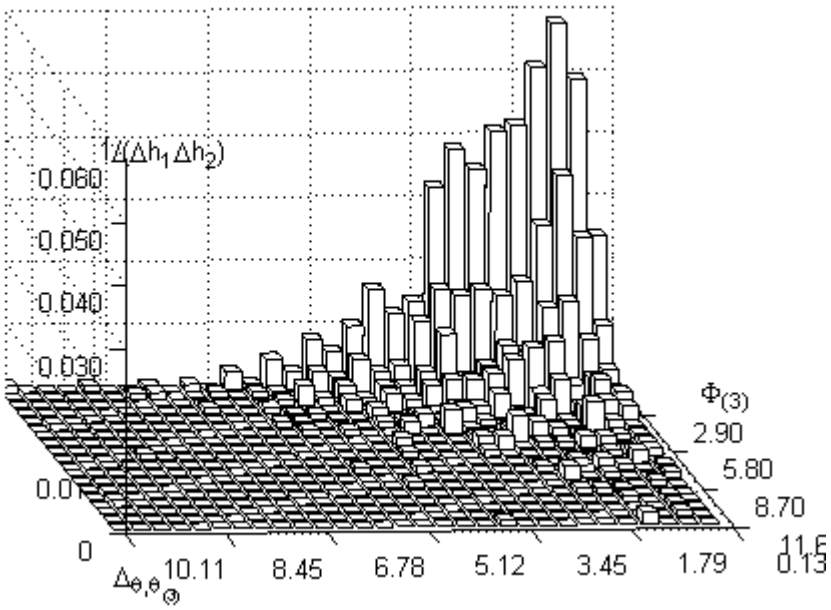


Рис. 26. Гистограмма закона распределения двумерной случайной величины $(\Delta_{\theta,\hat{\theta}_{(1)}}, \Phi_{(s)})$ для s=3.

Исследуем поведение характеристик $\Delta_{\theta,\hat{\theta}_{(1)}}$ и $\Delta_{\Phi,\Phi_{(s)}}$ в зависимости от значений N и σ. Для модели с параметрами $\theta^T = (2{,}5, -3{,}4)$, базисным век-

тором $f^T(x)=(1,x,x^2,x^3)$, $x\in X=[-1,1]$ воспользуемся D- оптимальным планом экспериментов $\xi^*=\begin{Bmatrix}-1.0 & -0.447 & 0.447 & 1\\ n_1 & n_1 & n_1 & n_1\end{Bmatrix}$, $\sum_{i=1}^{4} n_i = N$, $n_1=n_2=n_3=n_4$. Число модельных просчетов равно NM=100. На рис. 27 показаны величины $\min\limits_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (сплошная линия), $\max\limits_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (пунктирная линия), $E_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (штрих-пунктирная линия), а на рис. 28 — $\min\limits_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$ (сплошная линия), $\max\limits_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$ (пунктирная линия), $E_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$ (штрих-пунктирная линия) при изменении N=8(4)36 и при $\sigma=2$

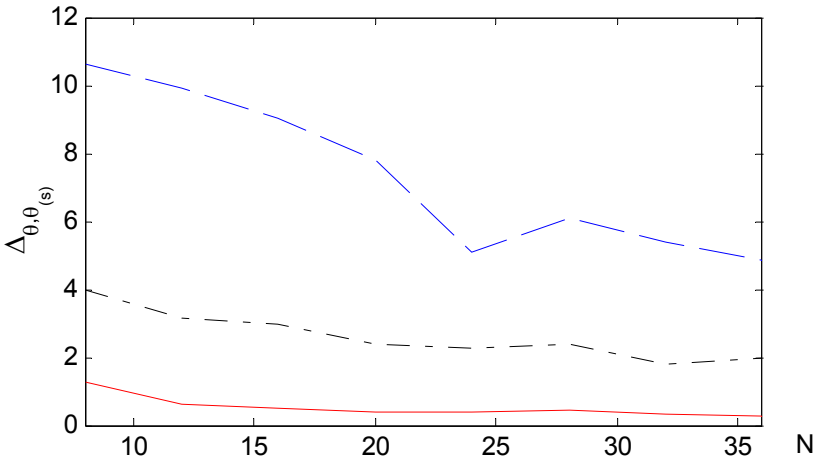


Рис. 27. Величины $\min\limits_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (сплошная линия), $\max\limits_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (пунктирная линия), $E_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (штрих- пунктирная линия).

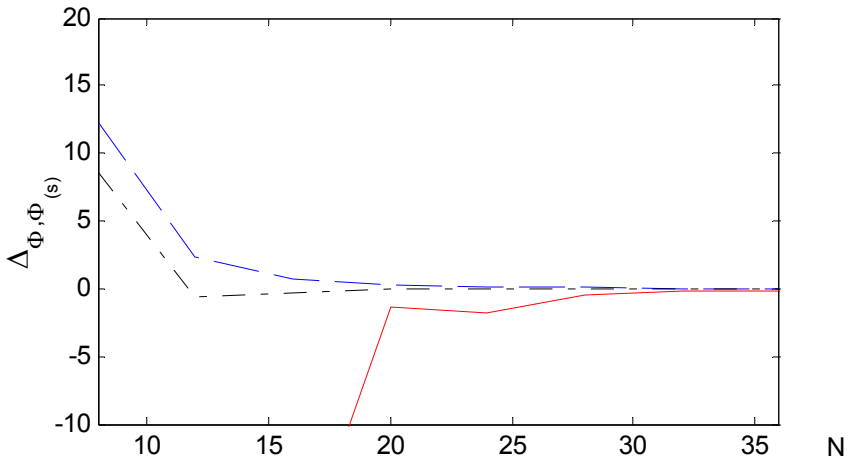


Рис. 28. Величины $\min\limits_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$ (сплошная линия), $\max\limits_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$ (пунктирная линия), $E_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$ (штрих-пунктирная линия).

Те же характеристики, но как функции σ, приведены на рис. 29 и 30.

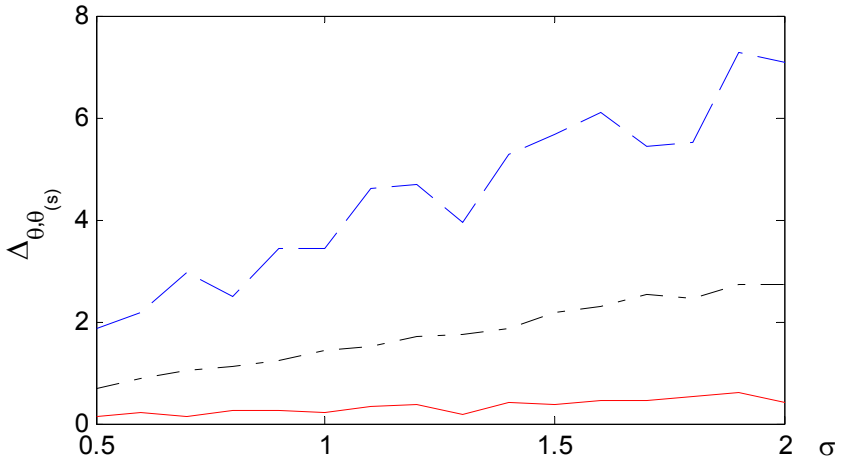


Рис. 29. Характеристики $\min\limits_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (сплошная линия), $\max\limits_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (пунктирная линия), $E_{\varepsilon}\Delta_{\theta,\hat{\theta}_{(s)}}$ (штрих- пунктирная линия), как функции σ.

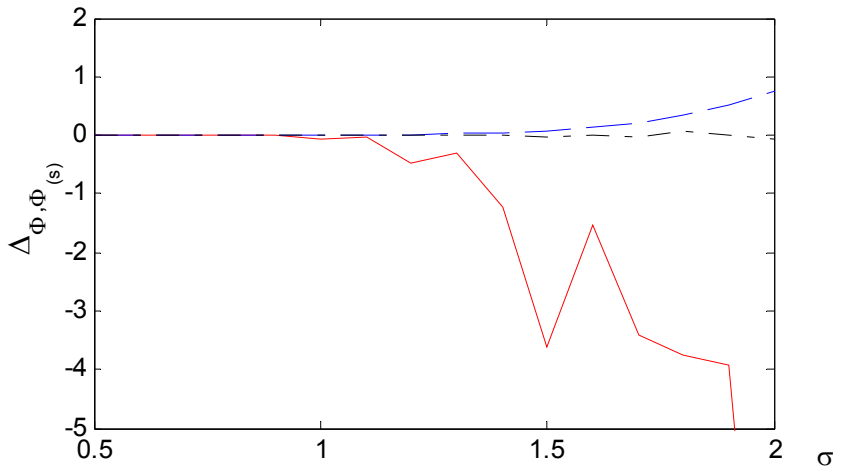


Рис. 30. Характеристики $\min_{\varepsilon} \Delta_{\Phi,\Phi_{(s)}}$ — сплошная линия, $\max_{\varepsilon} \Delta_{\Phi,\Phi_{(s)}}$ — пунктирная линия, $E_{\varepsilon}\Delta_{\Phi,\Phi_{(s)}}$ — штрих- пунктирная линия, как функции σ.

3. Переход от оптимальных планов к эффективным стратегиям — это назревшая необходимость

Вышерассмотренными примерами и модельными просчетами, таким образом, проиллюстрирован ответ на вопрос: можно ли для неоптимального плана получить характеристики для дисперсии оценки функции отклика не хуже, чем для оптимального, т. е., например, для неоптимального плана максимальное значение этой функции будет меньше, чем соответствующее значение для оптимального плана? Ответ — положительный. Отсюда можно сделать вывод, что оптимальность плана, построенного в соответствии с априорными предположениями о дисперсии ошибок наблюдения, не гарантирует оптимальности этого же плана с учетом апостериорных данных, т. е. оптимальный план фактически (в действительности, реально) может таковым не являться. Вышеприведенные расчеты и иллюстрации относятся к случаю, когда и оценивание параметров модели и оценивание дисперсии проводятся в соответствии с обобщенным (взвешенным) методом наименьших квадратов, т. е. с учетом апостериорных оценок дисперсий наблюдений, а планирование эксперимента проводилось на основе априорной информации о дисперсии будущих наблюдений. Предположим, что экспериментатор ошибся и оценил параметры в соответствии с обычным МНК (в соответствии с гипотезой о постоянстве в точках спектра плана дисперсии ошибок наблюдений). В тоже время, реальная дисперсия оценки функции отклика будет оценена с учетом апостериорных оценок дисперсий (т.е. с помощью обобщенного МНК). Результаты моделирования для этого случая приведены в работах [5], [6], [7], [8], [9].

В заключении приведем некоторые общие замечания по результатам рассмотренных выше примеров и модельных расчетов и сделаем соответ-

ствующие выводы. Оптимальные планы экспериментов, построенные для априорных предположений об ошибке наблюдений, с точки зрения апостериорной информации о ней в общем случае таковыми могут не являться. В частности, именно поэтому для них в общем случае может не выполниться утверждение теоремы эквивалентности и оптимальности, если перейти от априорных значений дисперсий к апостериорным. Таким образом, можно констатировать, что оптимальных (в классическом понимании) планов экспериментов не существует и такие планы, в принципе, не могут быть построены до проведения экспериментов (априори). Это, в сою очередь, означает, что, по-видимому, целесообразно заменить утратившее смысл словосочетание «оптимальный план» на какое-либо другое. Мы предлагаем использовать, как нам представляется, более подходящее понятие - «эффективная стратегия экспериментирования». Под ним мы будем понимать наилучший (эффективный) среди возможных вариантов вариант (стратегию) управления экспериментом. Таким образом, мы предлагаем использовать термин «стратегия» взамен слова «план», а слово «эффективный» использовать вместо слова «оптимальный». Первая замена может быть объяснена тем, что управление экспериментом перестает быть процедурой детерминированной. Если бы у экспериментатора была возможность повторить процедуру экспериментирования сначала, то получилась бы, в общем случае, другая (новая, отличная от проделанной первоначально, до этого) последовательность экспериментов и соответствующих им наилучших (в смысле снимаемой с объекта информации, т.е. наиболее информационно емких) условий экспериментирования. Таким образом, оптимальной процедуры экспериментирования (оптимального плана экспериментов), который можно было бы синтезировать до начала проведения экспериментов, в общем случае, не существует. В соответствии с этим, заранее не известна и оптимальная последовательность шагов экспериментирования, общее число экспериментов, общие затраты на экспериментирование, которые потребуются для достижения заданных пороговых значений для точностных характеристик оценок модели регрессии и т.д. Остается лишь на каждом шаге экспериментирования пытаться достичь наилучших изменений этих характеристик, при этом до проведения экспериментов мы не уверены в том, будут ли эти шаги действительно лучшими, не говоря уже об общей их оптимальности, что предполагается при построении планов экспериментов в рамках классической теории ПЭ.

4. Пакет программ для моделирования характеристик оптимальных классических планов экспериментов в среде MATLAB

Вышеприведенные расчеты были произведены с помощью программ, разработанных в среде MATLAB. Программы и алгоритмы разработанны авторами программ синтеза, моделирования и сравнительного

анализа эффективных стратегий экспериментирования, априорных и апостериорных оценок параметров и моделей регрессии. Очевидным образом алгоритмы и соответствующие им программы обобщаются на многомерный случай и более общие постановки задач.

Ниже показаны некоторые части программ (алгоритмов):

Расчёт информационной матрицы $M(\xi, y) = \sum_{i=1}^{m} \hat{\sigma}_{\bar{y}}^{-2}(x_i) f(x_i) f^T(x_i)$ по новому подходу:

```
PLAN=[
 -1        4
 -0.447 4
  0.447 4
  1       4];

for i=1:n
  for j=1:m
     X(j)=PLAN(i,j);
  end
  f=Model(X);
  for j=1:k
     F(i,j)=f(j);
  end
end

ran=Random('norm',0.,sigma,n,100);

for i=1:n
    ransr=sum(ran(i,1:PLAN(i,m+1)))/PLAN(i,m+1);
    dispers_sum=0.;
      for j=1:PLAN(i,m+1)
       dispers_sum=dispers_sum+(ran(i,j)-ransr)^2;
      end
    dispers(i)=dispers_sum/((PLAN(i,m+1)-1)*PLAN(i,m+1));
end
D2=diag(dispers);
Mwave=inv((F'*inv(D2))*F);
```

Здесь n — количество точек плана, m — количество переменных регрессионной модели, Mwave — $M(\xi, y)$, PLAN — план (стратегия) эксперимента, функция Model(X), возвращает значение базисного вектора модели $f(x)$ в некоторой точке $x(X)$.

На официальном сайте авторов http://www.fb.nstu.ru/~experiment/practice.htm представлены програмы (алгоритмы и тексты) показывающие: «неэффективность» оптимальных классических планов экспериментов; не инвариантность таких планов по области экспериментирования; сравнения оптимальных планов и эффективных стратегий; асимптотические свойства информационной матрицы нового подхода; программы построения эффек-

тивных стратегий экспериментов (новый подход), построение эффективной стратегии, где для уточнения оценок параметров используется bootstrep-метод и другие.

5.Выводы и замечания

Использование классических оптимальных планов экспериментов (см. [1], [2], [3]) на практике может приводить к неожиданным результатам: фактическая эффективность планов (в смысле фактических апостериорных значений функционалов от дисперсий оценок параметров модели или оценки регрессионной модели) может быть намного меньше ожидаемой и гарантируемой, например теоремой эквивалентности (см. тождество (7)).

В свою очередь, неоптимальный план эксперимента может быть более эффективным, чем оптимальный.

Неэффективность оптимального плана может быть более существенной, если учесть, что базисный вектор регрессионной модели $f(x)$, как правило, неизвестен.

По всей видимости, одним из выходов в данной ситуации может служить использование последовательного планирования экспериментов, например, на симплексных структурах или организация последовательной схемы экспериментирования с использованием апостериорной дисперсии оценок функции отклика. Выше мы аргументировали использование в таких случаях словосочетания «эффективные стратегии управления экспериментом».

В связи со всем вышесказанным, каталогами классических оптимальных планов экспериментов (см., например, [4]) следует пользоваться с определенной осторожностью.

Литература

1. *Kiefer J.* General equivalence theory for optimum designs (approximate theory) // The Annals of Mathematical Statistics.— 1974.— V.2.— N.5.— P.849–879.
2. *Kiefer J., Wolfowitz J.* The equivalence of two extremum problems // Canadian Journal of Mathematics.— 1960.— V.12.— P.363–366.
3. *Федоров В. В.* Теория оптимального эксперимента.— М.: Наука, 1971.
4. Таблицы планов эксперимента для факторных и полиномиальных моделей / *Бродский В.З., Бродский Л.И.* и др.— М.: Металлургия, 1982.— 752 с.
5. *Наумов А. А.* Эффективность моделей экспериментирования // Актуальные проблемы электронного приборостроения АПЭП-2002: Тез. докл. VI Международ. конф.— Новосибирск: Изд-во НГТУ, 2002.— Т.6.— С.56–59.

6. *Наумов А. А., Веснин С. В.* К задаче последовательной обработки данных на симплексных структурах // Планирование эксперимента, идентификация, анализ и оптимизация многофакторных систем.— Новосибирск: Новосиб. электротехн. ин-т, 1990.— С.56–60.
7. *Naumov A.* Investigation of Experiment Design Effectiveness and First Steps to Random Experimenting Strategy. I // Наука и образование в условиях социально-экономической трансформации общества: Материалы V Международ. науч. конф. (30–31 мая 2002).— Гродно: Изд-во ГПУ, 2002.— Ч.2.— С.217–222.
8. *Наумов А. А., Сенич В. В.* Эффективное управление экспериментом.— Новосибирск: ОФСЕТ, 2003.
9. *Naumov A. A.* Experiments Design Investigation: The Model II // Информатика и проблемы телекоммуникаций: Тез. докл. Международ. науч.-техн. конф.— Новосибирск: Изд-во СИБГУТИ, 2002.— С.114–116.
10. *Денисов В. И.* Математическое обеспечение системы ЭВМ-экспериментатор.— М.: Наука, 1977.
11. *Денисов В. И., Попов А. А.* Пакет программ оптимального планирования экспериментов.— М.: Финансы и статистика, 1986.
12. *Gribik P. R., Kortanek K. O.* Equivalence Theorems and Cutting Plane Algorithms for a Class of Experimental Design Problems // SIAM Journal Appl. Math.— 1977.— V.32.— N.1.— P.232–259.
13. *Bandemer H.* et al. Theorie und Anwendung der optimalen Versuchsplanung I. Handbuch zur Theorie.— Berlin: Akademie-Verlag, 1977.

УДК 618.3:519.24

MATLAB-ПРИЛОЖЕНИЕ ДЛЯ ПРОЕКТИРОВАНИЯ НОВЫХ СПЛАВОВ

Нургаянова О. С., Попов Д. В., Ганеев А. А.
Уфимский государственный авиационный технический университет, Уфа,
e-mail:onurgayanova@yandex.ru

Процесс проектирования новых сплавов основывается на проведении экспериментов. При этом большая часть времени отводится на проведение всевозможных опытов и исследование свойств полученных образцов, значительные средства расходуются на поставку исходных материалов, часть которых в дальнейшем может оказаться негодной для последующего применения и переработки. Для эффективного распределения всех ресурсов необходимо планирование в некотором смысле оптимального эксперимента, позволяющего получить сплав с заданными свойствами и рассчитать расходы по его получению.

Сокращения затрат можно добиться путем уменьшения числа опытов в матрице плана эксперимента, т. е. необходимо оптимизировать матрицу планирования, тогда целевая функция будет иметь вид:

$$E(N) \rightarrow max. \qquad (1)$$

В предположении, что за один раз получается один состав и совмещение затрат не рассматривается, оптимизируемая функция $E(N)$ от числа экспериментов N при заданных модели M и критерия оптимальности Ψ примет вид:

$$E(N) = \Psi^*(X^N) / S(X^N), \qquad (2)$$

где

$$X^N = Find(N, M, \Psi^*). \qquad (3)$$

Здесь $E(N)$ — оптимизируемая функция; N — количество опытов в матрице плана эксперимента; X^N — матрица D-оптимального плана; $S(X^N)$ — стоимость испытаний, которые необходимо провести; $\Psi^*(X)$– оптимальное значение выбранного критерия оптимальности ψ на множестве матриц плана эксперимента Ω_x; $Find(N,M,\Psi^*)$ — функция поиска оптимального плана; $M(\Phi)$ — вид математической модели влияния X на G; Φ — факторы (химические элементы и их соединения).

Теперь рассчитаем затраты, приходящиеся на разработку сплава. Они будут включать затраты на проведение эксперимента и затраты на внедрение новой технологии. Затраты на проектирование нового сплава складываются из материальных затрат и расходов на оплату труда:

$$S(X^N) = \sum_{j=1}^{m} S(X_j), \qquad (4)$$

где

$$S(X_j) = \sum_{i=1}^{n} \omega_i \cdot m_i \cdot (c_i + T_i) \cdot r \cdot d\, e \cdot \sum t_1 \cdot N\, a \cdot \sum p \cdot t_2 . \qquad (5)$$

Здесь $\boldsymbol{X^N}$ **—** D-оптимальная матрица планирования; $S(X^N)$ — себестоимость полученных образцов; $X_j=(x_j^1, x_j^2, \ldots, x_j^n)$ — химический состав сплава; x_j^i — массовая доля i-го элемента; $S(X_j)$ — затраты, приходящиеся на одно испытание; стоимостные затраты: ω_i — массовая доля i-го элемента выраженная в %; m_i — масса химического соединения содержащего i-ый элемент (кг.); c_i — стоимость i-го элемента (у.е.); T_i — затраты на транспортировку i-го элемента (у.е.); r — число режимов термообработки; d — количество дублирований; энергетические затраты: e — стоимость одного кВт·ч (у.е.); N — мощность оборудования (Вт.); t_1 _ время работы оборудования (ч.); затраты на рабочую силу: p — часовая оплата труда рабочих и инженеров; t_2 _ время, затрачиваемое на технологический процесс и исследование физико-химических свойств полученных образцов; a — процент отчислений.

Материальные затраты складываются из стоимости приобретенного сырья с учетом НДС и транспортных расходов и расходов на электроэнергию.

В дальнейшем будем считать, что функция $S(X_j)$ линейно зависит от своих параметров.

Из формул (2) и (3) видно, что основные сложности связаны с выбором модели $M(\Phi)$ и построением соответствующего плана эксперимента X. К решению данной проблемы предлагается следующий подход.

Центральным в теории планирования эксперимента является принцип оптимальности плана. В соответствии с ним план эксперимента должен обладать некоторыми оптимальными свойствами с точки зрения определенного, заранее выбранного критерия оптимальности плана.

Проведенный анализ критериев оптимальности планов экспериментов показал, что наиболее предпочтительным для оценок коэффициентов модели является критерий D-оптимальности, обеспечивающий минимум определителя (*det*) матрицы дисперсий-ковариаций:

$$\Psi^D = min\, det(\vec{X}^T \vec{X})^{-1} . \qquad (6)$$

Этот критерий связан с оценкой уравнения регрессии в целом, получение минимальной обобщенной дисперсии коэффициентов обеспечивает достаточные предсказательные свойства модели внутри области эксперимента. Согласно теореме [2], если план является D-оптимальным, то он является A- и E- оптимальным. Тем более что он является наиболее информативным, поскольку использует все элементы корреляционной матрицы.

Таким образом, оптимизационная задача нахождения D-оптимального плана эксперимента, состоящего из заданного числа N опытов, может быть сформулирована следующим образом:

$$|(X^{*T}X^*)| = \min|(X^TX)|, \text{ где } X \in \Omega_x, \qquad (7)$$

Здесь X^* — D-оптимальный план [2].

Теперь перейдем к построению самого D-оптимального плана. В связи с большой размерностью задачи предлагается воспользоваться каким-либо из эвристических методов, в данном случае генетическим алгоритмом.

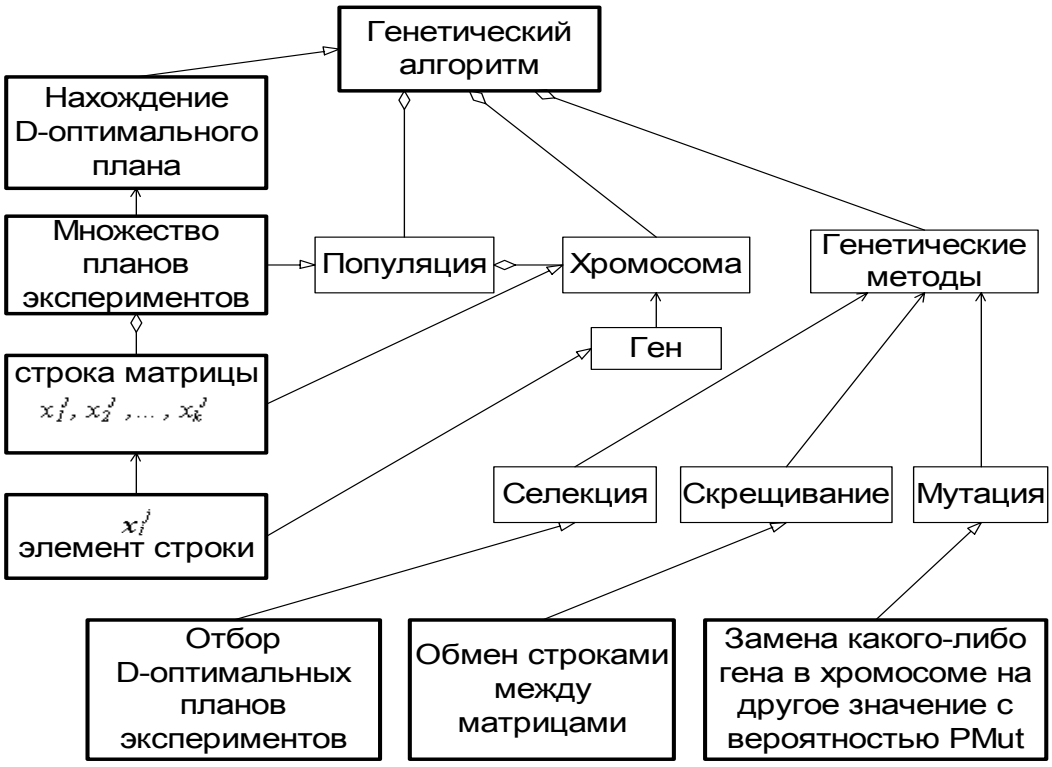


Рис. 1. Применение генетического алгоритма для построения D-оптимального плана.

При построении D-оптимального плана на основе генетического подхода используются следующие стадии: генерация случайной начальной популяции и генетические операторы — отбор элиты, скрещивание, селекция и мутация. На каждом шаге-итерации развития популяции динамически изменяется ее состав. Эти изменения моделируются генетическими операторами, которые применяются до тех пор, пока не будет завершена работа алгоритма.

Алгоритм комбинирует случайный поиск D-оптимальных планов «от плана к плану» с использованием генетических операторов (скрещивание, мутация, селекция), улучшающих популяцию планов с точки зрения оценочного критерия.

Определим основные понятия ГА применительно к построению D — оптимального плана (рис. 1):

- популяция — всевозможные планы экспериментов.
- хромосомы — строки матрицы плана эксперимента.
- гены — элементы строк (концентрации химических элементов в сплаве).

С учетом выше сказанного была разработана система планирования эксперимента *«OptimalDE»*, позволяющая генерировать матрицы планирования для конкретного сплава и по ним рассчитывать стоимость проведения эксперимента (рис. 2). Цены на химические элементы содержаться в загружаемой базе.

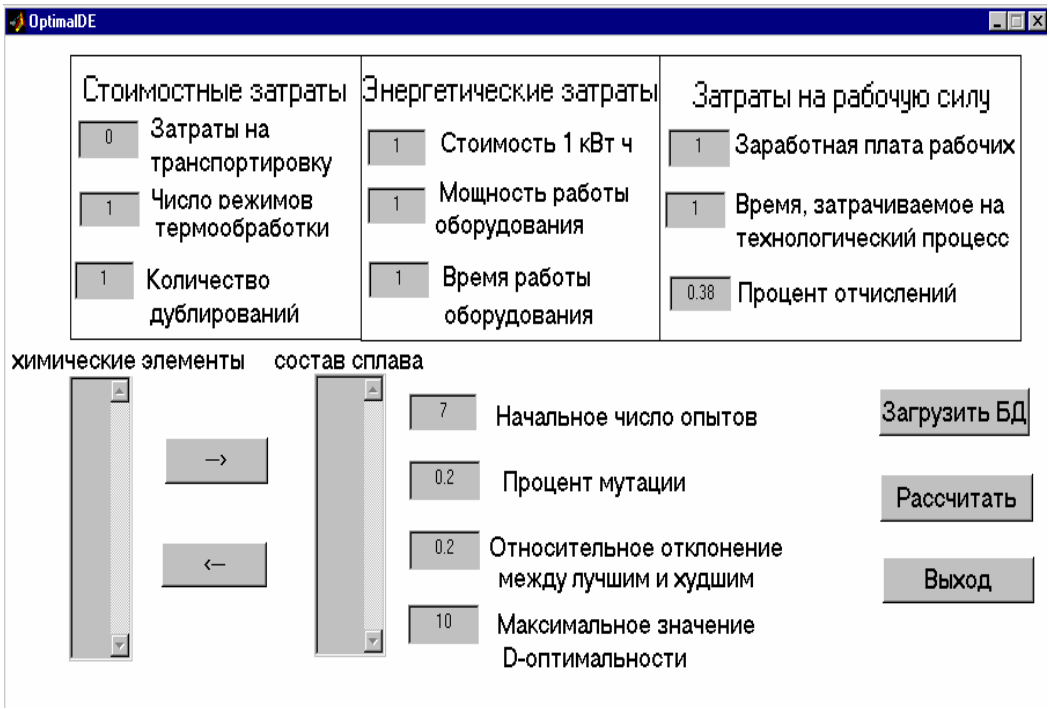


Рис. 2. Главное окно программы *«OptimalDE»*.

Программа написана на языке системы MATLAB.

На основе разработанной системы были получены следующие результаты.

Для 12-ти элементного сплава (*Ni, Co, Cr, Sn, Pb, Cd, Mg, Al, Zn, Fe, C, Mo*) стоимость эксперимента составила 938000 рублей, т. е. на проведение одного опыта требуется порядка 55176 рублей (см. файл *Result12.txt*). Значение *D*-оптимальности плана составляет 0,0253.

В целом же такой подход к разработке новых сплавов с использованием системы планирования эксперимента позволяет ускорить процесс проектирования новых сплавов, минимизировать затраты на их синтез и рассчитать стоимость проведения экспериментов.

Литература

1. *Тамразов А. М.* Планирование и анализ регрессионных экспериментов в технологических исследованиях.— Киев: Наукова думка, 1987.— 176 с.
2. *Хартман К. и др.* Планирование эксперимента в исследовании технологических процессах.— М.: Мир, 1977.— 552 с.

3. Прикладной регрессионный анализ / И. Вучков, Л. Бояджиева, Е. Солаков / Пер. с бол. и предисл. Ю. П. Адлера.— М.: Финансы и статистика, 1987.— 239 с.
4. *Налимов В. В., Голикова Т. И.* Логические основания планирования эксперимента.— М.: Металлургия,1980.— 152с.
5. *Ганеев А. А.* Повышение жаропрочности литейных никелевых сплавов с использованием методов активного и пассивного экспериментов: Автореф... докт.техн. наук: 05.16.04 Литейное производство.—Екатеринбург, 2000.— 42 с.

УДК 538.221: 669.14

ПРЕОБРАЗОВАНИЕ ЭЛЕКТРОННОЙ ПЛОТНОСТИ СОСТОЯНИЙ ЧИСТОГО МЕТАЛЛА В СРЕДЕ MATLAB

Обухов А. Г., Волошинский А. Н.
Тюменский государственный нефтегазовый университет, Тюмень,
e-mail:aobukhov@tgngu.tyumen.ru

1. Постановка задачи

Исследование процессов переноса в металлах и сплавах представляет собой одну из важнейших задач современной физики твердого тела. Основным методом теоретического исследования и прогнозирования кинетических свойств металлических сплавов является модифицированный вариант приближения когерентного потенциала [1, 2].

Свойства конкретных переходных металлов и их сплавов во многом определяются структурой полосы проводимости, значениями плотности состояний и ее производной вблизи уровня Ферми, и так далее [3]. Поэтому расчеты электронных, магнитных и кинетических свойств металлов и сплавов часто проводятся с использованием реалистической модели плотности состояний металлических систем и привлечением результатов расчетов зонной структуры металлов, полученных из «первых принципов» [4].

Это особенно важно при рассмотрении систем, образованных переходными металлами, плотности состояний которых являются сложными функциями энергии [5, 6]. Математические трудности, связанные с введением в расчет сложных кривых плотностей состояний металлов и сплавов стали в последнее время вполне преодолимы благодаря возросшей мощности вычислительной техники, широко используемой при итерационном решении уравнений приближения когерентного потенциала. Поэтому многие исследуемые свойства получили не только качественное, но и количественное объяснение.

Электронные плотности состояний переходных металлов, как результат зонных расчетов, являются немонотонными, сложными функциями энергии. График плотности состояний титана, например, имеет вид (рис. 1). Существенным моментом является то, что, спектр представлен относительно небольшим количеством характерных точек, абсциссы которых расположены неравномерно на энергетической оси. Числовые значения координат этих точек, как правило, представляются в виде таблицы, или двумерного массива (рис. 2). В данном примере электронная плотность состояний титана представлена в виде двумерного массива REti размерностью 18×10, в котором первые 9 строк

соответствуют значениям плотности состояний, а следующие 9 строк — соответствующим значениям энергии.

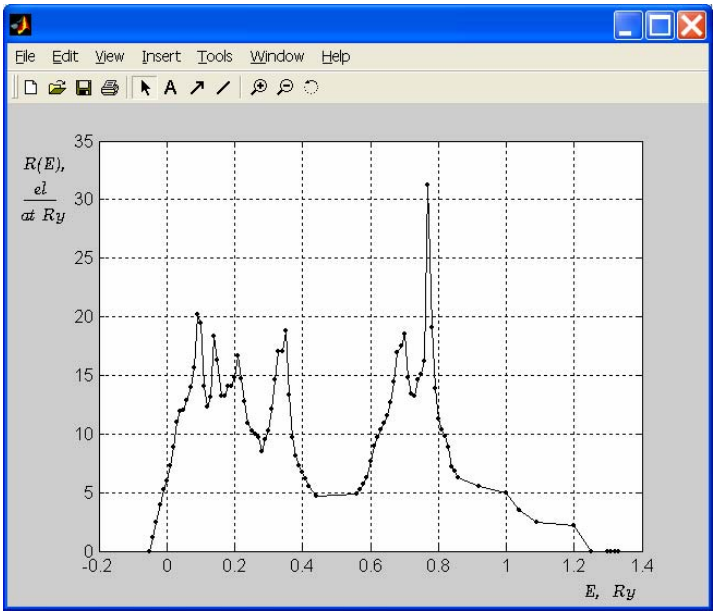
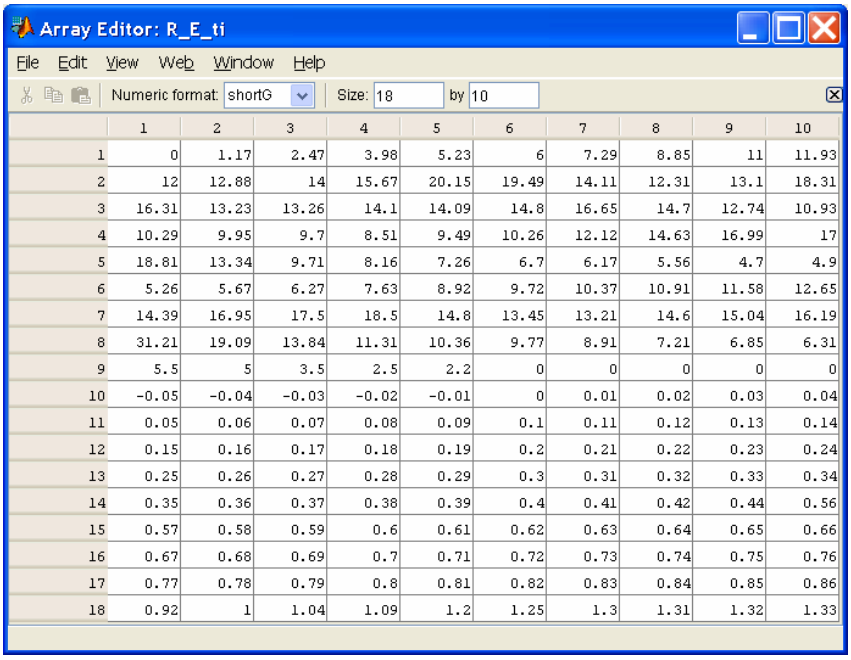


Рис. 1. Графическое изображение электронной плотности состояний титана.



	1	2	3	4	5	6	7	8	9	10
1	0	1.17	2.47	3.98	5.23	6	7.29	8.85	11	11.93
2	12	12.88	14	15.67	20.15	19.49	14.11	12.31	13.1	18.31
3	16.31	13.23	13.26	14.1	14.09	14.8	16.65	14.7	12.74	10.93
4	10.29	9.95	9.7	8.51	9.49	10.26	12.12	14.63	16.99	17
5	18.81	13.34	9.71	8.16	7.26	6.7	6.17	5.56	4.7	4.9
6	5.26	5.67	6.27	7.63	8.92	9.72	10.37	10.91	11.58	12.65
7	14.39	16.95	17.5	18.5	14.8	13.45	13.21	14.6	15.04	16.19
8	31.21	19.09	13.84	11.31	10.36	9.77	8.91	7.21	6.85	6.31
9	5.5	5	3.5	2.5	2.2	0	0	0	0	0
10	-0.05	-0.04	-0.03	-0.02	-0.01	0	0.01	0.02	0.03	0.04
11	0.05	0.06	0.07	0.08	0.09	0.1	0.11	0.12	0.13	0.14
12	0.15	0.16	0.17	0.18	0.19	0.2	0.21	0.22	0.23	0.24
13	0.25	0.26	0.27	0.28	0.29	0.3	0.31	0.32	0.33	0.34
14	0.35	0.36	0.37	0.38	0.39	0.4	0.41	0.42	0.44	0.56
15	0.57	0.58	0.59	0.6	0.61	0.62	0.63	0.64	0.65	0.66
16	0.67	0.68	0.69	0.7	0.71	0.72	0.73	0.74	0.75	0.76
17	0.77	0.78	0.79	0.8	0.81	0.82	0.83	0.84	0.85	0.86
18	0.92	1	1.04	1.09	1.2	1.25	1.3	1.31	1.32	1.33

Рис. 2. Табличное задание электронной плотности состояний титана.

Для введения в численные расчеты электронной плотности состояний необходимо выполнить следующие преобразования для соответствующего ей двумерного массива значений.

Во-первых, из данного массива сформировать двумерный массив размерностью $n \times 2$, один из столбцов которого соответствует значениям энергий, а второй — соответствующим плотностям состояний.

Во-вторых, используя данный ряд значений энергий, сформировать из него массив равноотстоящих узлов, количество которых n может значительно превышать количество первоначально заданных. От величины n зависит точность последующих численных расчетов, поэтому она варьируется в процессе вычислений.

В-третьих, с помощью интерполяции получить массив соответствующих значений плотностей состояний.

Конечно, поставленная задача, и многие другие подобные задачи могут быть решены и традиционными способами формирования циклов, как это делается в языках программирования, однако благодаря методике векторизации, реализованной в системе MATLAB [7], решение подобных задач отличается компактностью, особым изяществом и, как следствие, высокой скоростью вычислений.

2. Преобразование массива плотности состояний в среде MATLAB

Выделив половину исходного массива REti (первые 9 строк в рассматриваемом примере), сохраним его на жестком диске под именем Rti.txt. Поскольку данный файл содержит прямоугольный массив чисел с одинаковым количеством элементов в каждой строке, то для импорта данных из этого файла следует воспользоваться М-функцией *load*, создающей одноименную переменную в рабочей области. Набрав в командном окне строку

>> load Rti.txt

получим в рабочей области переменную Rti размером 9×10 (рис. 3).

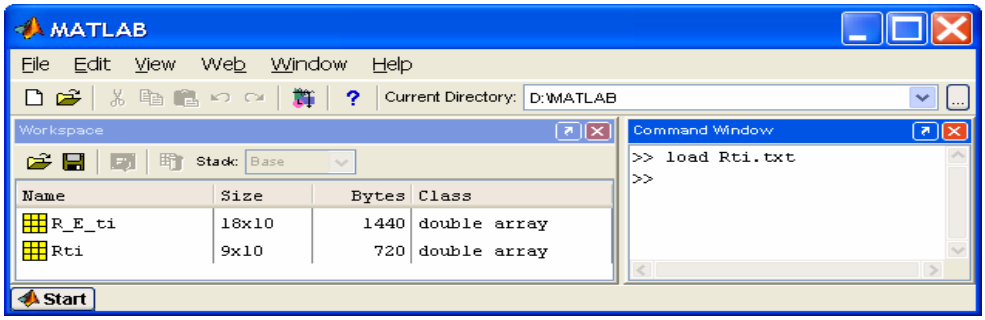
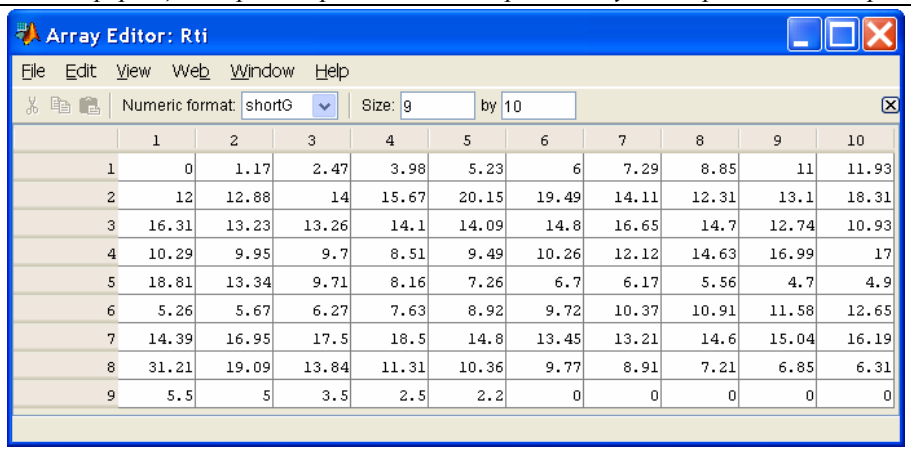


Рис. 3. Переменная Rti в рабочей области как результат импорта данных.

Двойной щелчок левой кнопки мыши или нажатие клавиши Enter на выделенном имени переменной Rti позволяет загрузить окно редактора данных, убедиться в корректности импорта данных, или произвести редактирование (рис. 4).



	1	2	3	4	5	6	7	8	9	10
1	0	1.17	2.47	3.98	5.23	6	7.29	8.85	11	11.93
2	12	12.88	14	15.67	20.15	19.49	14.11	12.31	13.1	18.31
3	16.31	13.23	13.26	14.1	14.09	14.8	16.65	14.7	12.74	10.93
4	10.29	9.95	9.7	8.51	9.49	10.26	12.12	14.63	16.99	17
5	18.81	13.34	9.71	8.16	7.26	6.7	6.17	5.56	4.7	4.9
6	5.26	5.67	6.27	7.63	8.92	9.72	10.37	10.91	11.58	12.65
7	14.39	16.95	17.5	18.5	14.8	13.45	13.21	14.6	15.04	16.19
8	31.21	19.09	13.84	11.31	10.36	9.77	8.91	7.21	6.85	6.31
9	5.5	5	3.5	2.5	2.2	0	0	0	0	0

Рис. 4. Окно редактора данных с загруженным массивом Rti.

Транспонируем полученный массив, выполнив команду:

>> **Gti=Rti'**

В результате получим прямоугольный массив Gti размерностью 10×9 (рис. 5).

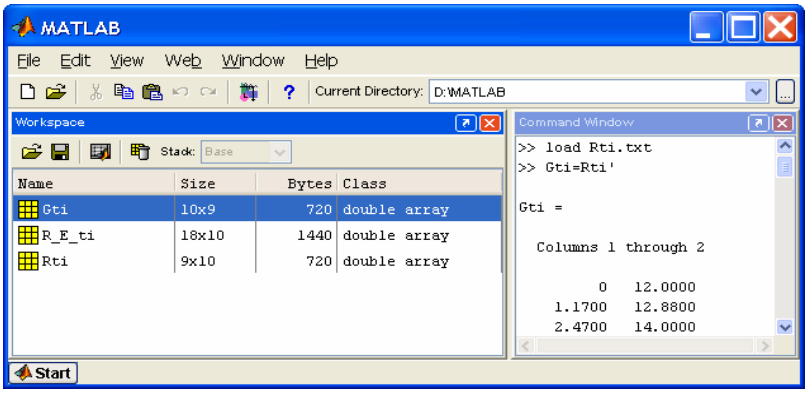


Рис. 5. Транспонированный массив плотности состояний Gti.

Используя одномерное индексирование, получим из него одномерный вектор-столбец хранения y размерностью 90×1 (рис. 6), введя в командном окне оператор:

>> y=Gti(:)

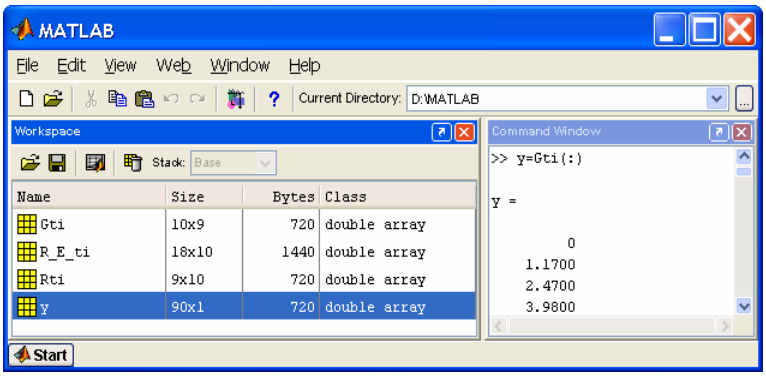


Рис. 6. Одномерный вектор-столбец хранения y.

Одномерный массив y является вектором-столбцом значений плотности состояний, просмотреть который можно в редакторе массивов двойным щелчком мыши (рис. 7).

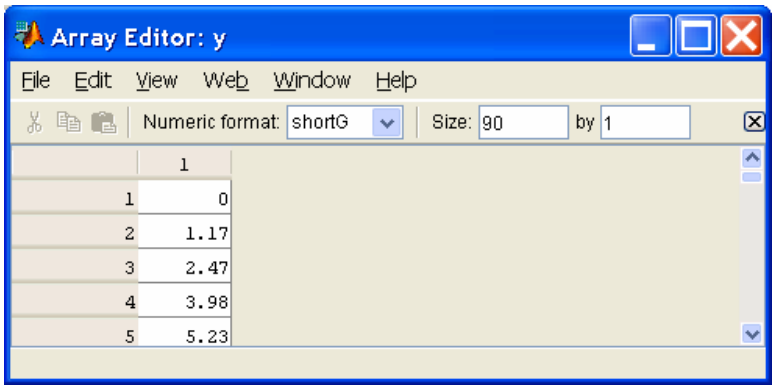


Рис. 7. Окно редактора данных с загруженным массивом y.

Выделив вторую половину исходного массива REti (оставшиеся 9 строк в рассматриваемом примере), сохраним его на жестком диске под именем Eti.txt. Повторяя аналогичные преобразования с содержимым этого файла, а именно, выполнив команды

>> **load Eti.txt**
>> **Eti=Eti'**
>> **x=Eti(:)**

получим одномерный массив x размерностью 90×1 (рис. 8), являющийся вектором-столбцом значений энергий, просмотреть который можно в редакторе данных (рис. 9).

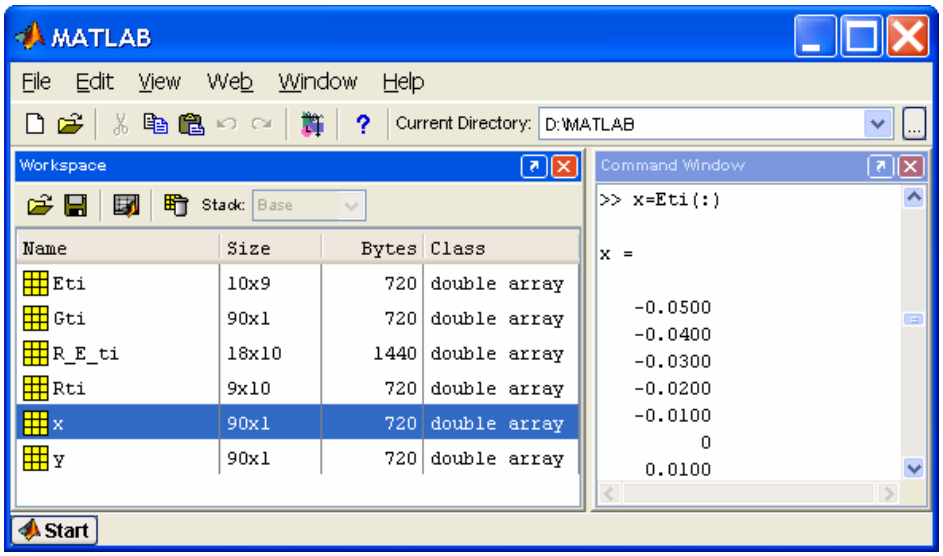


Рис. 8. Одномерный вектор-столбец хранения x.

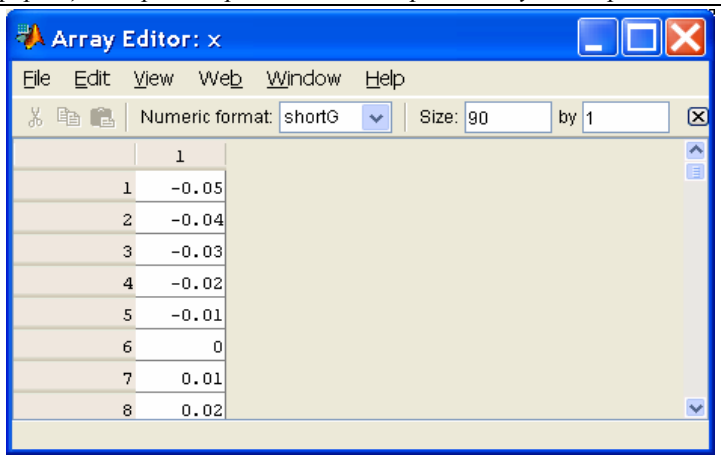


Рис. 9. Окно редактора данных с загруженным массивом x.

Таким образом, первая из сформулированных задач решена.

Для решения второй задачи следует воспользоваться М-функцией **linspace**(x1, x2, n), которая формирует линейный массив равноотстоящих узлов размером $1 \times n$, начальным и конечным элементом которого являются точки x1 и x2. Поскольку массив x имеет начальное и конечное значения соответственно —0.05 и 1.33, а количество значений можно взять, например, $n = 200$, то набрав в командном окне строку

>> **xlin = linspace (—0.05, 1.33, 200)**

получим одномерный массив равноотстоящих узлов размером 1×200, начальным и конечным элементом которого являются значения —0.05 и 1.33.

Используя одномерное индексирование и введя в командном окне оператор

>> **E=xlin(:)**

получим вектор-столбец хранения E, размерностью 200×1 (рис. 10)

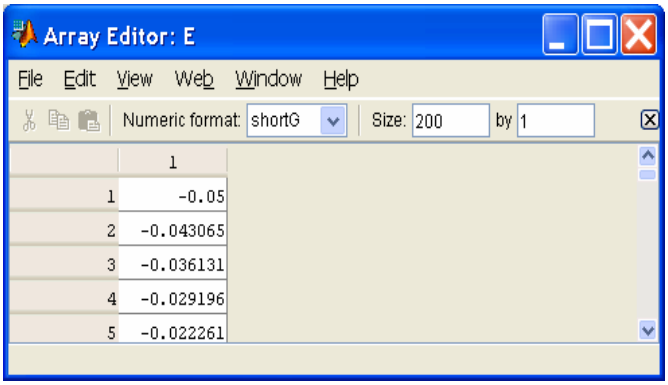


Рис. 10. Окно редактора данных с загруженным массивом E.

Таким образом, решена вторая из сформулированных задач.

Для окончательного решения поставленной задачи необходимо воспользоваться М-функцией одномерной табличной интерполяции **interp1**:

Синтаксис этой функции yi=**interp1**(x,y,xi,'method') — где вектор x определяет точки, в которых заданы значения y, а вектор yi содержит элементы, соответствующие элементам xi и полученные интерполяцией вектора y. Параметр method определяет метод интерполяции. В частности, 'nearest' — ступенчатая интерполяция, 'linear' — линейная интерполяция, 'spline' — кубическая сплайн-интерполяция, 'cubic' — интерполяция многочленами Эрмита.

Применительно к рассматриваемому примеру интерполирование массива y значений электронной плотности состояний многочленами Эрмита осуществляется набором команды:

>> GE = interp1 (x, y, E, 'cubic')

Размерность полученного массива GE 200×1 (рис. 11) и элементами его являются значения электронной плотности состояний титана в равноотстоящих узлах значений энергии.

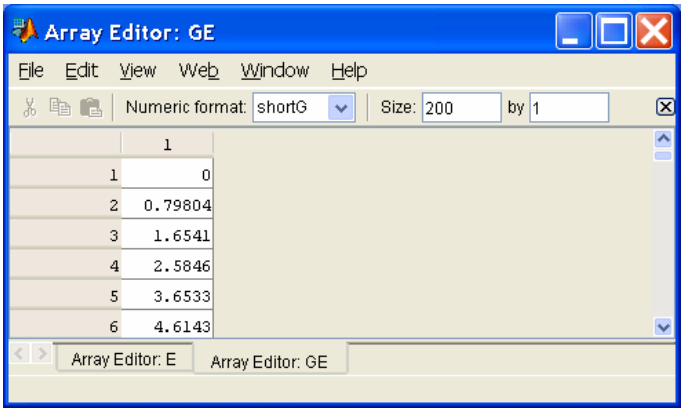


Рис. 11. Окно редактора данных с загруженным массивом GE .

В заключение следует отметить, что при наличии на жестком диске двух файлов с данными Rti.txt и Eti.txt описанная выше процедура преобразования плотности состояний может быть записана в виде:

```
>> load Rti.txt ; Gti=Rti' ; y=Gti(:)
>>  load Eti.txt ; Eti=Eti' ; x=Eti(:)
>>  xlin=linspace(-0.05,1.33,200) ; E=xlin(:)
>> GE=interp1(x,y,E,'cubic')
```

Каждая строка заканчивается нажатием клавиш Shift+Enter, а последняя строка — клавишей Enter. Результатом выполнения такой совокупности команд будет создание четырех одномерных векторов-столбцов y, x, E, GE (рис. 12). Целью преобразований являются последние два массива.

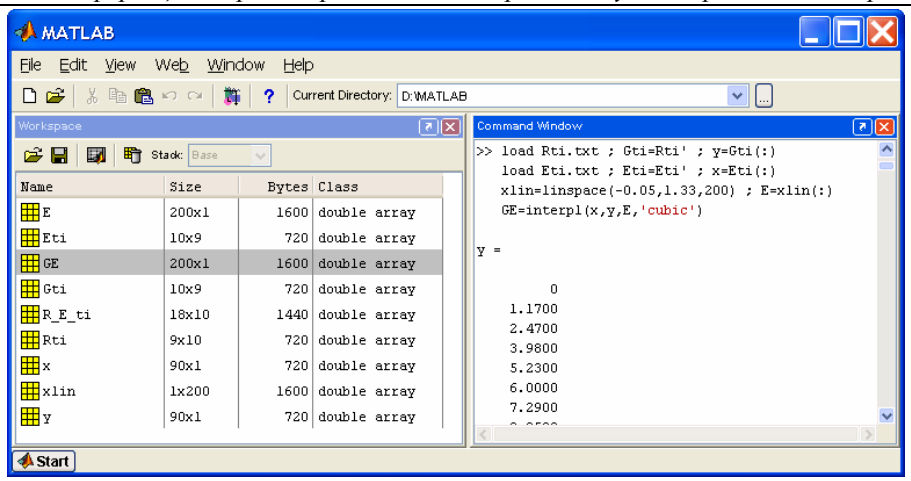


Рис. 12. Результат преобразования массива плотности состояний

График полученной зависимости плотности состояний от энергии представлен на рис. 13 и построен с помощью команды:

>> **plot (E, GE)**

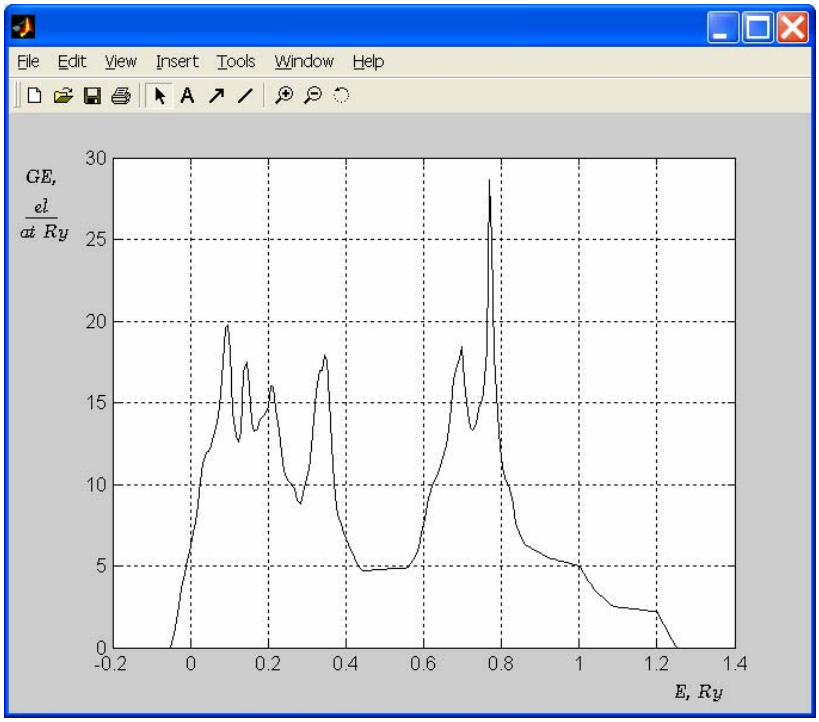


Рис. 13. Графическое изображение проинтерполированной электронной плотности состояний титана GE .

Литература

1. *Циовкин Ю. Ю., Вишнеков Л. Ю., Волошинский А. Н.* Расчет концентрационной зависимости остаточного электросопротивления бинарных сплавов в двухполосном приближении когерентного потенциала // ФММ.— 1991.— №7.— С.48–58.

2. *Циовкин Ю. Ю., Волошинский А. Н.* Расчет температурных и концентрационных зависимостей электросопротивления сплавов в двухполосном приближении когерентного потенциала // ФММ.— 1993.— №75.— Вып.3.— С.25–37.
3. *Харрисон У.* Электронная структура и свойства твердых тел: Пер. с англ. под ред. Алферова Ж. И. Т.1,2. — М.: Мир, 1983.
4. *Эренрейх Г., Шварц Л.* Электронная структура сплавов.— М.: Мир, 1979.— 198 с.
5. *Landolt-Bernstein.* Numerical Data and Functional Relationships in Science and Technology. V.13. Group III Subvolume C. Electron states and Fermi Surface of Elements.— Berlin, Springer, 1984.— 607 p.
6. *Morruzzi V. L., Janak J. F., Williams A. R.* Calculated Electronic Properties of Metals.— New-York: Pergamon Press, 1978.— 202 p.
7. *Потемкин В. Г.* MATLAB 6: Среда проектирования инженерных приложений.— М.: ДИАЛОГ–МИФИ, 2003.— 448 с.

УДК 537.81

АВТОМАТИЗИРОВАННЫЙ РАСЧЕТ ЛОГОМЕТРИЧЕСКИХ ЭЛЕКТРОМЕХАНИЧЕСКИХ ПРЕОБРАЗОВАТЕЛЕЙ В ПАКЕТЕ FEMLAB

Панков М. А., Сбитнев С. А., Шмелев В. Е.
Владимирский государственный университет, Владимир,
e-mail:pank_off@land.ru

Проектирование электромеханических преобразователей для современной промышленности основано на анализе схем замещения и обобщенных схем электрических машин при определенных режимах работы, которые не позволяют найти оптимальные параметры и характеристики устройства заданной конструкции. Это приводит к созданию многочисленных прототипов и проведению серий экспериментов, что является длительным и ресурсоемким процессом. Сокращение затрат проектирования возможно при создании виртуальных моделей электромеханических преобразователей с заданной геометрией, анализ их осуществляется современными методами теории электромагнитного поля.

Математическое описание магнитного поля в различных системах базируется на дифференциальных уравнениях в частных производных. При решении численным методом они преобразуются в систему алгебраических уравнений, решение которой дает аппроксимацию поля в дискретных точках пространства. Сложный характер математической модели заставляет вносить упрощения и делать физические допущения. Наиболее эффективный способ получить распределение магнитного поля системы индукторов состоит в нахождении векторного магнитного потенциала $\vec{A}$ [1]:

$$\vec{B} = rot\vec{A}. \qquad (1)$$

Уравнение магнитостатического поля относительно векторного магнитного потенциала, считая все электрические токи сторонними и учитывая уравнения материальной связи между векторами магнитной индукции $\vec{B}$ и напряженности магнитного поля $\vec{H}$, записывается следующим образом:

$$rot(\mu_a^{-1} rot\vec{A}) = \vec{\delta} + rot(\mu_a^{-1}\vec{B}_r). \qquad (2)$$

Для расчета магнитостатического поля в неоднородной среде с учетом условия калибровки $div\vec{A} = 0$ и $\nu'_a = (\mu'_a)^{-1}$ — удельного магнитного сопротивления среды, занимающей наибольший объем в расчетной области, получим:

$$rot((\mu_a^{-1} - \nu'_a) rot\vec{A}) - (\nabla\nu'_a\nabla)\vec{A} = \vec{\delta} + rot(\mu_a^{-1}\vec{B}_r). \qquad (3)$$

Уравнение (3) позволяет представить магнитостатическое поле в неоднородной среде непрерывным полем векторного магнитного потенциала. При моделировании необходимо обеспечить единственность решения данного уравнения путем дополнения граничными условиями для искомого потенциала или тангенциальной составляющей вектора напряженности магнитного поля на поверхности, ограничивающей расчетную область. Это дает возможность применять конечноразностные и конечно-элементные методы без существенных ограничений, но с хорошей сходимостью, а для анализа структур сложной геометрии лучший результат дает метод Галеркина с конечными элементами [2].

Наиболее удобной программной средой для решения вышеизложенной задачи является среда математического моделирования FEMLAB. В ней удается построить сложную геометрию моделируемого объекта, описав дифференциальными уравнениями в частных производных с учетом граничных условий. В первом приближении была рассмотрена двумерная задача для системы постоянных магнитов, создающая перекрестное магнитное поле (см. рис.1).

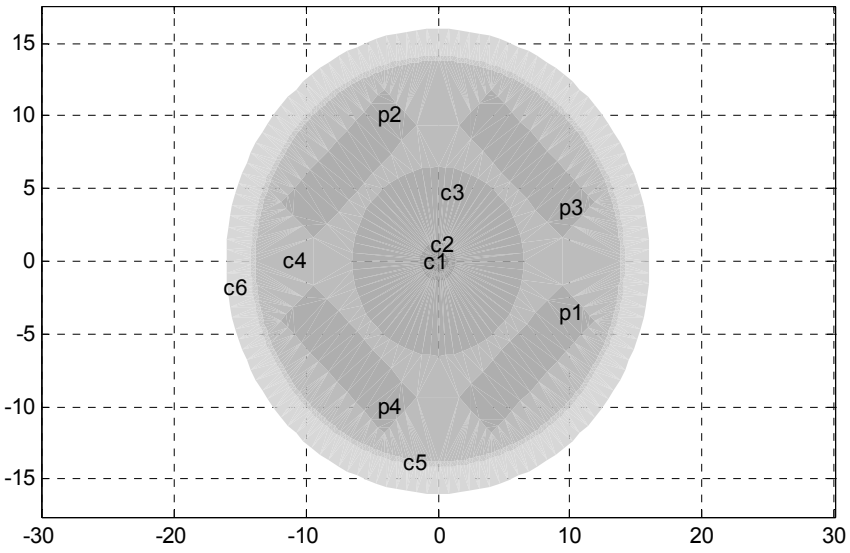


Рис. 1. Геометрия исследуемого электромеханического преобразователя.

На рисунке: c1 — вал; c2, c3 — внутренний и наружный радиус основного магнита; c4, c5 — внутренний и наружный радиус экрана (толщина экрана L=0.35); c6 — наружный радиус расчетной области; p1, p2 — первая обмотка; p3, p4 — вторая обмотка.

Проанализировав структуру в системе математического моделирования FEMLAB, получено распределение векторного магнитного потенциала и магнитного поля системы (см. рис.2).

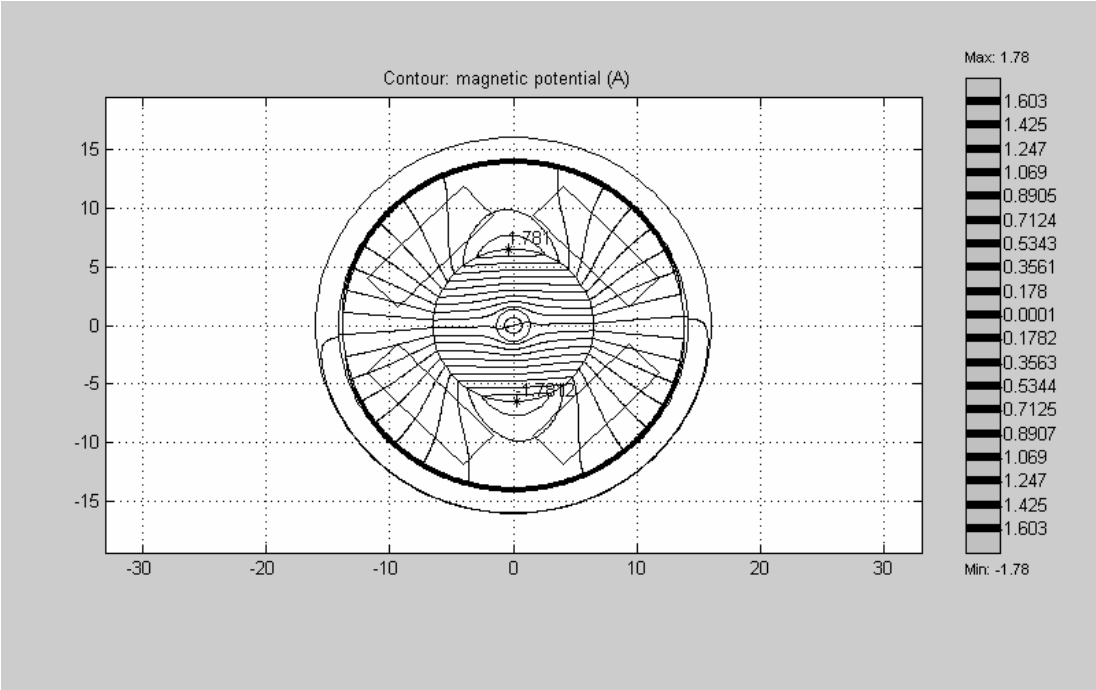


Рис. 2. Распределение векторного магнитного потенциала системы.

Основываясь на данных распределения векторного магнитного потенциала и магнитной индукции, построим рабочие (угловые) и силовые характеристики устройства, учитывающие его конструктивные особенности. Момент создаваемый магнитным полем оценим по формуле:

$$\overrightarrow{M_M} = \mu_0^{-1} \int_{V_T} (\vec{r_0} \times (\vec{B}\nabla)\vec{B} - 0{,}5 grad(B^2))dV =$$

$$= \mu_0^{-1} \left(\oint_{S_T} (\vec{r_0} \times \vec{B})(\vec{B}d\vec{S}) - \int_{V_T} (\vec{B}\nabla)\vec{r_0} \times \vec{B}dV - 0{,}5 \oint_{S_T} (B^2 \vec{r_0} \times d\vec{S}) \right). \quad (4)$$

Используя параметрический решатель при нахождении распределения магнитного поля, в зависимости от изменения значений токов обмоток, получим угловые характеристики момента вблизи положения равновесия (рабочая зона) при максимальном положительном токе в первой обмотке и нулевом токе во второй при номинальном значении остаточной магнитной индукции возвратного магнита (0,1 Тл, график справа) и при отклонении этого параметра от номинального вверх на 10% (0,11 Тл, график слева) (см. рис. 3), а также при отклонении этого параметра от номинального вниз на 10% (0,09 Тл, график справа) (см. рис. 4).

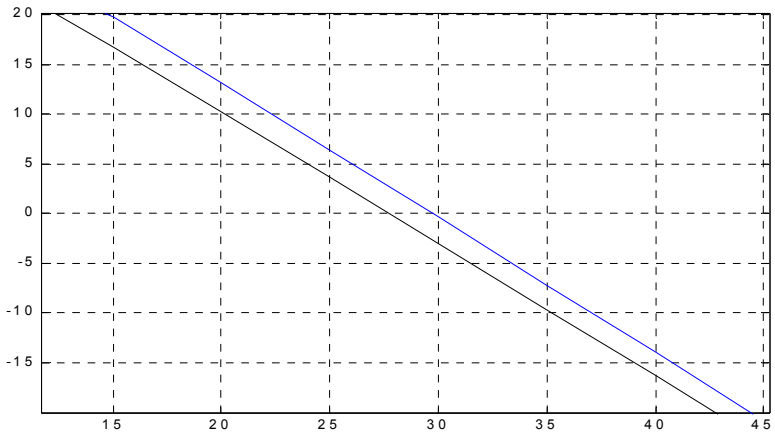


Рис. 3. Угловые характеристики момента вблизи положения равновесия, при отклонении вверх.

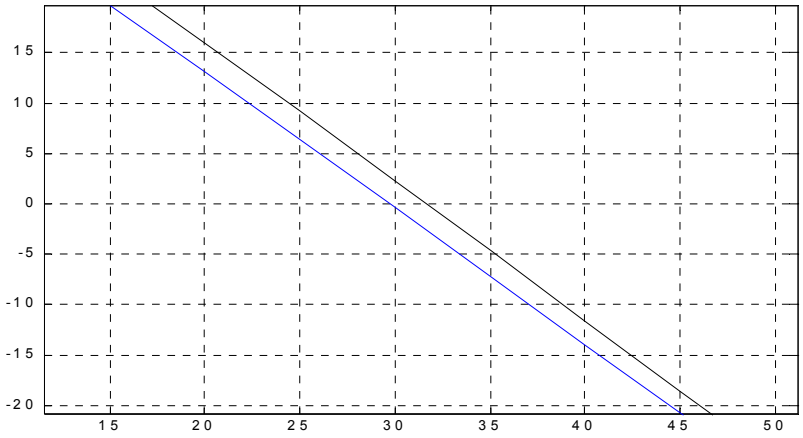


Рис. 4. Угловые характеристики момента вблизи положения равновесия, при отклонении вниз.

Как видно, отклонение положения стрелки (ошибка показания прибора) составляет примерно 2 градуса, в каждом из опытов. Разработанный вычислительный сценарии моделирует логометрический электромеханический преобразователь. Также он позволяет моделировать множество отклонения конструкции по исходному чертежу. Такие как: отклонения положения оси прибора и возвратного магнита, неравномерность распределения магнитной проницаемости и остаточной магнитной индукции в деталях и др. В развитии текущей двумерной модели, будет разрабатываться трехмерная, создающая полное физическое соответствие реальному устройству.

Литература

1. *Шмелев В. Е., Сбитнев С. А.* Теоретические основы электротехники.— Владимир: Владимирский гос. университет, 2003.— 88 с.
2. *Флетчер К.* Численные методы на основе метода Галеркина.— М.: Мир, 1988.— 352 с.

УДК 004

ОШИБКИ, ОБНАРУЖИВШИЕСЯ В ПАКЕТЕ MATLAB

Петров Ю. П.,
Санкт-Петербургский государственный университет; Санкт-Петербург,
Чертков К. Г.
East-Consept; Санкт-Петербург,
e-mail: Chertkov_kg@pisem.net

Высокая и заслуженная популярность программного пакета MATLAB, его широкая распространенность, заставляет с особым вниманием отнестись к тем ошибкам, с которыми неизбежно встретится его пользователь.

Неизбежные ошибки (которые можно исправить, но о способах их исправления пакет MATLAB [1], [2] ничего не говорит) встретимся прежде всего при:

1. Численном решении систем обыкновенных дифференциальных уравнений
2. При использовании многочисленных алгоритмов, включающих эквивалентные преобразования решаемых систем
3. Решении обобщенной задачи нахождения собственных значений.

Ошибки могут возникать и при решении других задач (поскольку все ошибки объединены, как увидим далее, одной общей причиной), но другие задачи будут рассмотрены позже, в отдельных статьях.

Д. ***Системы обыкновенных дифференциальных уравнений.***

Поскольку на практике коэффициенты дифференциальных уравнений получаются из опыта и измерения, и известны почти всегда лишь с ограниченной точностью, то результаты численного решения только тогда имеют практический смысл, когда неизбежным малым погрешностям коэффициентов соответствуют малые погрешности решений, а это будет тогда, когда для любого конечного $t = t_0$ решение $x(t_0)$ зависит от коэффициентов непрерывно.

Пакет MATLAB для решения систем дифференциальных уравнений рекомендует:

1. Сперва преобразуй систему к нормальной форме Коши — т. е. к системе n уравнений первого порядка, к виду

$$\begin{aligned} \dot{x}_1 &= f_1(x_1,\dots x_n), \\ &\dots\dots\dots\dots\dots\dots, \\ \dot{x}_n &= f_n(x_1,\dots x_n). \end{aligned} \qquad (1)$$

Преобразование должно выполняться, разумеется, с помощью эквивалентных преобразований, т. е. преобразований, при которых решения преобразованной системы (1) совпадают тождественно с решениями исходной системы. Правила эквивалентных преобразований хорошо известны.

2. Затем решай численно преобразованную систему (1)

Поскольку для системы n дифференциальных уравнений первого порядка при выполнении условий Липшица для функций $f_1,\dots f_n$ — а на практике условия Липшица почти всегда выполняются — решения зависят от коэффициентов непрерывно, то пакет MATLAB по умолчанию предполагает, что и в исходной системе решения будут зависеть от коэффициентов непрерывно, и поэтому полученное решение имеет практический смысл. На самом деле, как это было продемонстрировано в работе [3] это совсем не так и именно это обстоятельство для ряда систем уравнений (но конечно не для всех!) ведет к неизбежным ошибкам.

Пример

Рассмотрим систему дифференциальных уравнений с двумя переменными x_1 и x_2 (где $D = d/dt$):

$$\begin{aligned} (D^3 + 4D^2 + 5D + 2)x_1 - (D^2 + 2D + 1)x_2 = 0, \\ (D^2 + 4D + 5)x_1 - (D + 1)x_2 = 0. \end{aligned} \qquad (2)$$

Если исключить переменную x_2, то относительно x_1 придем к уравнению третьего порядка

$$(D^3 + 5D^2 + 7D + 3)x_1 = 0. \qquad (3)$$

Характеристический полином системы (2) равен

$$\Delta = \lambda^3 + 5\lambda^2 + 7\lambda + 3 \qquad (4)$$

и имеет корни $\lambda_1 = -3$; $\lambda_2 = \lambda_3 = -1$. Общее решение системы (2) имеет вид

$$x_1(t) = C_1 e^{-3t} + (C_2 t + C_3)e^{-t}. \qquad (5)$$

Систему (2) обычными традиционными эквивалентными преобразованиями нетрудно привести к нормальной форме Коши, которая в данном случае имеет вид

$$\begin{cases} \dot{x}_1 = -3x_1 - x_2 - x_3; \\ \dot{x}_2 = x_3; \\ \dot{x}_3 = -x_2 - 2x_3. \end{cases} \qquad (6)$$

Нетрудно проверить, что система (6) имеет тот же самый характеристический полином (4) и то же самое общее решение (5) что и исходная система (2). Преобразование системы (2) в систему (6) — безусловно эквивалентное преобразование. Поскольку все решения системы (6) зависят от всех ее коэффициентов непрерывно, то пользователи «по умолчанию» уверены в том, что и у исходной системы (2) решения зависят от всех ее коэффициентов непрерывно. На самом деле этого нет — и это приведет пользователя к неизбежной ошибке. Пользователь будет считать, что полученное им решение имеет практический смысл, а на самом деле оно практического смысла иметь не будет, поскольку сколь угодно малые погрешности в некоторых коэффициентах системы (2) изменят ее решение коренным образом.

Действительно, рассмотрим кроме системы (2) близкую к ней систему

$$\begin{gathered}(D^3+4D^2+5D+2)x_1-(D^2+2D+1)x_2=0,\\(D^2+4D+5)x_1-(0,999D+1)x_2=0,\end{gathered} \tag{7}$$

т. е. всего один коэффициент из системы (2) изменился на одну тысячную долю первоначальной величины. Характеристический полином системы (7) имеет вид

$$\Delta=-0,001\lambda^3+0,996\lambda^3+4,995\lambda^2+7\lambda+3 \tag{8}$$

и имеет помимо трех корней λ_1, λ_2, λ_3, очень близких к корням характеристического полинома (4) системы (2) большой положительный корень $\lambda_4=1001$. Поэтому в общем решении системы (7) будет присутствовать чрезвычайно быстро возрастающий член вида Ce^{1001t} и поэтому решения системы (7) будут очень сильно отличаться от решений системы (2).

Нетрудно проверить, что и у системы

$$\begin{gathered}(D^3+4D^2+5D+2)x_1-(D^2+2D+1)x_2=0,\\(D^2+4D+5)x_1-((1-\varepsilon)D+1)x_2=0,\end{gathered} \tag{9}$$

где ε сколь угодго малое положительной число — в общем решении появится очень быстро возрастающий член, который для малых ε имеет вид

$$C(\exp(t/\varepsilon)), \tag{10}$$

где C — постоянная интегрирования.

Таким образом, сколь угодно малая, неизбежная на практике погрешность в коэффициентах исходной системы уравнений (2) может привести к коренному изменению его решений. Связано это с тем, что у системы (2) как и многих других систем дифференциальных уравнений (примеры приведены в [3], [4]) — отсутствует непрерывная зависимость решений от коэффициентов или от входящих в них параметров.

А у преобразованной системы все обстоит совсем по-другому - ее решения зависят от всех ее коэффициентов непрерывно и поэтому доста-

точно малые изменения коэффициентов не приведут к заметным изменениям решений. Это следует из общей теоремы о непрерывной зависимости решений систем дифференциальных уравнения, записанных в нормальной форме Коши, от коэффициентов и параметров (поскольку система (6) безусловно удовлетворяет условию Липшица), но может быть проверенно и непосредственно. Нетрудно проверить, что решения системы (6) зависят от всех ее коэффициентов непрерывно (а у системы (2) этого нет).

Отсюда и возникает ошибка пользователя пакета MATLAB: он преобразует систему (2) (или другую подобную ей по свойствам) к нормальной форме, к виду (6), получит ее решение (так например при начальных условиях $x_1(0)=1,\ x_2(0)=0,\ x_3(0)=0$ это будет решение $x_1=e^{-3t}$) и будет уверен, что это решение с учетом неизбежных малых погрешностей значении коэффициентов исходной системы имеет практический смысл. На самом деле это решение (как и любое другое решение) практического смысла не имеет. Попытки использовать его приведут к неизбежным ошибкам в результате расчета, а это может стать (как показано в [3]) причиной аварий и даже катастроф.

В избежание недоразумений подчеркнем, что в этих ошибках пользователя нет никакой вины составителей пакета MATLAB. Составители опирались на математические знания своего времени и не могли учитывать совсем недавние, и во многом неожиданные научные результаты, опубликованные например в [3], [4], где было показано что эквивалентные преобразования (и в частности преобразования систем дифференциальных уравнений к нормальной форме), не меняющее самих решений как таковых, в то же время могут изменять некоторые важные свойства решений (и в частности непрерывную зависимость решений от коэффициентов и параметров).

Для предотвращения ошибок необходимо дополнить пакет MATLAB предостережением пользователю о возможных ошибках и дополнить пакет программой устранения ошибок по методам приведенным в [3], ввести например, программу использующую правило матриц степеней [3, с.78–84].

III. ***Вычислительные алгоритмы, использующие цепочки эквивалентных преобразований.***

Во многих алгоритмах используются цепочки эквивалентных преобразований. Так, наиболее распространенным способом вычисления определителей высокого порядка является способ последовательного сведения путем эквивалентных преобразований к определителям меньшего порядка. Решение систем, состоящих из n алгебраических уравнений с n неизвестными методом Гауса сводится к преобразованию исходной системы в системы меньшего порядка, с меньшим числом неизвестных и т. п.

В совсем недавние счастливые времена, когда верили, что эквивалентные преобразования ничего не меняют, считалось достаточным проверить эквивалентность используемых преобразований. В работах [3, 4, 5] было показано, что преобразования (в том числе и используемые в пакете MATLAB), эквивалентные в классичесом смысле, могут изменять корректность решаемой задачи и тогда неизбежные сколь-угодно малые погрешности округления могут сразу, за один шаг преобразования, привести к грубой ошибке в решении.

В статье [7] приведен пример численного решения обобщенной алгебраической проблемы собственных значений, т. е. нахождения значений параметра λ при которых существуют не нулевые решения системы линейных уравнений вида

$$(\lambda B - A)x = 0, \tag{11}$$

где A и B квадратные матрицы $n \times n$ матрицы. В этой статье рассмотрен пример , в котором n=5, а B — квазиединичная матрица, в которой на главной диагонали стоят 4 единицы и один ноль, а остальные элементы нули. Задача решалась методом последовательного исключения переменных x_1, x_2, x_3 и т. д. путем эквивалентных преобразований. При этом получалось лишнее собственное число целиком зависящее от погрешностей округления и происходящее из-за того, что одно из преобразований было эквивалентным в классическом смысле, но не в расширенном — в согласии с явлением предсказанным ранее в [3, с.74–75]. Традиционная методика ослабления влияния погрешностей округления путем перехода к вычислениям с удвоенной точность в данном случае не привела к правильному решению.

Поскольку описываемое явление, как было показано в [3, с.77], не возникает в классической задаче вычисления собственных значений, то ошибку пользователя можно устранить, если пакет MATLAB дополнить рекомендацией: обобщенную проблему собственных значений сводить к классической (как это опытные вычислители и делали) используя уравнения не содержащие λ для сокращения числа переменных и уравнений, как это показано в [3, с.77].

Заключение

Пользователь пакета MATLAB может получить ошибочные решения ряда практических задач. Эти ошибки не могут быть поставлены в вину разработчикам пакета. Они связанны с новыми, относительно недавними научными открытиями [3, 4, 5], обнаружившими, что привычное использование эквивалентных преобразований может приводить к изменению корректности решаемой задачи и тем самым являться источником грубых ошибок в расчетах, которые могуь повлечь за собой аварии и даже катастрофы. Для предотвращения этих ошибок можно использовать методы опи-

санные в [3, 4, 5]. Их можно ввести в пакет MATLAB как необходимое дополнение.

Литература

1. *Андриевский Б. Р., Фрадков А. Л.* Элементы математического моделирования в средах MATLAB 5 и SciLab.— СПб.: Наука, 2001.— 386 с.
2. *Конев В. Ю., Мироновский Л. А.* Основные функции пакета MATLAB.— СПб, 1994.— 75 с.
3. *Петров Ю. П. Петров Л. Ю.* Неожиданное в математике и его связь с авариями и катастрофами последних лет.— СПб: Изд-во СПбГУ.— 1-е изд., 1999.— 108 с.; 3-е изд.— 2002.—141 с.
4. *Петров Ю. П.* Устойчивость линейных систем управления при вариациях параметров // Автоматика и телемеханика.— 1994.— №11.— С.186–189.
5. *Петров Ю. П.* Новые главы теории управления.— СПб: Изд-во СПбГУ, 2000.— 156 с.
6. *Данилевич Я. Б., Петров Ю. П.* О необходимости расширения понятия эквивалентности математических моделей // Доклады РАН.— 2000.— Т.371.— №4.— С.473–475.
7. *Чертков К. Г.* Исследование чувствительности к погрешностям округления собственных значений линейных систем // Известия ТГУ. Серия: Проблемы управления электротехническими объектами.— 2002.— С.138–139.

УДК 004

ОБ ОШИБКАХ ПАКЕТА MATLAB

Петров Ю. П.,
Санкт-Петербургский государственный университет; Санкт-Петербург,
Шароватов В. Т.
Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова, Санкт-Петербург,
e-mail: rivas@rivas.spb.su

Широкая распространенность пакета прикладных программ (ППП) MATLAB заставляет с особым вниманием отнестись к ошибкам, недавно обнаруженным в этом пакете. Об этих ошибках необходимо срочно предостеречь пользователей и принять все меры к их устранению, иначе они могут стать причиной неверных результатов, а значит причиной возможных аварий и катастроф.

Ошибки обнаружены в следующих программах ППП MATLAB:

1. Численного решения систем обыкновенных дифференциальных уравнений;
2. Расчета и проверки устойчивости и запасов устойчивости технических объектов и систем управления;
3. Решения обобщенной задачи о собственных значениях;
4. Решения интегральных уравнений.

Выполним краткий разбор этих ошибок.

1. Решение дифференциальных уравнений.

Необходимой предпосылкой достоверности результатов расчета является, как известно, наличие непрерывной зависимости решений от коэффициентов и параметров, поскольку коэффициенты математических моделей реальных объектов определяются из опыта или измерения и известны всегда с ограниченной точностью. Для систем, представленных в нормальной форме, т. е. систем, состоящих из n уравнений первого порядка, непрерывная зависимость решений от коэффициентов и параметров доказана [1].Поскольку почти всякую систему можно привести к нормальной форме с помощью несложных эквивалентных преобразований, то еще недавно считалось очевидным, что та же непрерывная зависимость имеет место для любых систем. Однако это далеко не так. Недавно было обнаружено [2], что существуют «особые» системы дифференциальных уравнений (и соответствующие этим системам технические объекты), не имеющие непрерывной зависимости решений от параметров. После приведения таких систем к нормальной форме будет казаться, что непрерывная зависимость появилась, но в реальном объекте она отсутствует.

Поскольку ППП MATLAB осуществляет численное решение систем дифференциальных уравнений путем предварительного приведения их к нормальной форме, то это означает: при каждой встрече с «особой» системой ППП MATLAB выдаст неверный результат решения, что станет возможной причиной аварии или катастрофы.

«Особые» системы встречаются не очень часто, поэтому и аварии, порожденные ошибками расчета, не являются массовыми и их до последнего времени внимательно не исследовали. Однако ряд катастроф, происшедших с пассажирскими самолетами на участках полета, когда самолет летел под управлением автопилота, по-видимому был вызван наличием таких систем. Мириться с авариями, приводящими к гибели людей, нельзя, поэтому ППП MATLAB необходимо обязательно и срочно пополнить дополнительными программами, выявляющими «особые» системы и обеспечивающими достоверность результатов расчета при встрече с ними. Теоретические основы подобных программ изложены в [2] и в [3], и наиболее подробно в [4]. Работы по их реализации проводятся авторами доклада.

1. Пример

Математической модели системы управления электроприводом с квадратичным спектром возмущающих воздействий является, как показано в [2] система дифференциальных уравнений:

$$\left(D^3+4D^2+5D+2\right)x_1-\left(D^2+2D+1\right)x_2=0; \tag{1}$$

$$\left(D^2+4D+5\right)x_1-\left[D+1\right]x_2=0, \tag{2}$$

где x_1 — частота вращения электродвигателя, x_2 -управляющее воздействие, D — оператор дифференцирования.

Характеристический полином системы (1)–(2) равен

$$\Delta=\lambda^3+5\lambda^2+7\lambda+3 \tag{3}$$

и имеет корни $\lambda_1=-3;\lambda_2=\lambda_3=-1$. Общее решение системы (1)–(2) предстает в виде

$$x_1(t)=c_1e^{-3t}+\left(c_2t+c_3\right)e^{-t} \tag{4}$$

и является устойчивым.

Систему (1)–(2) эквивалентными преобразованиями можно привести к нормальной форме:

$$\begin{aligned}\dot{x}_1&=-3x_1-x_2-x_3\\ \dot{x}_2&=x_3\\ \dot{x}_3&=-x_2-2x_3\end{aligned} \tag{5}$$

которая, как нетрудно проверить, имеет тот же характеристический полином (3), что подтверждает эквивалентность преобразования системы (1) — (2) в систему (3).

Но система (1)–(2) является «особой» и не имеет непрерывной зависимости решений от коэффициентов и параметров.

Нетрудно проверить, что у системы

$$\left(D^3+4D^2+5D+2\right)x_1-\left(D^2+2D+1\right)x_2=0; \tag{6}$$

$$\left(D^2+4D+5\right)x_1-\left[(1-\varepsilon)D+1\right]x_2=0, \tag{7}$$

где ε — сколь угодно малое положительное число, в общем решении появиться очень быстро возрастающий экспоненциальный член, который для малых ε имеет вид:

$$C_4\exp\left(\frac{t}{\varepsilon}\right). \tag{8}$$

Пример: если $\varepsilon=0{,}001$, то есть один из коэффициентов уравнения (2) изменился всего на одну тысячную долю первоначального значения, то характеристический полином (3) изменится и станет равным:

$$\Delta=-0{,}001\lambda^4+0{,}996\lambda^3+4{,}995\lambda^2+7\lambda+3 \tag{9}$$

и помимо трех корней, близких к корням $\lambda_1=-3;\lambda_2=\lambda_3=-1$, полином (3) будет иметь большой положительный корень $\lambda_4=1001$, т. е. в решениях системы (1) — (2) появится стремительно растущий член $C4e^{+1001t}$.

Если же проверка поведения систем будет проводиться пользователем ППП MATLAB, как рекомендует пакет, после приведения системы (1)–(2) к нормальной форме (5), то он ничего этого не увидит.

Таким образом, реальный объект, математической моделью которого является система (1)–(2) является крайне опасным, он может коренным образом изменить свои свойства при сколь угодно малых, неизбежных на практике, вариациях своих параметров, а ППП MATLAB, об этом ничего не скажет. Результатом окажется ошибочное решение, т. е. коренное расхождение между результатом расчета и реальным поведением рассчитываемого объекта, а это может привести к аварии или катастрофе.

Для исправления опасных ошибок ППП MATLAB он должен быть дополнен специальными программами, о которых уже говорилось.

2. Ошибки в программах расчета устойчивости ППП MATLAB

Наличие этих ошибок вытекает из уже изложенного материала: если коренное расхождение между результатом расчета, использующего ППП MATLAB и реального, возникает, как показывает пример с системой (1)–(2), даже при конечных *t*, то не меньшее расхождение будет и при расчетах устойчивости , предполагающих $t\to\infty$.

Система управления, описываемая уравнениями (1)–(2) имеет нулевой запас устойчивости и очень опасна, а ее исследование с помощью программ ППП MATLAB даст совершенно ошибочный ответ о запасах устой-

чивости и послужит основой для неверного технического решения, ведущего к аварии.

Ошибки ППП MATLAB в рассматриваемом случае могут быть исправлены теми же дополнительными программами, что и в пункте 1.

3.Ошибки ППП MATLAB при решении обобщенной задаче о собственных значениях.

4. Ошибки ППП MATLAB при решении интегральных уравнений.

Они имеют все ту же причину — не учет возможного изменения корректности решаемой задачи при эквивалентных преобразованиях, используемых при ее решении. Теоретические основы методов распознавания возможных ошибок и избежания их изложены в работах [3], [4]. Работа по программному обеспечению этих методов проводится авторами.

Заключение

Для довольно широкого, пока еще не до конца определенного круга практических задач, использование программ ППП MATLAB приводит к ошибкам расчета с возможными последствиями в виде аварий рассчитываемых объектов. Для предупреждения ошибок ППП MATLAB должен быть дополнен специальными программами, работа над которыми проводится авторами. Методика выделения особых систем, при встрече с которыми возможны ошибки, изложена в недавно вышедшей книге [4].

Литература

1. *Матвеев Н. М.* Обыкновенные дифференциальные уравнения.— СПб.: Специальная литература, 1996.— 371 с.
2. *Петров Ю. П., Петров Л. Ю.* Неожиданное в математике и его связь с авариями и катастрофами последних лет.— СПб.: СпбГУ, 2002.— 141 с.
3. *Петров Ю. П., Сизиков В. С.* Корректные; некорректные и промежуточные задачи с приложениями.— СПб.: Изд-во «Политехника», 2003.
4. *Петров Ю. П.* Новые главы теории управления и компьютерных вычислений.— СПб.: Изд-во «БВХ», 2004.— 194 с.

УДК 004

СОРТИРОВКИ ПО АЛЬТЕРНАТИВНЫМ ПРИЗНАКАМ СРЕДСТВАМИ MATLAB

Петрова К. Ю.
Университет аэрокосмического приборостроения, Санкт-Петербург,
e-mail:xen@excite.com

Введение

В пакете MATLAB имеется ряд функций связанных с сортировкой. Однако стандартная функциональность имеет ряд неприятных ограничений. Некоторые из идей, уже ставших стандартом при программировании на C++ и воплощенных в stdlib и STL, могут быть применены при разработке тулбокса, реализующего сортировки по альтернативным признакам. В работе делается попытка сохранить основные достоинства и избежать некоторых недостатков стандартных сортировок.

1. Функции сортировки в пакете MATLAB

В пакете MATLAB на данный момент реализованы следующие функции, связанные с сортировкой — **sort** и **sortrows**. Они обладают следующими удобными свойствами:

- возможность сортировать многомерный массив по заданной размерности;
- возможность сохранить перестановку индексов, произведенную при сортировке.

В ряде случаев этих возможностей достаточно и для сортировки по альтернативным признакам. Например, рассортируем массив по остатку по модулю 7:

```
% инициализация
x=floor(rand(1,10)*100);
% запомнили перестановку индексов в массиве i
[s,i]=sort(rem(x,7));
% получили отсортированный массив
y=x(i);
% удостоверились в правильности сортировки
rem(y,7)
```

Такой подход можно применить, если известно отображение в пространство вещественных чисел, сохраняющее порядок, подобно тому, как это было показано на примере.

Тем не менее, стандартные функции обладают некоторыми недостатками:

- невозможность сортировать по различным критериям;
- невозможность сортировать массивы ячеек с численным (а не строковым) содержимым;
- невозможность сортировать массивы объектов классов, отличных от численных (double, int8, и т. д.).

Второй и третий недостатки очевидны, первый же требует небольшого комментария. Например, при сортировке массива комплексных чисел происходит по модулю, а затем по углу, однако может возникнуть необходимость сортировки по вещественной части, а затем по мнимой.

Для устранения этих недостатков можно попытаться адаптировать идеологию, используемую при программировании на C++.

2. Популярные алгоритмы сортировок

Существует два основных класса сортировок — распределяющая сортировка, основанная на том, что число возможных значений ключа различно, и сортировка сравнениями, использующая прямое сравнение значений ключа. Понятно, что для реализации сортировки объектов произвольного класса используется второй подход. Основной критерий сравнения различных алгоритмов — оценка времени исполнения. Для сортировок сравнениями давно доказана теорема о максимальном быстродействии $O(n \log n)$, а для сортировок распределением — $O(n)$. Используется также такой показатель качества, как естественность поведения. При естественном поведении частично отсортированный массив будет досортирован быстрее, чем не отсортированный.

Ниже приведен список наиболее простых и популярных алгоритмов сортировки сравнениями.

1. Простые вставки.
2. Бинарные вставки.
3. Двухпутевые вставки.
4. Метод Шелла.
5. Пузырьковая сортировка.
6. Модифицированная пузырьковая сортировка.
7. Быстрая сортировка.
9. Сортировка выбором.
10. Пирамидальная сортировка.
11. Сортировка слиянием.

На сегодняшний день самыми эффективными методами сортировки считаются быстрая сортировка (среднее время выполнения — $O(n \log n)$, и максимальное — $O(n^2)$), распределяющая сортировка и быстрая сортировка с составными ключами. Для прикладных задач, использующих элемен-

ты сортировки также очень полезна пирамидальная сортировка (среднее время выполнения — $O(n \log n)$).

Наиболее полную информацию по вопросу можно получить в [1]. В упрощенном и сжатом виде описание основных алгоритмов можно найти в сети, например, в [2]–[4].

3. Сортировка в C++

Рассмотрим стандартные функции сортировки, используемые C++. Наиболее традиционным способом является вызов функции qsort (stdlib.h), которая в качестве первых трех аргументов принимает адрес и размер массива, а также размер элемента массива, а в качестве четвертого аргумента — адрес функции, осуществляющей сравнение (MS VC++ 6.0):

void qsort(void *base, size_t num, size_t width, int (__cdecl *compare)(const void *elem1, const void *elem2));

Более продвинутые механизмы работы с данными предлагаются в библиотеке STL [5]. В ней описывается три основных класса объектов — контейнерные классы, алгоритмы и итераторы. Механизм шаблонов позволяет реализовать структуры данных (массивы, списки, очереди, хеш-таблицы и т. п.) и алгоритмы работы с ними (последовательный доступ, случайный доступ, поиск элемента, перестановка, сортировка) независимо от типа объектов, организованных в структуру. На примере сортировки это означает, что для сортировки массива целых чисел используется тот же код, что и для сортировки массива записей с заданной операцией сравнения. Очевидно, что это существенно упрощает программирование. На данный момент в STL реализованы следующие шаблоны алгоритмов, связанных с сортировкой:

- **sort**
- **stable_sort**
- **partial_sort**
- **partial_sort_copy**
- **is_sorted**

В более ранних версиях STL в **sort** использовался алгоритм быстрой сортировки, но сейчас используется т.н. «интроспективная сортировка», являющаяся разновидностью быстрой сортировки. Устойчивая сортировка (**stable_sort**) иногда полезна при сортировке массива записей, имеющих несколько полей. Для этого применен алгоритм сортировки слиянием. В шаблоне **partial_sort** используется пирамидальная сортировка. Назначение шаблона функции **is_sorted** явствует из названия. Параметры всех шаблонов сортировок выглядят примерно одинаково:

template <class RandomAccessIterator> void sort(RandomAccessIterator first, RandomAccessIterator last);

```
template <class RandomAccessIterator, class StrictWeakOrdering> void
sort(RandomAccessIterator first, RandomAccessIterator last, StrictWeakOrdering
comp);
```

Ниже приведен пример вызова алгоритма устойчивой сортировки из стандартного теста, поставляемого вместе с STL:

```
using namespace std;
int stblsrt1_test(int, char**)
{
 cout<<«Results of stblsrt1_test:»<<endl;
 int array[6] = { 1, 50, -10, 11, 42, 19 };

 stable_sort(array, array + 6);
 for(int i = 0; i < 6; i++)
  cout << array[i] << ' ';
 cout << endl;
 return 0;
}
```

Помимо этого, в STL введено понятие «упорядоченных контейнеров», т. е. имеется возможность объявить контейнер «отсортированным» и в дальнейшем все операции над ним производить, не нарушая порядка.

В MFC также используется механизм контейнеров, реализованный в виде шаблонов (CArray, CMap, CList), однако алгоритмы типа сортировки не реализованы.

4. Альтернативная реализация сортировок в MATLAB

Представляется целесообразным разработка тулбокса, содержащего основные алгоритмы сортировки и удовлетворяющего следующим требованиям:

1. Возможность сортировать по различным критериям.
2. Возможность сортировать как простые массивы, так и массивы ячеек.
3. Возможность сортировать массивы объектов классов, отличных от numeric (double, int8, и т. д.).
4. Возможность сортировать многомерный массив по заданной размерности.
5. Возможность сохранить перестановку индексов, произведенную при сортировке.
6. Возможность компиляции функций сортировки в mex-файлы и использования их при создании независимых приложений.

К выполнению пунктов 1–3 можно подойти двумя способами — описывая объект, для которого определена операция сравнения, или использую команду eval или ее разновидности.

Пункты 4 и 5 предполагают разбор входных и выходных параметров функции сортировки. Это можно сделать, используя функции varargin,

varargout, nargin и nargout, позволяющие описывать функцию с меняющимся количеством аргументов:

```
function varargout=mysort(varargin)
% function varargout=mysort(varargin);
%       [y]=mysort(x,sort_function);
%       [y]=mysort(x,sort_function,sort_method);
%       [y,i]=mysort(x,sort_function);
%       [y,i]=mysort(x,sort_function,sort_method);
%       [y]=mysort(x,dimension,sort_function);
%       [y]=mysort(x,dimension,sort_function,sort_method);
%       [y,i]=mysort(x,dimension,sort_function);
%       [y,i]=mysort(x,dimension,sort_function,sort_method);
```

Пункт 6 накладывает определенные ограничения на реализацию функции. Так, при использовании компилятора версии 3.0 (имеется в виду версия тулбокса MATLAB Compiler, а не компилятора C++) нельзя использовать объекты, а также функцию eval со строковым аргументом, содержащим имена переменных из рабочего пространства. В более ранних версиях функцию eval вообще нельзя использовать. В дальнейшем будем предполагать этот пункт наименее существенным, и обсудим способ реализации, который заведомо ему не удовлетворяет.

Вне зависимости от того, как именно реализовывать функцию сортировки, алгоритм распадается на следующие подзадачи:

- Анализ входных и выходных аргументов
- Выбор нужного измерения
- Сортировка одномерного массива

Наиболее важным в этом списке является последний пункт, на котором следует остановиться подробнее.

5. Реализация сортировок в MATLAB с использованием объектов

Наиболее правильным с точки зрения современных представлений об организации программного обеспечения видится использование концепции контейнеров, содержащих объекты, поддерживающие операцию сравнения. Тем не менее, переносить идеи, реализованные в STL, непосредственно в среду MATLAB невозможно по причине отсутствия механизма шаблонов. Помимо этого в MATLAB довольно специфическая структура классов (рис. 1), которая предъявляет определенные требование к реализации сортируемых объектов. Все объекты, в том числе и пользовательские, являются потомками класса ARRAY, что делает принципиально невозможным разделение понятий контейнера и содержащихся в нем объектов.

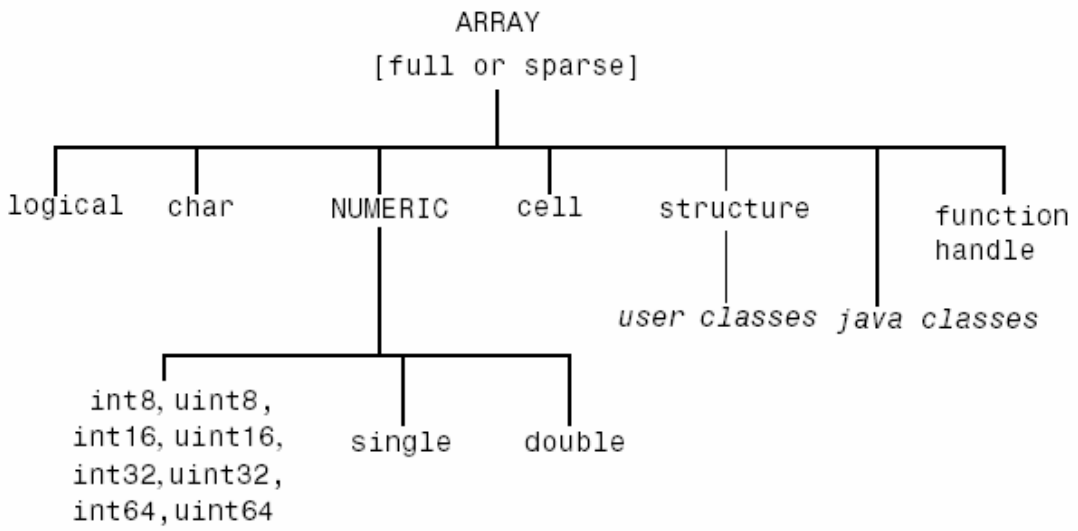


Рис. 1. Дерево классов MATLAB [6].

Покажем основные шаги, необходимые при реализации массива объектов с сортировкой на примере комплексных чисел с сортировкой по вещественной части, а затем по мнимой.

Методы класса должны располагаться в отдельном каталоге под названием @name, где name — имя класса. Конструктор класса описывается при этом в файле name.m. Защищенные методы объекта при этом располагаются в подкаталоге private. Так, методы создаваемого класса будут находиться в каталоге @complex2, а конструктор complex2. Для того, чтобы обеспечить правильное взаимодействие с системой, в конструкторе необходимо обработать следующие три ситуации:

- Вызов без параметров и инициализация объекта данными по умолчанию
- Вызов с параметром того же класса и создание нового объекта копированием
- Вызов с явным указанием полей данных

Проверки типов осуществляются при помощи команды isa, например:

isa(n,'double')
isa(c,'complex2')

Как следует из рис. 1, любой пользовательский класс является потомком класса STRUCTURE, поэтому создание нового объекта следует начинать с описания его как структуры, например:

```
c = struct('re',[],'im',[]);
```

Выходным параметров конструктора является вновь созданный объект. Внутри конструктора следует явно описать принадлежность нового объекта своему классу:

```
c = class(c,'complex2');
```

При необходимости можно определить иерархию классов — т. е. перечислить классы-предки для данного класса. Это делается при помощи функции inferiorto.

Помимо конструктора, для нормального использования класса необходимо реализовать методы set и get, осуществляющие чтение/запись полей (в MATLAB все поля класса считаются частными, т. е. доступ к ним может осуществляться только через методы), метод display, осуществляющий вывод объекта на экран, а также методы, связанные с индексацией:

- Subsref
- Subsasgn
- End
- Length
- Size
- Subsindex

Первые два метода используются для организации индексации самого объекта, третий, четвертый и пятый — вспомогательные методы используемые при индексации, и их назначение такое же, как и у одноименных функций над числовыми массивами. Шестой же метод предназначен для использования объекта в качестве индекса некоторого ассоциативного массива. Этот метод для решения задачи сортировки можно не реализовывать. Рассмотрим требования к реализации subsref и subsindex:

```
function val =subsref(c,index)
function c =  subsasgn(c,index,val)
```

Здесь c — объект класса, val — старое или новое значение поля, элемента массива или группы элементов, a index — структура с полями type и subs. Поле type может принимать значения '.', '()' или '{}' в зависимости от типа используемой индексации. Поле subs содержит значение индекса в зависимости от выбранного способа индексации. В нашем случай для реализации работающего примера достаточно обработать типы '.' и '()'. Ниже приведен текст функции subsref для поддержки одномерной индексации класса complex2:

```
function c = subsasgn(c,index,val)

if index.type=='.'
 if index.subs=='re'
  c.re=val;
 end

 if index.subs=='im'
  c.im=val;
 end

 return;
end
```

```
if index.type=='()'
 s=index.subs{:};
 if isa(val,'complex2')
  c.re(s)=val.re;
  c.im(s)=val.im;
 else
  c.re(s)=real(val);
  c.im(s)=imag(val);
 end

end
```

Метод **subsasgn** реализуется аналогично.

Отдельных пояснений заслуживают перегруженные методы. Для примера сортировки комплексных чисел будет достаточно описать метод gt, вместо которого можно будет писать инфиксный оператор '>':

```
function l=gt(c1,c2)
if isa(c1,'complex2') & isa(c2,'complex2')
 l= (c1.re > c2.re) |  (c1.re== c2.re & c1.im > c2.im);
else
 if isa(c1,'double')
  cc1=complex2(c1);
 else
  cc1=c1;
 end
 if isa(c2,'double')
  cc2=complex2(c2);
 else
  cc2=c2;
 end
 l=cc1>cc2;
end
```

Для удобства также полезно описать методы horzcat и vertcat, что сделает возможным употребление следующей конструкции:

```
>> c1=complex2(sqrt(-8)+3)
3 + 2.828427e+000*i
>> c2=complex2([1,2])
1
2
>> c3=[c1, c2]
3 + 2.828427e+000*i
1
2
```

Также, хотя и необязательно, полезной бывает реализация методов-конвертеров, преобразующих объекты в другие типы. Пример такого метода, преобразующего complex2 в double, приведен ниже:

```
function d=double(c)
if isa(c,'complex2')
  d=c.re+c.im*sqrt(-1);
else
```

```
  d=c;
end
```

После того, как описаны все структуры массива или массива ячеек со сравнением, для сортировки можно использовать функцию, не содержащую явного указания на тип сортируемого объекта. Ниже приведен пример функции пузырьковой сортировки одномерного массива:

```
function y=BubbleSort(x)
% сортировка одномерного вектора
% метод пузырька
b=length(x);
y=x;
while b>1
 t=0;
 for j=1:b-1
   if y(j)>y(j+1)
     tmp=y(j);
     y(j)=y(j+1);
     y(j+1)=tmp;
     t=j;
   end
 end
 b=t;
end;
```

Эту функция можно применить как к массиву класса complex2, так и к массиву любого другого класса, поддерживающего индексацию типа '()' и операцию сравнения gt:

```
>> x=complex2(roots([1 1 1 1]))
-1.000000e+000
-1.665335e-016 + 1.000000e+000*i
-1.665335e-016 - 1.000000e+000*i
>> BubbleSort(x)
-1.000000e+000
-1.665335e-016 - 1.000000e+000*i
-1.665335e-016 + 1.000000e+000*i
```

6. Реализация сортировок в MATLAB с использованием feval

Для того, чтобы сделать возможным генерацию mex-файлов на основе создаваемого алгоритма, следует отказаться от использования объектов. Удобным при этом представляется использование функции **feval**, которая осуществляет вызов заданной MATLAB-функции с заданными аргументами. В отличие от **eval**, эта функция не является препятствием для компилятора MATLAB. Покажем это на примере двух функций **test01** и **test02**:

```
function a=test01(b)
a=eval('b+b');

function a=test02(b)
a=feval('plus',b,b);
```

Ниже приведен протокол их выполнения в виде обычных файлов и mex-файлов, соответственно:

```
>> clear test02
>> !erase test02.dll
>> clear test01
>> !erase test01.dll
>> test10(8)

ans =

   16

>> test02(8)

ans =

   16

>> mcc -x test01
>> test01(8)
??? Undefined function 'b'.

Error in ==> C:\MATLAB6p5\work\test01.dll
>> mcc -x test02
>> test02(8)

ans =

   16
```

Таким образом, сортировку по альтернативному признаку можно реализовать с использованием «безопасной» для компилятора функции **feval**. Ниже приведен текст функции для сортировки одномерного массива методом простых вставок, принимающей в качестве второго аргумента имя функции сравнения:

```
function y=InsertSort(x,sort_func)
y=x;
for j=2:length(y)
 i=j-1;
 xx=y(j);
 while (i>0) & (feval(sort_func,xx,y(i)))
  y(i+1)=y(i);
  i=i-1;
 end
 y(i+1)=xx;
end
```

Опишем альтернативную функцию сравнения **rootsort**:

```
function y=rootsort(r1,r2)
% function y=rootsort(r1,r2)
```

```
% sort by real part, than by complex
y=0;
if real(r2)>real(r1)
   y=1;
elseif real(r1)>real(r2)
   y=0;
elseif imag(r2)>imag(r1)
   y=1;
end
```

Теперь воспользуемся этими функциями для сортировки массива комплексных чисел:

```
>> d=roots(ones(6,1))'

d =

  0.5000 - 0.8660i   0.5000 + 0.8660i  -1.0000          -0.5000 - 0.8660i  -0.5000 + 0.8660i

>> d1=InsertSort(d,'rootsort')

d1 =

  -1.0000          -0.5000 - 0.8660i  -0.5000 + 0.8660i   0.5000 - 0.8660i   0.5000 + 0.8660i
```

Литература

1. *Кнут Д.* Искусство программирования.— М.: Мир, 1978.
2. *Кукушкин Б. А.* Описания комбинаторных алгоритмов. Методические указания к выполнению лабораторных работ по курсу «Комбинаторные алгоритмы».— СПб.: Институт точной механики и оптики, 1995.
3. FAQ по сортировкам (http://faqs.org.ru).
4. *Niemann T.* Sorting and Searching: A Cookbook. (http://www.geocities.com/SoHo/2167/book.html).
5. Standard Template Library Programmer's Guide. (http://www.sgi.com).
6. MATLAB The Language of Technical Computing. Using MATLAB Version 6.— The MathWorks, Inc., 1984–2002.

УДК 519.6

ОПЕРАЦИИ НАД ПОЛИНОМАМИ НЕСКОЛЬКИХ ПЕРЕМЕННЫХ В СИСТЕМЕ MATLAB

Птичкин В. А.
Белорусский государственный университет информатики и радиоэлектроники, Минск, Белоруссия,
e-mail:ptichkinv@gw.bsuir.unibel.by

Операции над полиномами нескольких переменных по традиции относят к компьютерной математике (символьной алгебре). Несомненно, любая система символьной математики должна включать в себя действия над полиномами нескольких переменных. Все такие системы — мощные независимые программные продукты. Перенос результатов, полученных в них, в другую программную среду, используемую для вычислений, например, в C - задача довольно трудоемкая и требующая достаточно высокой квалификации. Поэтому полезно наделить какой-нибудь объектно-ориентированный язык высокого уровня способностью обрабатывать символьные выражения. Для языка C++ эта работа уже проделана [1]. Основное достоинство такого подхода к символьным вычислениям заключается в том, что такую систему легко расширить и приспособить к конкретной задаче, решаемой в среде C++ [1].

Язык MATLAB с недавних пор также является объектно-ориентированным языком. Однако задачи расширения системы MATLAB решаются иным способом — с помощью специального модуля Toolbox, в котором хранятся различные пакеты расширения системы. Многие из них содержат специальные средства для интеграции с другими программами. В частности, в систему встроено ядро одной из самых мощных систем символьной математики Maple V, которое используется пакетами расширения Symbolic Math Toolbox и Extended Symbolic Math Toolbox. Благодаря ним стали возможными символьные вычисления и в среде MATLAB [2].

На наш взгляд, весьма сомнительно, что такую систему легко расширить и приспособить к конкретной задаче, решаемой в среде MATLAB. То же самое можно сказать относительно интеграции упомянутого ядра с другими пакетами прикладных программ, такими как Control System Toolbox. Вполне возможно, что опытный пользователь решит проблему сопряжения двух пакетов, но одним из самых существенных преимуществ системы MATLAB является дружественность ее к пользователям разного уровня квалификации.

На наш взгляд, необходимость выполнения операций над полиномами возникает у пользователей самого разного уровня. В приложениях нередко полиномы используются только как форма приближенного пред-

ставления функций. Рассматривать полиномы в этих условиях как алгебраическую структуру, производить операции над ними без потери точности, на наш взгляд, чаще всего лишено практического смысла. Достаточно располагать только программной поддержкой рутинных, но очень трудоемких для ручного счета операций, чтобы получить практически полезные для многих приложений результаты.

Предлагать для этих целей использование пакетов символьной математики нам кажется не только расточительством, но и не способствующим их распространению для решения этого класса задач. Более того, ни в документации по пакету расширения Symbolic Math Toolbox, ни в одном из (известных автору) руководств [2], [3] или справочников [4] не приводятся примеры применения алгебраических операций к полиномам нескольких переменных. Казалось бы, именно с них должно начинаться знакомство с символьной математикой. Вместо этого в самых первых примерах используются функции дифференцирования, интегрирования, арифметики произвольной точности и функции линейной алгебры [2], [4]. На наш взгляд, это не случайно. По-видимому, ни создатели, ни популяризаторы упомянутого пакета не считали полиномы достаточно серьезной структурой символьной математики для практических приложений.

Сказанное не относится к роли этой структуры в математике, о чем свидетельствует специальная литература [5], а только к отношению разработчиков систем символьной математики к полиномам. Другими словами, мы имеем в виду не только разработчиков системы MATLAB, но и других систем, даже самой мощной из них — системы Maple. В подтверждение можно привести демонстрационный пример использования функций выделения коэффициентов полиномов (**coeff** и **collect**) системы Maple V R4/R5 к обработке двух достаточно простых полиномов [3]. После трехкратного применения функции **coeff** (с разными параметрами) к каждому из полиномов, а затем функции **collect** к каждому полиному система выдает сообщение об ошибке, которая «носит неустранимый характер и после первого появления начинает возникать даже в тех вычислениях, которые до этого проходили гладко» [3]. Более того, она характеризуется как часто возникающая и для устранения которой может потребоваться перегрузка системы.

Классы полиномов нескольких переменных

Полиномы одной переменной уже давно используются в системе MATLAB, хотя они и не оформлены в качестве основного или встроенного класса. В документации и учебных пособиях по программированию в системе MATLAB это делается только в качестве примера создания пользовательского класса [6]. Аналогичным образом не трудно создать и класс полиномов нескольких переменных mpol [7].

Из сказанного ранее следует, что целью создания данного класса не является решение задач символьной алгебры даже в рамках полиномиальных моделей, хотя некоторые методы этого класса по отдельности решают те же локальные задачи, что и методы символьной алгебры. Об одной из основных целей уже сказано — это обеспечение дружественного интерфейса для неискушенных в компьютерной математике пользователей при решении определенного рода задач, которые уже решались с использованием полиномиальных или параметрических представлений до появления компьютерной математики.

Характерным примером такого рода задач являются задачи классической теории регулирования и управления, где описание объекта исследования представляется в виде передаточной функции — отношения двух полиномов переменной *s* или *z*. Числитель и знаменатель передаточной функции можно считать полиномами одной переменной только при фиксированных значениях всех параметров. Оставляя свободными *n* параметров передаточной функции, следует рассматривать ее как отношение полиномов *n* + 1 переменной. При небольшом числе свободных параметров и невысоком порядке передаточных функций возникает вполне естественное желание оперировать передаточными функциями в параметрической форме для последующего выбора численных значений параметров при параметрическом синтезе системы. Выполнение условий устойчивости является непременным атрибутом параметрического синтеза. Для того, чтобы воспользоваться алгебраическими критериями устойчивости, например, критерием Рауса-Гурвица, также необходимо выполнять операции над коэффициентами передаточной функции, которые являются полиномами от ее параметров. Такие примеры востребованности программного обеспечения алгебраических операций над полиномами, на наш взгляд, можно продолжать достаточно долго.

Другой областью применения методов разрабатываемого класса является использование элементарных операций над полиномами в многократно выполняемых циклах при решении вычислительных задач, использующих полиномиальную аппроксимацию функций. Узкая направленность методов данного класса (операции только над полиномами) позволяет создать более эффективные алгоритмы, чем при использовании аналогичных методов компьютерной алгебры, которые носят универсальный характер. К задачам такого рода относятся решение дифференциальных уравнений методом Пикара, с помощью рядов Ли и даже анализ нелинейных отображений и хаотического поведения [1].

Различия в назначении методов (выполнения основных операции) должно определять определенные различия в их реализации. Это можно реализовать или созданием нескольких методов одного и того назначения (например, суммирования) в рамках одного класса mpol, или созданием отдельных классов. В пользу последнего варианта свидетельствует жела-

ние даже на первых шагах не ограничиваться операциями над полиномами, а распространить разрабатываемые методы на дробно-рациональные функции. Под ними здесь понимается отношение (частное) двух полиномов. Основные арифметические операции над дробно-рациональными функции очень просто выражаются через операции над полиномами. То же относится и к операции дифференцирования, чего нельзя сказать об операции интегрирования. Поэтому трудно удержаться от распространения методов выполнения арифметических операций над полиномами на методы выполнения аналогичных операций над дробно-рациональными функциями, тем более, что область применения таких функций значительно шире области применения чисто полиномиальных моделей. Упомянутые методы в настоящее время объединены нами в класс ratiompol.

Основные методы класса mpol

mpol-объект p представляется в виде записи, состоящей из трех полей: строки коэффициентов p.coeff, массива имен переменных p.varnames, массива ячеек (номера и степени переменной в отдельном члене) p.numberdegry [7].

Методы класса разделены на основные, дополнительные, вспомогательные и служебные. К основным относятся: конструктор (mpol), преобразование типа (char), вывод на терминал (display) и арифметические операции (plus, minus, uminus, mtimes, mpower).

Из дополнительных упомянем только о методах:

- **valuempol**(p, nam, val) — вычисление значения полинома p при значении переменной nam = val;
- **valuempolallvar**(p, val) - вычисление значения полинома p при заданном значении val всех переменных;
- **polyofmpol**(a, p) - вычисление значение полинома a от одной переменной p класса mpol.

К вспомогательным отнесем методы:

- **compress**(p) — сжатие полинома во внутреннем представлении (приведение подобных и т. п.);
- **charfull**(p), **charshort**(p) — полное и краткое представление полинома. В последнем случае единичные коэффициенты и степени, также знаки умножения не выводятся.

Эти функции могут применяться как самостоятельно, так и при реализации других методов. Например, функция **compress**(p) используется при реализации алгебраических операций, а функция **charshort**(p) может использоваться вместо функции **char** в методе **display**. Последней возможностью мы воспользовались в приведенных ниже демонстрационных примерах для сокращения числа знаков в выводимой методом **display** строке результата выполнения операции или функции.

Вообще, большинство имен методов класса **mpol** предопределено синтаксисом объектно-ориентированного языка, а в случае возможности произвольного их назначения выбирались имена, близкие к названиям аналогичных функций и методов языка MATLAB и пакета Symbolic Math Toolbox.

Самое большое отличие в представлении полиномов в классе mpol и в языке Maple V заключается в возможности задания полинома, не объявляя переменные символьными, как в языке Maple V, или полиномами — объектами класса **mpol**, что, конечно, допустимо. То же самое можно сказать и о классе **ratiompol**. Для задания объекта этого класса можно воспользоваться конструктором с двумя аргументами — числителем и знаменателем, либо операцией деления класса **mpol**, которая возвращает результат в виде объекта класса **ratiompol**. Поясним сказанное на примерах.

Примеры операций над полиномами нескольких переменных

Демонстрацию методов класса **mpol** начнем с конструктора

```
>> q=mpol('v+y+1+x+z+2z+3*y+4x+ 5z^2+6y^3+x*z+10-x^2*z^1')
 q = v+y+1+x+z+2z+3y+4x+5z^2+6y^3+xz+10-x^2z.
```

Здесь использован сокращенный вариант функции преобразования полинома в строку **charshort** в методе **display**, в соответствии с которым знак умножения между именами переменных не выводится. В строке — аргументе конструктора он, конечно должен присутствовать. Различие между двумя вариантами этой функции покажем на примере.

```
>> char(q)
ans =
1*v+1*y+1+1*x+1*z+2*z+3*y+4*x+5*z^2+6*y^3+1*x*z+10-1*x^2*z.
```

Для более компактного представления полинома можно воспользоваться функцией compress.

```
>> q1=compress(q)
 q1 = v+4y+11+5x+3z+5z^2+6y^3+xz-x^2z.
```

Вопреки ранее высказанному намерению, нами выбрано имя, несколько отличающееся от наименования аналогичной функции **collect** в пакете Symbolic Math Toolbox по причине некоторого различия в форме представления результата этими функциями. Заметим, что в рассматриваемом классе необходимость явного использования этой функции может возникнуть только до первого применения какой-либо алгебраической операции, так как она неявно используется на заключительной стадии любой из них. Это свойство можно использовать для установления определенного порядка одночленов в полиноме. Например, для представления полинома по возрастающим степеням, можно использовать временный полином с определенным порядком переменных в качестве первого операнда в какой-либо алгебраической операции.

```
>> temp=mpol('1+x+y+z+v');
>> q=temp+q-temp
 q = 11+5x+4y+3z+v+5z^2+6y^3+xz-x^2z.
```

Конечно, полином можно задавать и в виде алгебраического выражения от переменных, которые предварительно должны быть определены как объекты класса mpol.

```
>> x=mpol('x');
>> q1 = 5*x+11-10*x^2.
```

И тот и другой способы равноправны и они могут смешиваться в одном выражении:

```
>> q3=q-q1+mpol('z*x^2')
 q3 = +4y+3z+v+5z^2+6y^3+xz+10x^2.
```

Продемонстрируем и другие методы данного класса. Сначала представим степенную функцию от полинома.

```
>> q4=mpol('1-2x');
>> q5=q4^5*mpol('1-y-z')
q5 = 1-10x+40x^2-80x^3+80x^4-32x^5-y+10xy-40x^2y+80x^3y-
80x^4y+ 32x^5y-z+10xz-40x^2z+80x^3z-80x^4z+32x^5z.
```

Далее, используя принятое в языке MATLAB представление полинома строкой коэффициентов, вычислим q6=q5^2+1:

```
>> q6=polyofmpol([1 0 1],q5).
```

Проверить правильность вычислений можно или «ручным» счетом, или определив значение полинома при численных значения всех или одной переменной.

```
>> q7=valuempol(q6,'x',1)
 q7 = 2-2y-2z+y^2+2yz+z^2;
>> q8=valuempolallvar(q6,[1 2 3])
q8 = 17.
```

Методы класса ratiompol включают в себя только конструктор, преобразование объекта в стоковый тип и основные алгебраические операции. Так как каждый объект данного класса представляет собой пару объектов класса mpol, то недостаток функций рассматриваемого класса легко компенсировать методами последнего класса. Продемонстрируем два способа конструирования объектов рассматриваемого класса и некоторые операции над ними.

Зададим передаточную функцию $W_0(s)$ = объекта регулирования в виде отношения полиномов $n_0(s)/d_0(s)$:

```
>> n0=mpol('s+1');
>> d0=mpol('s^2+s+1');
>> w0=n0/d0;
>> class(w0)
ans =
ratiompol,
```

а корректирующее звено непосредственным использованием конструктора рассматриваемого класса.

```
>> wcor=ratiompol('k','T*s+1')
  num.wcor = k;

_________________
denum.wcor = Ts+1.
```

Определим передаточную функцию по ошибке замкнутой системы.

```
>> werr=1/(1+wcor*w0)
  num.werr = Ts^3+s^2+Ts^2+s+Ts+1;

_________________
denum.werr = Ts^3+s^2+Ts^2+s+Ts+1+sk+k.
```

Метод display, как это только что показано, преобразует объект в две строки, представляющих числитель и знаменатель рациональной функции. Не составляет труда обеспечить доступ к любому полю объекта рассматриваемого класса.

Для наглядности выводимых данных желательно было бы сгруппировать коэффициенты при одинаковых степенях числителя и знаменателя, но этого не сделано по принципиальным соображениям: представление полиномов класса mpol допускает только бесскобочную форму представления. Вместо этого предлагается метод **koeff** вывода коэффициентов полинома по любой из переменных в классе mpol.

Использование ортогональных полиномов для решения «проблемы разбухания результатов»

Отметим еще один принципиальный момент в понимании полинома как алгебраической структуры, т. е. как объекта символьной алгебры, и как определенной формы аппроксимации функции, аналитическое выражение которой может быть даже и не известно. И в том и в другом случае может проявиться одна из самых серьезных проблем символьной математики — *проблема разбухания результатов* [3]. Если полином рассматривать не более чем аппроксимационную модель, то пути преодоления этой проблемы очевидны: полином высокой степени можно аппроксимировать полиномом более низкой степени. Эту аппроксимацию можно осуществить наилучшим в определенном смысле образом, если использовать разложения по той или иной системе ортогональных полиномов. Само разложение полинома по той или иной системе ортогональных полиномов представляет собой сравнительно простую задачу, сводящуюся к операциям над полиномами. По всей видимости, основная трудность при решении данной задачи заключается в выборе или формировании системы ортогональных полиномов, особенно когда речь идет о полиномах нескольких переменных.

Имеется достаточно большое число систем ортогональных полиномов (одной переменной) с хорошо изученными свойствами. Для пользова-

теля, имеющего представление об аппроксимирующих свойствах классических систем ортогональных полиномов, не составит труда выбрать наиболее подходящую из них для каждого конкретного случая и воспользоваться ею для понижения порядка полиномов в случае возникновения проблемы разбухания результатов.

Обобщение результатов на случай нескольких переменных с теоретической точки зрения не представляет принципиальных трудностей, но настолько трудоемок, что редко встречается в практических приложениях. Исключение составляет только случай, когда метрическое пространство, в котором осуществляется минимизация ошибки аппроксимации, представляет собой произведение метрических пространств по каждой переменной в отдельности и, следовательно, система ортогональных полиномов равна произведению систем ортогональных полиномов по соответствующим переменным [8].

В остальных случаях следует провести ортогонализацию последовательности одночленов, что практически осуществимо для полиномов невысокой степени небольшого числа переменных. Для этого необходимо располагать значениями скалярных произведений членов ортогонализируемой последовательности. Если ограничиться рассмотрением одночленов, ортогональных с весом [8], то для проведения процесса ортогонализации необходимо располагать значениями моментов весовой функции. Выбор упомянутой весовой функции, по существу, означает выбор метрического пространства, в котором будет осуществляться наилучшее приближение полинома высокой степени полиномом более низкой степени.

С точки зрения теории функций понижение порядка аппроксимирующего полинома за счет разложения его по той или системе ортогональных полиномов лишено смысла. Однако рассматриваемая здесь задача понижения порядка полинома не относится к классической. Для этого достаточно привести пример, связанный с аппроксимацией произведения двух полиномов нескольких переменных. Если число слагаемых в произведении столь велико, что делает невозможным практическое выполнение дальнейших операций с его участием, то возникает желание наилучшим образом понизить порядок полученного полинома.

Простое отбрасывание одночленов высших степеней можно интерпретировать как аппроксимацию результирующего полинома посредством конечного отрезка ряда Тейлора. Эта аппроксимация является наилучшей только в окрестности нулевой точки. Для наилучшей аппроксимации в гиперкубе со сторонами [-1, 1] следует воспользоваться конечным отрезком разложения по системе полиномов, полученных в результате произведения систем полиномов Лежандра $\{L_i(x_j)\}$ по каждой переменной x_j в отдельности. Для аппроксимации в произвольном параллелепипеде (параллельном координатным осям) необходимо использовать произведение систем полиномов Лежандра, каждая из которых ортогональна на своем отрезке.

В теории функций рассматриваются полиномы Лежандра, ортогональные на фиксированном интервале [–1, 1]. По-видимому, считается, что при необходимости аппроксимации функции в ином интервале изменения аргумента он будет приведен к стандартному посредством линейного преобразования. После аппроксимации нетрудно осуществить обратное преобразование и получить требуемый результат. Тот же самый результат можно получить, используя «параметрические» полиномы Лежандра $\{L_i(x_j)\}$, ортогональные на отрезке $[a_j, b_j]$. Последний подход предпочтителен, когда приходится варьировать интервалом (параллелепипедом) аппроксимации для нахождения наилучшего приближения функций в заранее неизвестном интервале (параллелепипеде) решения системы нелинейных уравнений.

После введения нормирующего множителя, весовую функцию, принимающую постоянное значение в определенном параллелепипеде, можно интерпретировать как плотность распределения вероятностей равномерно распределенной в этом параллелепипеде случайной величины. Это позволит сравнительно легко вычислять моменты рассматриваемой весовой функции как смешанные моменты многомерной равномерно распределенной случайной величины. Вообще, вероятностная интерпретация задачи аппроксимации может оказаться более понятной для многих специалистов, нежели та же задача в терминах теории функций.

В свое время в теории автоматического управления широкой популярностью пользовался метод статистической линеаризации, самой существенной частью которого является использование коэффициентов линеаризации нелинейных элементов. Последние можно назвать первыми двумя коэффициентами разложения нелинейных функций по параметрическим полиномам Эрмита, если дополнить определяющую их весовую функцию нормирующим множителем и ввести два параметра так, чтобы она могла интерпретироваться как выражение плотности вероятности нормально распределенной случайной величины с произвольными параметрами.

В этом случае и задачу разложения функции по упомянутой системе ортогональных полиномов можно интерпретировать как задачу аппроксимации нелинейной функции полиномом из условия минимума среднеквадратической ошибки от замены первой из них второй при условии, что на вход преобразователя, описываемого нелинейной функцией, поступает нормально распределенный случайный сигнал с определенным математическим ожиданием и дисперсией. Из сказанного следует, что полученную таким образом аппроксимацию нельзя применять при изучении преобразований с теми же самыми моментами первого и второго порядков, но с иным законом распределения, нежели нормальный. В этом случае нельзя только гарантировать минимальность среднеквадратической ошибки, но это не затрагивает качественную характеристику аппроксимации.

С практической точки зрения выбор между полиномами Лежандра и

Эрмита определяется весом, с которым суммируются ошибки аппроксимации в различных точках пространства аппроксимации. Относительно полиномов Лежандра можно сказать, что они осуществляют аппроксимацию, не отдавая предпочтения ни одной точке из параллелепипеда, в котором весовая функция отлична от нуля. При вероятностной интерпретации приходится говорить о равномерном в этом параллелепипеде распределении.

Если иметь в виду радиофизические и радиотехнические приложения, то нормальный закон распределения гораздо чаще применяется, чем равномерный. Поэтому в первую очередь нами рассматривались именно полиномы Эрмита. Полиномы Лежандра использовались нами только для простоты характеристики характера аппроксимации функций нескольких переменных в многомерном пространстве. Для аналогичной характеристики полиномов Эрмита пришлось бы говорить не о параллелепипеде, в котором весовая функция постоянна, а об эллипсоидах рассеяния нормально распределенной величины.

Параметрические полиномы Эрмита

Под полиномами Эрмита $H_n = H_n(x)$ обычно понимаются полиномы, ортогональные на интервале (- ∞, ∞) с весом

$$\varphi(x) = \exp\left(-\frac{x^2}{2}\right).$$

Они удовлетворяют рекуррентному соотношению [9]

$$H_n(x) = xH_{n\text{-}1}(x) - (n-1)H_{n\text{-}2}(x), (H_0 = 1,\ H_1 = x),$$

по которому можно определить полином H_n любого порядка. Например,

$$H_0 = 1,$$
$$H_1 = x,$$
$$H_2 = x^2 - 1,$$
$$H_3 = x^3 - 3x,$$
$$H_4 = x^4 - 6x^2 + 3,$$
$$H_5 = x^5 - 10x^3 + 15x,$$
$$H_6 = x^6 - 15x^4 + 45x^2 - 15.$$

Если дополнить вес нормирующим множителем $(2\pi)^{-1/2}$, то данную весовую функцию можно рассматривать как плотность вероятности нормально распределенной случайной величины. От значения нормирующего множителя зависит величина нормы полиномов, но эта информация используется только при определении коэффициентов разложения функции. В настоящей работе, посвященной операциям над полиномами, численные методы определения коэффициентов разложения одних полиномов по дру-

гим не используются, поэтому нормы полиномов можно и не определять.

Формирование этих полиномов не трудно осуществить программно с использованием двух операций над полиномами одной переменной, но в этом нет практической необходимости. Для параметрических полиномов такая генерация уже представляет практический интерес, даже если речь идет о полиномах одной переменной.

Под параметрическими полиномами Эрмита $H_n = H_n(x, m, D)$ будем понимать полиномы, ортогональные на интервале $(-\infty, \infty)$ с весом

$$w(x) = \frac{1}{\sqrt{2\pi D}} \exp\left(-\frac{(x-m)^2}{2D} \right). \quad (1)$$

Для простоты рассуждений ограничимся пока рассмотрением полиномов $H_n = H_n(x, D) = H_n(x, 0, D)$. Не трудно показать, что они удовлетворяют рекуррентному соотношению

$$H_n(x,D) = xH_{n-1}(x,D) - (n-1)DH_{n-2}(x,D), (H_0 = 1, H_1 = x),$$

по которому также не трудно определить полином H_n любого порядка.

Для вычисления полиномов можно использовать функцию

```
function H=hermD(x,D,n)
H=[1 x x^2-D];
for i=3:n
   Hi=H(i)*x-(i-1)*D*H(i-1);
   H=[H Hi];
end
if n == 1
  H=[1 x];
end
```

Например,

```
>> x=mpol('x');
>> D=mpol('D');
>> H=hermD(x,D,7);
>> disph(H)
H(0) = 1;
H(1) = x;
H(2) = x^2-D;
H(3) = x^3-3xD;
H(4) = x^4-6x^2D+3D^2;
H(5) = x^5-10x^3D+15xD^2;
H(6) = x^6-15x^4D+45x^2D^2-15D^3;
H(7) = x^7-21x^5D+105x^3D^2-105xD^3.
```

Приведенным выражениям можно придать более привычный вид:

$$H_0 = 1,$$
$$H_1 = x,$$
$$H_2 = x^2 - D,$$
$$H_3 = x^3 - 3Dx,$$
$$H_4 = x^4 - 6Dx^2 + 3D^2,$$
$$H_5 = x^5 - 10Dx^3 + 15D^2x,$$
$$H_6 = x^6 - 15Dx^4 + 45D^2x^2 - 15D^3.$$

Отсюда следует, что

$$1 = H_0,$$
$$x = H_1,$$
$$x^2 = H_2 + D,$$
$$x^3 = H_3 + 3DH_1,$$
$$x^4 = H_4 + 6DH_2 + 3D^2,$$
$$x^5 = H_5 + 10DH_3 + 15D^2H_1,$$
$$x^6 = H_6 + 15DH_4 + 45D^2H_2 + 15D^3.$$

Последние уравнения можно рассматривать как результат разложения различных степеней переменой x по системе полиномов Эрмита $H_n = H_n(x, D)$. Тогда и коэффициенты при этих полиномах можно рассматривать как коэффициенты разложения соответствующей функции в параметрический ряд Эрмита. Заметим, что при их определении численные методы не использовались даже в символической форме, как и значения нормы полиномов.

Очевидно, что каждое из только что приведенных выражений самым непосредственным образом может быть использовано для аппроксимации полиномов степени k полиномом степени q ($q < k$). Для этого в ортогональных разложениях x^j ($j > q$) отбросить слагаемые, содержащие полиномы H_n ($n > q$). Получающееся таким образом приближение обозначим $x^{j, q}$.

Например, если необходимо аппроксимировать полином шестого порядка полиномом четвертого порядка, то степени x^5 и x^6 в представлении полинома следует заменить приближенными выражениями

$$x^{5,4} = 10DH_3 + 15D^2H_1 = 10Dx^3 - 15D^2x,$$
$$x^{6,4} = 15DH_4 + 45D^2H_2 + 15D^3 = 15Dx^4 - 45D^2x^2 + 15D^3.$$

Сформировать систему параметрических полиномов $H_n = H_n(x, m, D)$ с двумя произвольными параметрами гораздо проще, чем доказать, что они ортогональны. В этом отношении может помочь упоминавшаяся выше вероятностная интерпретация полиномов Эрмита.

Данная система полиномов может быть получена из системы поли-

номов $H_n = H_n(x, D)$ подстановкой $x = x - m$. Для ее формирования может быть использована приведенная выше программа **hermD**. Например,

```
>> H=hermD(mpol('x-m'),D,6);
>> disph(H)
H(0) = 1;
H(1) = x-m;
H(2) = x^2-2xm+m^2-D;
H(3) = x^3-3x^2m+3xm^2-3xD-m^3+3mD;
H(4) = x^4-4x^3m+6x^2m^2-6x^2D-4xm^3+12xmD+m^4-6m^2D+3D^2;
H(5) = x^5-5x^4m+10x^3m^2-10x^3D-10x^2m^3+30x^2mD+5xm^4-
30xm^2D+15xD^2-m^5+10m^3D-15mD^2;
H(6) = x^6-6x^5m+15x^4m^2-15x^4D-20x^3m^3+60x^3mD+15x^2m^4-
90x^2m^2D+45x^2D^2-6xm^5+60xm^3D-90xmD^2+m^6-15m^4D+45m^2D^2-15D^3.
```

Для проверки ортогональности данных полиномов воспользуемся вероятностной интерпретацией скалярного произведения $H_n(x)H_m(x)$ в рассматриваемом пространстве с весом (1). Оно равно математическому ожиданию функции $H_n(x)H_m(x)$ случайного аргумента x, плотность вероятности которого равна весовой функции (1). Математическое ожидание $E\{p(x)\}$ полинома $p(x)$ является линейной комбинацией моментов случайной величины x.

Значения моментов нормального распределения хорошо известны:

$$E\{x\} = m,\ E\{x^2\} = D + m^2,\ E\{x^3\} = 3Dm + m^3,\ E\{x^4\} = 3D^2 + 6Dm^2 + m^4, \ldots.$$

Определим некоторые скалярные произведения $E\{H_n(x)H_m(x)\}$.

```
>> D=mpol('D'); m=mpol('m');
>> Ex = m; Ex2 = D + m^2; Ex3 = 3*D*m + m^3; Ex4 = 3*D^2 +  6*D*m^2 + m^4;
>> H=hermD(mpol('x-m'),D,6);
>> char(H(4))
ans =
1*x^3-3*x^2*m+3*x*m^2-3*x*D-1*m^3+3*m*D.
```

Для определения математического ожидания $E\{H_0(x)H_4(x)\}$ в выражении $H_4(x)$ необходимо заменить степени x^j величины x соответствующими моментами. Так как таких функций в разработанном классе не имеется, проделаем эту подстановку в командной строке. В более длинных выражениях это же можно проделать в редакторе системы MATLAB.

```
>> EH4=1*Ex3-3*Ex2*m+3*Ex*m^2-3*Ex*D-1*m^3+3*m*D
 EH4 = ;
```

Вывод пустой строки свидетельствует о нулевом значении данного выражения. Для того, чтобы убедиться в этом воспользуемся функцией **charfull**, которая и предназначена для вывода содержимого всех полей объекта в форме текстовой строки.

```
>> charfull(EH4)
ans =
+0*D^1*m^1+0*m^3
>> H1H2=H(2)*H(3)
 H1H2 = x^3-3x^2m+3xm^2-m^3-xD+mD;
>> char(H1H2)
```

```
ans =
1*x^3-3*x^2*m+3*x*m^2-1*m^3-1*x*D+1*m*D
>> EH1H2=1*Ex3-3*Ex2*m+3*Ex*m^2-1*m^3-1*Ex*D+1*m*D
 EH1H2 = ;
>> H1H3=H(2)*H(4);
>> char( H1H3)
ans =
1*x^4-4*x^3*m+6*x^2*m^2-4*x*m^3-3*x^2*D+6*x*m*D+1*m^4-3*m^2*D
>>EH1H3=1*Ex4-4*Ex3*m+6*Ex2*m^2-4*Ex*m^3-3*Ex2*D+  6*Ex*m*D+  1*m^4-
3*m^2*D
 EH1H3 = .
```

Ограничимся этими примерами проверки ортогональности полученной системы полиномов.

Система полиномов нескольких переменных, как уже отмечалось, может быть получена произведением систем полиномов по каждой переменной в отдельности. Весовая функция, с которой ортогональны полученные таким образом полиномы, равна произведению весовых функций вида (1), что соответствует плотности вероятности нормально распределенного случайного вектора с независимыми координатами.

При необходимости получения системы полиномов, ортогональных с весовой функцией, равной плотности вероятности многомерного вектора общего вида, следует известным образом перейти к вектору с некоррелированными (а следовательно, и независимыми) составляющими и уже для них составить систему ортогональных полиномов описанным выше образом. Уже разработанные методы класса mpol позволяют это сделать сравнительно просто. Без использования программного обеспечения формирование и использование ортогональных полиномов Эрмита даже третьего — четвертого порядков только двух коррелированных переменных связано со значительными трудностями [10].

Литература

1. *Тан К. Ш., Стиб И.-Х., Харди Й.* Символьный C++: Введение в компьютерную алгебру с использованием объектно-ориентированного программирования: пер. со 2-го англ. изд.— М.: Мир, 2001.— 622 с.
2. *Дьяконов В. П., Абраменкова И. В., Круглов В. В.* MATLAB 5.3.1 с пакетами расширений.— М.: Нолидж, 2001.— 880 с.
3. *Дьяконов В. П.* Компьютерная математика. Теория и практика.— М.: Нолидж. 2001.— 1296 с.
4. *Дьяконов В., Круглов В.* Математические пакеты расширения MATLAB. Специальный справочник.— СПб.: Питер. 2001.— 480 с.
5. *Stumfeld B.* Solving Systems of Polynomial Equations // Conf. Board of Mathematical Sciences by the American Mathematical Society Providence.— Rhode Island USA, 2002.— 152 p.
6. *Потемкин В. Г.* Введение в MATLAB.— М.: Диалог-МИФИ, 2000.— 247 с.

7. *Птичкин В. А*. Класс полиномов многих переменных // Тез. докл. Всерос. научн. конф. «Проектирование научных и инженерных приложений в среде MATLAB» (28–29 мая 2002).— М.: ИПУ РАН, 2002.— С.42–43.
8. *Колмогоров А. Н., Фомин С. В.* Элементы теории функций и функционального анализа.— М.: Наука, 1968.— 496 с.
9. *Джексон Д.* Ряды Фурье и ортогональные полиномы: пер. с англ.— М.: Государственное издательство иностранной литературы, 1948.— 260 с.
10. *Птичкин В. А.* Анализ многослойных нейронных сетей методами статистической линеаризации и полиномиальной аппроксимации // Нейрокомпьютеры: разработка, применение.— 2004.— №1.— С.5–18.

УДК 004.9

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ОБРАЗОВАНИЯ ЭЛЕКТРИЧЕСКОГО ПРОБОЯ В ПАКЕТЕ MATLAB

Росинский А. Д., Коломийцева С. В.
Дальневосточный государственный университет путей сообщения, Хабаровск,
e-mail: svk@festu.khv.ru

Введение

Для проектирования высоковольтных установок, работающих в атмосферном воздухе, необходимо знать зависимость напряжения пробоя, скорость нарастания тока от геометрических размеров катода и анода, формы электрического поля и атмосферных условий. Но экспериментальное измерение этих параметров является дорогостоящим и сложным процессом. В связи с этим часто используются различные математические модели, приближенно вычисляющие указанные параметры.

Цель работы — создание и исследование модели пробоя для системы игла-плоскость, учитывающей лавинную ионизацию, рекомбинацию и появление фотоэлектронов, а также разработка графического интерфейса пользователя в среде MATLAB.

Общая постановка задачи

При достижении электрическим полем некоторого критического значения в веществе быстро образуется стримерный канал (плазменный канал с высокой проводимостью), его образование сопровождается интенсивным свечением и ударной волной.

В работе моделируется явление пробоя под действием лавинной ионизации. Упрощенная схема акта ионизации выглядит так [1]: налетающий электрон взаимодействует с внешним атомным электроном, который подлежит вырыванию и считается неподвижным (пренебрегаем энергией теплового движения по сравнению с энергией ионизации). Столкновение двух электронов упругое. Налетающий электрон при столкновении передает атому некоторую энергию. Если эта энергия превышает его энергию связи в атоме, то электрон освобождается — произошла ионизация атома. Этот механизм ионизации атома из основного состояния называется прямой ионизацией, для нее характерны высокие значения напряжений. Таким образом, число электронов двигающихся к аноду увеличивается с пройденным расстоянием от катода, эта группа свободных электронов называется лавиной электронов. Для ее образования необходимы первичные электроны, инициаторы лавины. В роли такого источника (первичного ионизатора)

выступает острие иглы в плоском конденсаторе, находящейся на катоде и имеющий нулевой потенциал, вследствие автоэлектронной эмиссии.

Лавина развивается в виде распространяющегося облака электронов, позади нее остается положительный пространственный заряд ионов. После прохождения лавиной разрядного промежутка электроны попадают на анод, а положительные ионы распределены вдоль канала прохождения лавины. Сильно ионизированный ствол лавины испускает много фотонов, которые выбивают электроны из нейтральных молекул газа (фотоэлектроны), инициируя дополнительные лавины. Если поле пространственного заряда, оставленного основной лавиной сравнимо с внешним, то новые лавины будут двигаться к стволу главной лавины, таким образом, вдоль нее будет наблюдается максимальный рост ионов. Процесс происходит как самораспространяющийся стример. Стример представляет собой плазму, состоящую из электронов и ионов.

Стримерная теория искрового разряда основана на рассмотрении отдельных электронных лавин, перехода лавины в стример и механизма роста стримера [2, 3].

Решение задачи в общей постановке

В данной работе рассматривается макроскопическая трехмерная модель стримера. Макроскопической называется модель, в которой пространство разбивается на малые квазиоднородные элементы, характеризующиеся числом частиц, энергией, плотностью заряда и т. п.

Число частиц в элементарном объеме характеризуется начальной концентрацией и потоком (броуновским движением и дрейфом под действием внешнего поля для заряженных частиц).

Заряженные частицы появляются вследствие автоэлектронной эмиссии. Тогда поток частиц j можно описать выражением

$$j = \frac{e^2}{2\pi h}\sqrt{\frac{\varepsilon_0}{e\varphi}}\frac{E^2}{\varepsilon_0 + e\varphi}\exp\left(-\frac{8\pi\sqrt{2m}(e\varphi)^{3/2}}{3hE}\right),$$

где h — постоянная Больцмана, ε_0 — энергия Ферми, φ — работа выхода из металла, m, e — масса и заряд электрона, E — напряженность электрического поля.

Тепловое (броуновское) движение определяется средней скоростью $\overline{V}$ частиц, которая находится из распределения Максвелла по скоростям. Число частиц успевающих свободно участвовать в броуновском движении за время τ определяется условием:

$$N = \frac{\pi}{8}\tau\overline{V}n,$$

где n — концентрация частиц, а множитель $\frac{\pi}{8}$ появляется при расчете средней скорости вдоль одной оси с учетом изотропности направлений. Эти частицы с вероятность 0,5 вылетают из ячейки. Тогда их число N_{in} подчиняется биноминальному закону, который при вычислениях приближается нормальным (число частиц достаточно велико $\sim 10^5$).

$$N_{in} = \Phi\left(\frac{N}{2}, \frac{N}{2}\right).$$

Для заряженных частиц основной вклад в поток составляет их дрейф под действием электрического поля. Сначала найдем распределение потенциала из уравнения Пуассона,

$$\Delta\varphi = \frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} + \frac{\partial^2\varphi}{\partial z^2} = -\frac{\rho}{\varepsilon_0}, \tag{1}$$

пользуясь методом конечных разностей, из (1) получим рекуррентную формулу:

$$\begin{aligned}\tilde{\varphi}(x,y,z) = \frac{1}{6}\big(&\varphi(x,y+\Delta,z) + \varphi(x-\Delta,y,z) + \varphi(x+\Delta,y,z) + \\ &+\varphi(x,y-\Delta,z) + \varphi(x,y,z-\Delta) + \varphi(x,y,z+\Delta)\big) + \frac{\Delta^2}{6\varepsilon_0}\rho(x,y,z).\end{aligned} \tag{2}$$

Расчет заканчивается, когда изменение потенциала (2) на следующем шаге будет меньше необходимой погрешности, введенной пользователем. По вычисленному значению потенциала находится напряженность поля:

$$E = -grad(\varphi).$$

Скорость дрейфа электронов можно связать с напряженностью электрического поля через их подвижность μ:

$$\vec{V}_{др} = \mu\vec{E}.$$

Изменение частиц за время τ определяется потоком $\vec{J}$:

$$\left.\begin{aligned}\vec{J} &= n\vec{V}_{др} = n\mu\vec{E} \\ \Delta N &= (\vec{J}\cdot\vec{S})\tau\end{aligned}\right\} \Rightarrow \begin{cases}\Delta N_x = nS\mu E_x \\ \Delta N_y = nS\mu E_y \\ \Delta N_y = nS\mu E_z\end{cases},$$

где S — площадь боковой поверхности ячейки, n — концентрация частиц, τ — время одного цикла моделирования, $\Delta N_x, \Delta N_y, \Delta N_z$ — изменение частиц по соответствующим направлениям. Также в изменение числа заряженных частиц вносят вклад рекомбинация и ионизация.

Рассмотрим процесс рекомбинации, когда при каждом соударении электрон–катион образуется нейтральная молекула с излучением фотона.

$$N_{соуд} = \alpha N_+ N_-,$$

$N_{соуд}$ — число соударений, определяющая убыль заряженных частиц и рост числа нейтральных, α — коэффициент пропорциональности, зависящий от вида газа и температуры.

Ионизация происходит если энергия частиц больше пороговой. Считается, что вся энергия, излученная при рекомбинации и набранная под действием внешнего электрического поля, идет на ионизацию.

Энергия электрического поля, переходящая в энергию частиц, определяется его работой над зарядами, считая, что за время τ соударений не происходит можно записать

$$\left.\begin{array}{l} E_{эл} = eN(\vec{E}\cdot\vec{l}) \\ \vec{l} = \tau\vec{V}_{др} \end{array}\right\} E_{эл} = eN\tau(\vec{E}\cdot\vec{V}_{др}),$$

где $\vec{l}$ — длина пути, пройденная за время τ.

Энергия излучения складывается из двух составляющих: излучение самой ячейки и излучение ее соседей. При расчете излучения соседних ячеек они заменялись точечными источниками, расположенными на среднем расстоянии, расстоянии между центрами ячеек:

$$\Delta E_{изл}^{i,j,k} = S_{i,j,k}\frac{d\Omega}{4\pi} = \frac{S_{i,j,k}}{4\pi}\frac{S_{эф}N^{2/3}}{(x_{i0}-x_i)^2+(y_{j0}-y_j)^2+(z_{k0}-z_k)^2},$$

где $\Delta E_{изл}^{i,j,k}$ — энергия излучения в ячейке с координатами x_{i0}, y_{j0}, z_{k0} от ячейки с координатами x_i, y_j, z_k, $S_{i,j,k}$ — энергия излучения этой ячейки за один шаг цикла моделирования, $d\Omega$ — телесный угол, под которым она видна, точнее частицы находящиеся в ней. $S_{эф}$ — Эффективная площадь соударения частиц с фотонами, N — их число, коэффициент 2/3 учитывает возможность заслонения частицами друг друга при однородном заполнении объема.

Тогда энергия от всех соседей запишется через сумму по всему объему, исключая рассматриваемую ячейку.

$$E_{изл}^{сосед} = \sum_{i\neq i0, j\neq j0, k\neq k0}\Delta E_{изл}^{i,j,k}.$$

Энергию, излучаемую ячейкой самой в себя, можно оценить как:

$$\Delta E_{изл}^{собст} = \frac{9}{4\pi}S_{i0,j0,k0},$$

тогда суммарная энергия излучения, набранная за один шаг:

$$E_{изл} = E_{изл}^{собст} + E_{изл}^{сосед}.$$

Энергия, набранная частицами в рассматриваемом объеме, выразится:

$$E = E_{эл} + E_{изл}.$$

Считая, что вся энергия идет на ионизацию легко найти число вновь образованных пар электрон–ион $\Delta N_{ион}$.

$$\Delta N_{ион} = \frac{E}{E_{ион}},$$

где $E_{ион}$ — энергия ионизации.

Тогда новое значение числа ионов и электронов выражается через предыдущие следующим образом:

$$\widetilde{N}_+ = N_+ - \Delta N_{рек} + \Delta N_{ион}$$
$$\widetilde{N}_- = N_- - \Delta N_{рек} + \Delta N_{ион},$$
$$\widetilde{N} = N + \Delta N_{рек} - \Delta N_{ион}$$

где $\widetilde{N}_+, N_+$ — число катионов на новом и предыдущем шагах, $\widetilde{N}_-, N_-$ — число электронов, $\widetilde{N}, N$ — число нейтральных частиц, $\Delta N_{рек}$ — число рекомбинаций, $\Delta N_{ион}$ — число ионизированных молекул.

Теперь можно найти плотность заряда:

$$\rho = \frac{(N_+ - N_-)e}{a^3}.$$

Зная плотность заряда можно найти распределение потенциала и начать новую итерацию цикла вычислений.

Заключение

Для программной реализации модели используется пакет математических программ MATLAB.

В настоящее время одним из обязательных атрибутов любой прикладной программы является интерактивный интерфейс пользователя, который разрабатывается для многократно решаемых задач с несколькими входными параметрами [4]. Наличие у программы графического интерфейса обладает следующими преимуществами: освобождает пользователя от необходимости вникать в детали программирования данного программного продукта, а также позволяет менять исходные данные, не прекращая выполнения текущей программы, то есть в реальном времени наблюдать за изменением решения задачи.

Управляющая форма с пользовательским интерфейсом, через которую осуществляется реализация изменения параметров, представлена на рис. 1.

Данный интерфейс позволяет отображать временные зависимости различных токов — ток, вытекающий с анода, регистрируемый на катоде (рис. 1, *а*) и протекающий через устройство (учитывается ток смещения) (рис. 1, *б*); изменять расстояние между обкладками, напряжение на них, параметры моделирования. Также меню «Свойства» позволяет менять параметры среды.

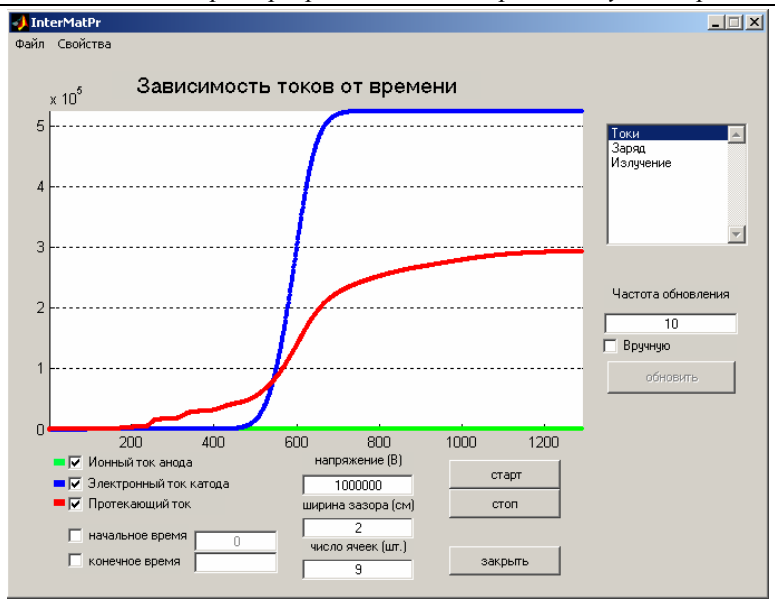


Рис. 1. Графический интерфейс для модели пробоя с результатами моделирования.

На рис. 1, *а* отчетливо видны две области: экспоненциального роста и насыщения. Первая связана с развитием электронной лавины, а вторая — с насыщением тока вследствие ограниченности тока электронной эмиссии.

При увеличении напряжения увеличивается ток электронной эмиссии, вследствие чего увеличивается ток насыщения, но коэффициент умножения электронов не меняется, кроме того, уменьшается время образования пробоя.

В целях экономии времени и вычислительных ресурсов используется малое количество элементарных ячеек.

Разработанная модель не учитывает тепловую энергию, прилипание электронов и зависимость подвижности зарядов от электрического поля. Несмотря на эти допущения, модель позволяет получать результаты, хорошо согласующиеся с экспериментальными данными и общими представлениями о пробое [1]. При больших расстояниях или высоких концентрациях модель хорошо описывает развитие стримера, а при малом расстоянии отчетливо видно образование таундесовского пробоя.

Литература

1. *Райзер Ю. П.* Физика газового разряда.— М: Наука, 1987.
2. *Мик Дж., Крегс Дж.* Электрический пробой в газах / Перевод с англ. под ред. д. т. н. Камелькова В. С.— М: Изд–во иностранной литературы, 1960.
3. *Леб Л. Б.* Основные процессы электрических разрядов в газах.— М: Гостехиздат, 1950.
4. *Поршнев С. В.* Компьютерное моделирование физических процессов в пакете MATLAB.— М: Горячая линия–Телеком, 2003.— 592 с.

УДК 519.68

СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ В MATLAB: MATHEMATICA SYMBOLIC MATH TOOLBOX

Семченок Н. М.
Санкт-Петербургский государственный университет, Санкт-Петербург,
e-mail: nikolay@paloma.spbu.ru

Чаще всего для проведения символьных вычислений в MATLAB используется пакет Symbolic Math Toolbox [1], являющийся совместной разработкой компаний MathWorks Inc. и Maplesoft Inc. Последняя версия этого пакета построена на базе вычислительного ядра системы Maple 8, которое, вобрав в себя практически все известные на сегодняшний день алгоритмы символьной математики, является инструментом, позволяющим пользователю MATLAB выполнять преобразование и упрощение математических выражений, решать трансцендентные уравнения, находить пределы функций и последовательностей, производные и первообразные, разлагать функции в степенные ряды, а также находить общие и частные решения обыкновенных дифференциальных уравнений.

Однако для того чтобы получить возможность работать с пакетом Symbolic Math Toolbox, пользователю системы MATLAB необходимо приобрести соответствующую лицензию, что, по мнению автора, делать не всегда целесообразно по следующим причинам.

Во-первых, стоимость Symbolic Math Toolbox часто сравнима со стоимостью полной версии системы Maple, в состав которой помимо вычислительного ядра входит не только достаточно удобный графический интерфейс, но и ряд дополнительных библиотек. Поэтому имеет смысл приобрести последнюю версию системы Maple, а затем, воспользовавшись API OpenMaple [3], попытаться интегрировать ее вычислительное ядро с системой MATLAB.

Во-вторых, пользователь может уже располагать не менее мощной чем Maple системой компьютерной алгебры Mathematica компании Wolfram Research Inc. В этом случае разумным представляется подключение к MATLAB вычислительного ядра системы Mathematica.

Пакет Mathematica Symbolic Math Toolbox призван решить обе эти задачи. С помощью этого пакета пользователь MATLAB, располагающий системами Maple 9 или Mathematica 4.x/5.x, получает доступ ко всем возможностям стандартного пакета Symbolic Math Toolbox. При этом синтаксис команд обоих пакетов совпадает, что исключает возникновение проблем при переносе приложений MATLAB между системами, работающими с различными ядрами символьной математики.

Строение представляемого пакета во многом повторяет строение Symbolic Math Toolbox, но содержит оригинальный код m-функций. Так же как и стандартный пакет, Mathematica Symbolic Math Toolbox состоит из функций, определяющих класс символьных выражений и методов для работы с ними, а также функций, отвечающих за организацию взаимодействия с вычислительным ядром системы Mathematica.

В Symbolic Math Toolbox, чтобы обратиться к ядру Maple, необходимо передать mex-функции **maplemex** строку, содержащую одну или несколько синтаксически правильных команд Maple. При этом результат также будет возвращен в виде текстовой строки.

В Mathematica Symbolic Math Toolbox для обращения к ядру системы Mathematica используется API J/Link [2], возможности которого позволяют возвращать результат не только в текстовой, но и в графической форме. Чтобы продемонстрировать эти возможности далее приведен программный код пяти m-функций, обеспечивающих простейшую форму взаимодействия между системами MATLAB и Mathematica.

Запуск ядра системы Mathematica можно осуществить с помощью m-функции **mkstart**.

```
function mkstart
%Глобальная переменная mklink служит указателем на
%текущее ядро системы Mathematica.
%Глобальная переменная mlstatus принимает значение
%true, если ядро загружено, и false
%в противном случае.
global mklink mlstatus
%Пытаемся запустить ядро MathKernel.
try
  mklink = com.wolfram.jlink.MathLinkFactory.createKernelLink('-linkmode  launch -
linkname D:/Math5/mathkernel');
%Игнорируем все пакеты, возвращаемые ядром системы %Mathematica в процессе загрузки
  mklink.discardAnswer;
  mlstatus = true;
catch
  error(sprintf('Не удалось запустить MathKernel'));
  mlstatus = false;
end
```

Запустим вычислительное ядро системы Mathematica из командной строки MATLAB.

```
>> mkstart
```

Функция **mathcmd** является аналогом функций **maplemex** и **maple** из Symbolic Math Toolbox.

```
function  outexpr = mathcmd ( inexpr )
global mklink mlstatus
  if mlstatus
     try
%Если ядро запущено, то преобразуем аргумент функции
```

```
%в java-строку.
      cmd = java.lang.String( inexpr );
%метод evaluateToInputForm отправляет ядру mklink
%строку команд системы Mathematica и
%возвращает полученный результат так же
%в виде строки (в формате InputForm).
%Второй аргумент evaluateToInputForm указывает,
%что строка результата не должна содержать переносов.
      result = mklink.evaluateToInputForm(cmd, 0);
%Преобразуем результат в строку MATLAB.
      outexpr = char(result);
    catch
   outexpr = inexpr
end
end
```

Найдем с помощью **mathcmd** первообразную некоторой функции.

```
>> mathcmd('Integrate[x^2*Sin[x],x]')

ans =

-((-2 + x^2)*Cos[x]) + 2*x*Sin[x]
```

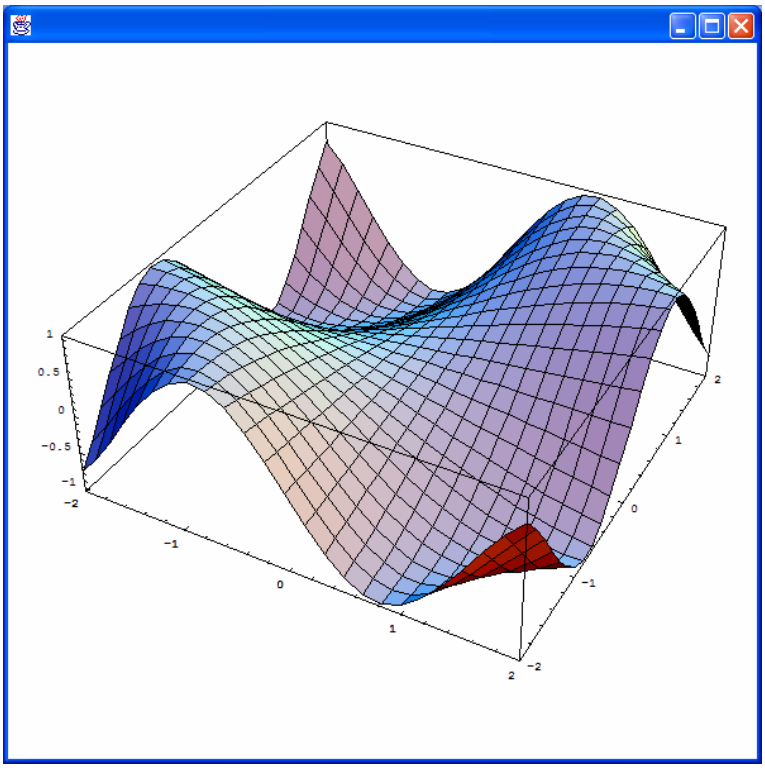
Функция **mathplot** принимает в качестве аргумента какую-либо графическую команду системы Mathematica и возвращает окно с соответствующим рисунком.

```
function mathplot( inexpr )
global mklink mlstatus
if mlstatus
  try
%Если ядро запущено, то преобразуем аргумент функции
%в java-строку.
  cmd = java.lang.String( inexpr );
%Создаем экземпляр MathFrame,
%являющегося наследником java.awt.Frame.
  mathFrame = com.wolfram.jlink.MathFrame;
%Устанавливаем размер окна.
  mathFrame.setSize(600, 600);
%Создаем экземпляр MathCanvas,
%являющегося наследником java.awt.Canvas и
%подсоединяем его к ядру Mathematica.
  mathCanvas = com.wolfram.jlink.MathCanvas(mklink);
%Устанавливаем размер рисунка.
  mathCanvas.setSize(550, 550);
%Помещаем рисунок в окно.
  mathFrame.add(mathCanvas);
%Указываем, что рисунок в данном случае будет
%графиком, а не формулой.
  mathCanvas.setImageType(com.wolfram.
  jlink.MathCanvas.GRAPHICS);
%Указываем, что для создания рисунка будут
%использоваться средства графического интерфейса
```

```
%системы Mathematica.
  mathCanvas.setUsesFE(1);
%Указываем выражение, вычисление
%которого порождает рисунок.
  mathCanvas.setMathCommand(cmd);
%Делаем окно видимым.
  mathFrame.setVisible(true);
 catch
  outexpr = inexpr
end
end
```

Построим с помощью **mathplot** график функции двух переменных.

```
>> mathplot('Plot3D[Sin[x*y], {x,-2,2},{y,-2,2}]')
```



Функция **mathexpr** вычисляет заданный набор выражений и возвращает графическое представление результата.

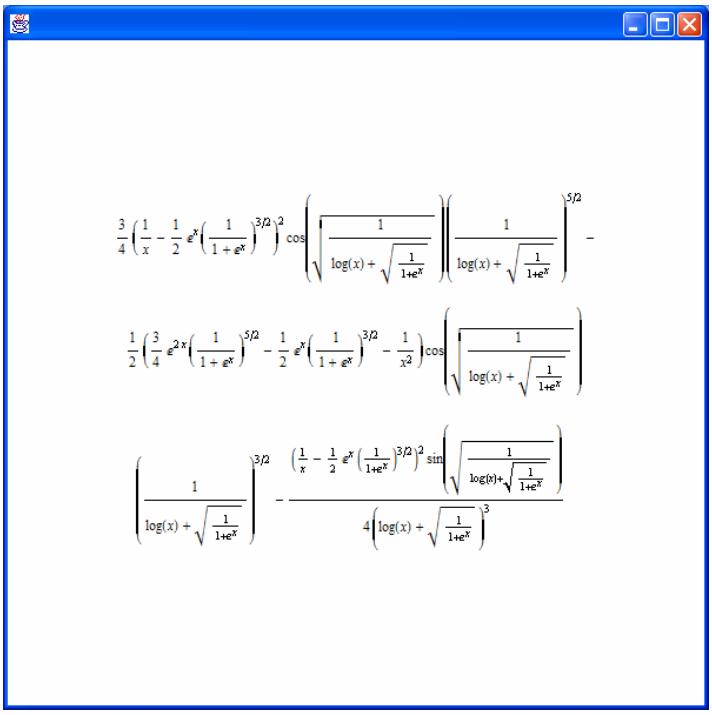
```
function mathexpr( inexpr )
global mklink mlstatus
if mlstatus
   try
%Если ядро запущено, то преобразуем аргумент функции
%в java-строку.
cmd = java.lang.String( inexpr );
%Создаем экземпляр MathFrame,
%являющегося наследником java.awt.Frame.
mathFrame = com.wolfram.jlink.MathFrame;
%Устанавливаем размер окна.
mathFrame.setSize(600, 600);
```

```
%Создаем экземпляр MathCanvas,
%являющегося наследником java.awt.Canvas и
%подсоединяем его к ядру Mathematica.
mathCanvas = com.wolfram.jlink.MathCanvas(mklink);
%Устанавливаем размер рисунка.
mathCanvas.setSize(500, 500);
mathFrame.add(mathCanvas);
%Указываем, что рисунок в данном случае будет
%графическим представлением математической формулы.

mathCanvas.setImageType(com.wolfram.jlink.MathCanvas.TYPESET);
%Задаем формат представления формулы.
mathCanvas.setUsesTraditionalForm( 1 );
%Указываем выражение, вычисление
%которого порождает рисунок.
mathCanvas.setMathCommand( cmd );
%Делаем окно видимым.
mathFrame.setVisible(true);
catch
   outexpr = inexpr
end
end
```

Выведем на экран результат дифференцирования достаточно громоздкой функции.

```
>> mathexpr('D[Sin[Sqrt[1/(Log[x]+...
  Sqrt[1/(1+Exp[x])])]],{x,2}]')
```



При завершении работы MATLAB ядро системы Mathematica выгружается автоматически. Чтобы выгрузить ядро самостоятельно, достаточно обратиться к методу **close**.

```
function  mkshutdown
global mklink mlstatus
if mlstatus
   mklink.close;
   mlstatus=0;
end
```

Ознакомившись с основными принципами взаимодействия систем MATLAB и Mathematica, совершим беглый обзор возможностей пакета Mathematica Symbolic Math Toolbox. При этом мы не будем останавливаться на описании синтаксиса и возможностей функций из этого пакета, так как в большинстве своем они полностью совпадают с одноименными функциями из стандартного пакета Symbolic Math Toolbox, а ограничимся лишь краткой демонстрацией некоторых функций, на примере работы которых покажем различия между этими двумя пакетами.

Прежде чем начать работу с пакетом Mathematica Symbolic Math Toolbox, его необходимо установить и правильно сконфигурировать. Для этого надо скопировать все файлы пакета в подкаталог toolbox, запустить систему MATLAB и выполнить команду **symbengine**. В результате будет произведен поиск доступных на данном компьютере вычислительных ядер систем Maple и Mathematica и пользователю будет предоставлена возможность выбрать инструмент для проведения символьных вычислений.

```
>> symbengine ( 'setup' )

Select a symbolic engine:
[1] Symbolic Math Toolbox ( Maple 8 OEM kernal )
[2] Maple 9.x in D:\Maple9
[3] Mathematica 5.x in D:\Math5
[4] Mathematica 4.x in D:\Math42

Symbolic Engine: 3

Please verify your choices:
Mathematica 5 in D:\Math5

Are these correct?([y]/n): y
```

Если выбран первый пункт списка, то символьные вычисления будут по-прежнему проводиться с помощью пакета Symbolic Math Toolbox, то есть с помощью ядра системы Maple 8.

Выбор второго пункта обеспечит возможность проведения символьных преобразований с помощью ядра системы Maple 9. Если при этом на машине пользователя установлен стандартный пакет Symbolic Math Toolbox, то отвечающая за обращение к ядру Maple и находящаяся в каталоге \toolbox\symbolic функция **maplemex**, будет заменена на новую функ-

цию с тем же именем, но уже привязанную к ядру системы Maple 9. При этом старая версия **maplemex** будет сохранена. Все остальные функции Symbolic Math Toolbox при этом не меняются, поэтому все преимущества, которые может извлечь пользователь из такой замены, заключаются лишь в том достаточно небольшом приросте возможностей Maple, который отличает две следующие друг за другом версии этой системы. Значительно больший смысл имеет использование Mathematica Symbolic Math Toolbox в том случае, если на компьютере пользователя установлены системы MATLAB и Maple 9, но не установлен пакет Symbolic Math Toolbox.

И наконец выбор последних двух пунктов настроит пакет Mathematica Symbolic Math Toolbox на работу с вычислительными ядрами систем Mathematica 5.x и Mathematica 4.x соответственно. При этом в случае если на компьютере уже установлен стандартный Symbolic Math Toolbox, будут заменены практически все файлы в каталогах \toolbox\symbolic и \toolbox\symbolic\@sym. Старые файлы будут сохранены в каталоге \toolbox\symbolic\tmp и легко могут быть восстановлены при переключении на работу со стандартным пакетом Symbolic Math Toolbox.

Работая в системе MATLAB, пользователь всегда может узнать, какое символьное ядро используется в данный момент.

```
>> symbengine ( 'info' )

Mathematica 5 in D:\Math42
```

В отношении набора функций пакет Mathematica Symbolic Math Toolbox почти полностью повторяет стандартный пакет символьных вычислений. В нем так же, как и в Symbolic Math Toolbox, содержатся функции, дающие возможность проводить вычисления в арифметике произвольной точности, функции математического анализа и линейной алгебры, функции, осуществляющие формальные преобразования и упрощения математических выражений, функции, позволяющие находить решения алгебраических и дифференциальных уравнений, а также функции двумерной и трехмерной графики. Однако в пакете есть и новые функции, такие как **install** (загрузка функции на языке Mathematica), **math2latex** (преобразование математического выражения в формат LaTeX), **math2mathml** (преобразование математического выражения в формат MathML), **mplot** (построение двумерного графика функции в формате системы Mathematica) и т. д.

При помощи конструктора **syms** создадим переменную символьного типа.

```
>> syms x
```

Вычислим в символьной форме значение определенного интеграла $\int_0^1 \frac{\ln(1-x)\ln(1-x^2)}{x^2}dx$. Следует отметить, что стандартный пакет Symbolic Math Toolbox с этой задачей не справляется.

```
>> y = int((log(1 - x)*log(1 - x^2))/x^2, x, 0, 1)

y =
        (pi^2 - 4*log(2)^2)/4
```

Напечатаем полученное выражение в естественной форме записи.

```
>> pretty( y )

ans =
        2         2
       pi  - 4 log(2)
       ---------------
       4
```

Тот же самый результат можно получить, обратившись напрямую к ядру системы Mathematica с помощью функции **mathematica**, заменившей в новом пакете функцию **maple**. При этом, разумеется, следует пользоваться правилами записи команд, принятыми в системе Mathematica.

```
>> mathematica('Integrate[(Log[1-x]* ...
    Log[1- x^2])/x^2, {x, 0, 1}]')

ans =

 2         2
Pi  - 4 Log[2]
--------------
     4
```

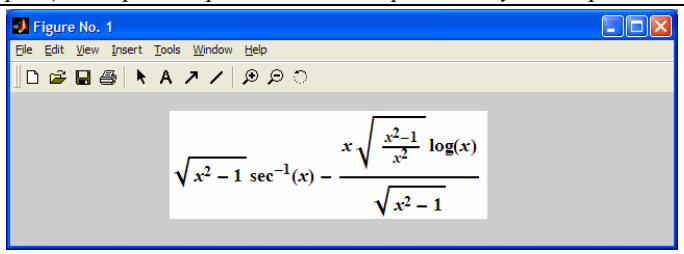
Чтобы результат, возвращаемый командой **mathematica**, можно было бы использовать в последующих вычислениях, его необходимо преобразовать в формат, принятый в Symbolic Math Toolbox. Это можно сделать с помощью опции **-input**.

```
>> mathematica('Integrate[(Log[1-x]* ...
    Log[1-x^2])/x^2, {x, 0, 1}]', '-input')

ans =
        (pi^2 - 4*log(2)^2)/4
```

Громоздкое математическое выражение значительно нагляднее в естественной форме записи, чем в текстовом или псевдографическом представлении. Команда **mathematica** с опцией **-traditional** возвращает графическое окно, содержащее изображение соответствующего математического выражения.

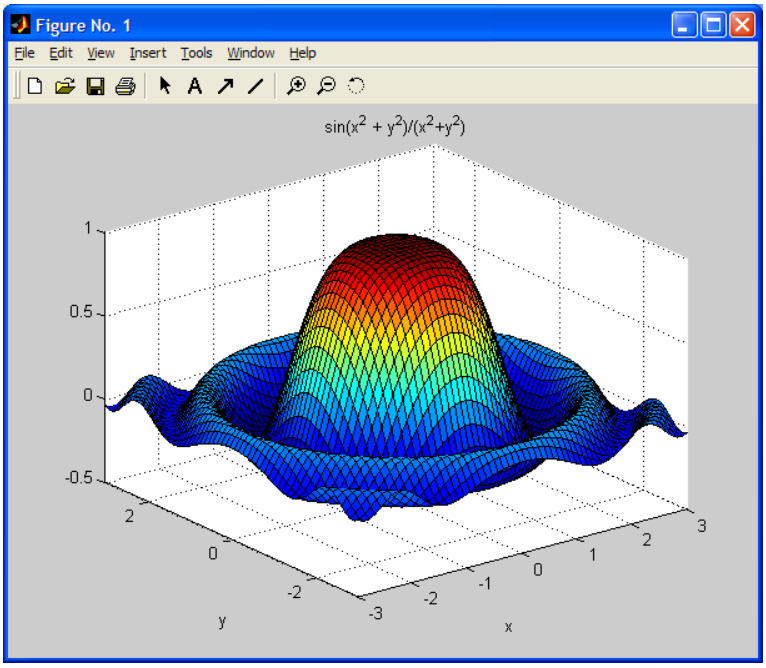
```
>>                    mathematica('Integrate[(x*ArcSec[x])/                ...
    Sqrt[x^2-1], x]', '-traditional')
```



Построить график функции в пакете Mathematica Symbolic Math Toolbox можно двумя способами.

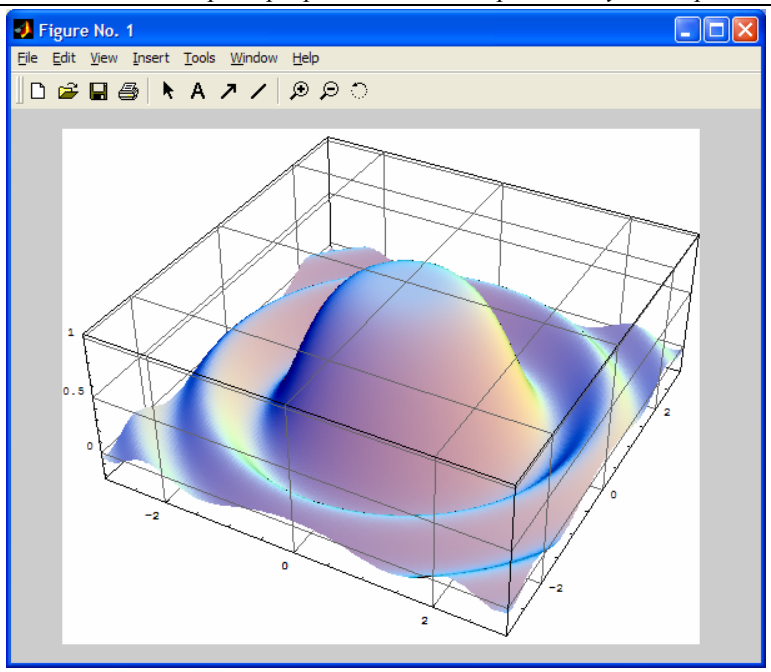
Во-первых, можно воспользоваться привычными функциями **ezplot**, **ezsurf** и т. п. В этом случае ядро системы Mathematica будет использоваться только для вычисления значений функций в точках решетки. Построение самого графика будет осуществляться средствами MATLAB.

```
>> ezsurf('sin(x^2 + y^2)/(x^2+y^2)', ...
    [-3,3], [-3,3])
```



Во-вторых, можно воспользоваться новыми функциями **mplot**, **mplot3d**, **mdisplay** и т. п., которые позволяют размещать в графическом окне MATLAB, изображения, порожденные соответствующими командами системы Mathematica. При этом пользователь может указать в качестве дополнительных аргументов таких функций любые допустимые в системе Mathematica графические опции.

```
>> mplot( 'sin(x^2 + y^2)/(x^2 + y^2)', ...
    [-3,3], [-3,3],'plotpoints', '200', 'mesh',...          'false', 'facegrids', 'All')
```



В результате пользователь, располагающий системами MATLAB и Mathematica, получает возможность создавать приложения, снабженные графическим интерфейсом допускающим как отображение математических формул в естественной форме записи, так и любой сколь угодно сложной графики построенной средствами системы Mathematica.

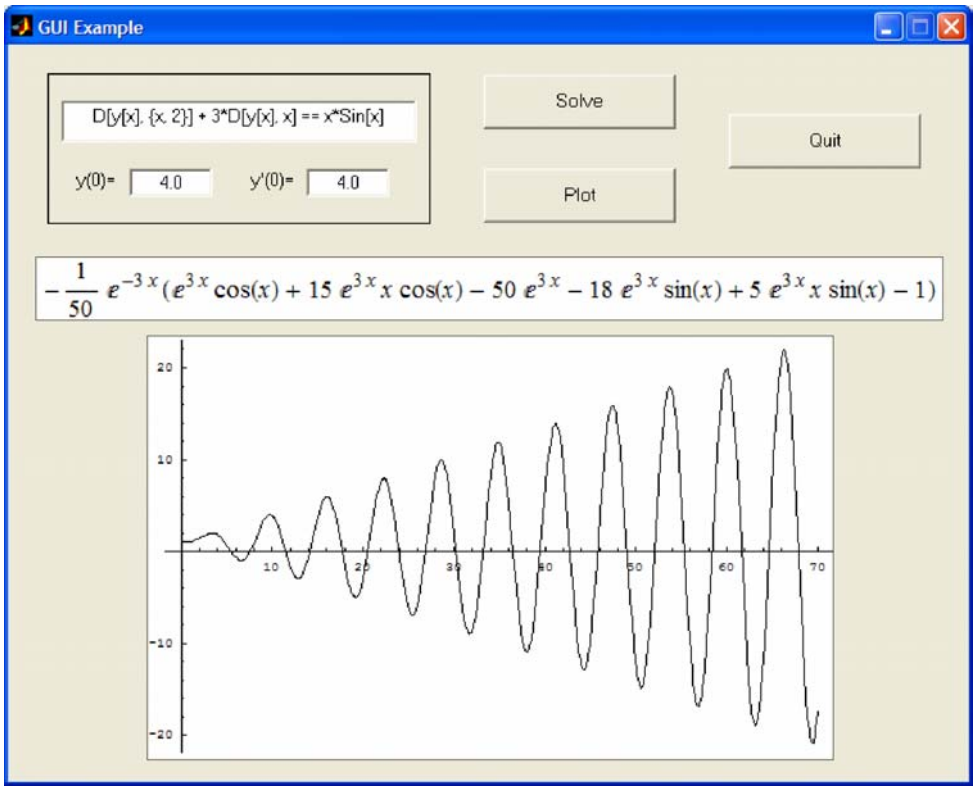


Рис. 1. Пример приложения, использующего Mathematica Symbolic Math Toolbox.

Для доступа к справочной системе Mathematica используется команда **mhelp**.

>> mhelp('Integrate')

Integrate[f, x] gives the indefinite integral of f with respect to x.

Integrate[f, {x, xmin, xmax}] gives the definite integral of f with respect to x from xmin to xmax.

Integrate[f, {x, xmin, xmax}, {y, ymin, ymax}] gives a multiple definite integral of f with respect to x and y.

Attributes[Integrate] = {Protected, ReadProtected}
Options[Integrate] := {Assumptions :> $Assumptions, GenerateConditions -> Automatic, PrincipalValue -> False}

Чтобы получить подробную информацию о какой-либо команде системы Mathematica, достаточно при вызове функции **mhelp** использовать опцию —**full**.

>> mhelp('Integrate', '-full')

В этом случае соответствующая страница справки системы Mathematica преобразуется в HTML и выводится в стандартное окно браузера справочной системы MATLAB.

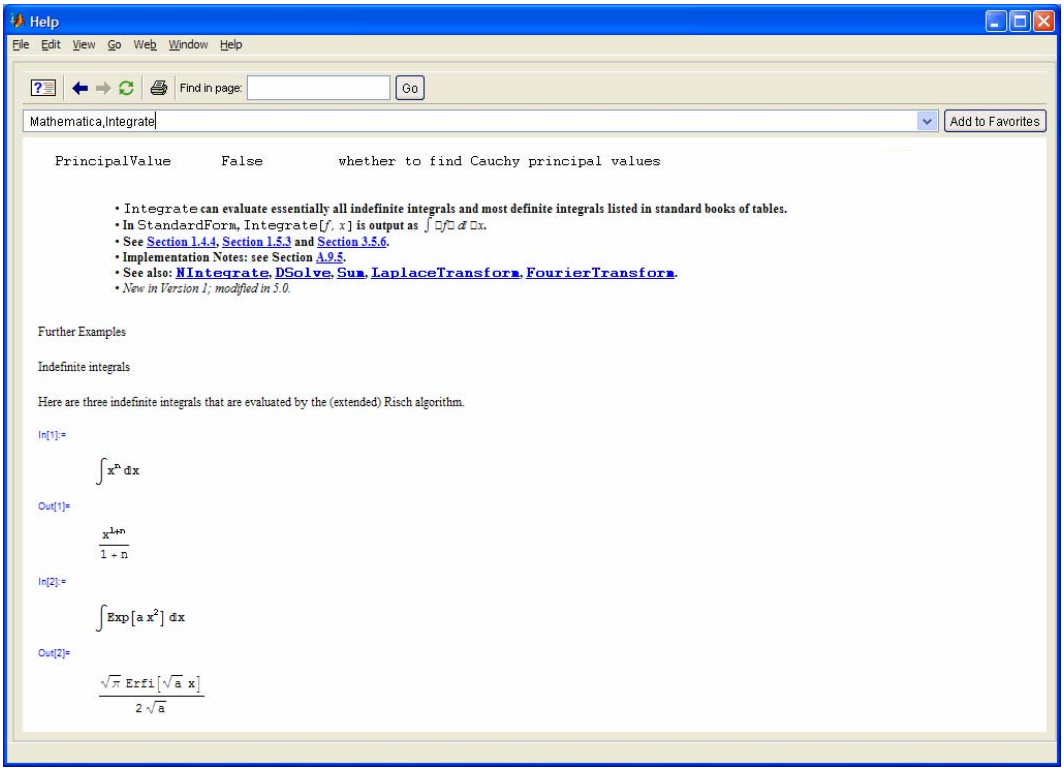


Рис. 1. Страница справки, соответствующая функции Integrate.

В настоящее время пакет Mathematica Symbolic Math Toolbox состоит из 83 основных и более чем 200 служебных функций, часть из которых

реализована на языках C, Java и Perl. Документация к пакету находится в стадии разработки. В обозримом будущем планируется ввести в состав пакета независимую от систем Maple и Mathematica символьную библиотеку, которая обладала бы большей частью возможностей Symbolic Math Toolbox (разумеется, без учета всего многообразия функций Extended Symbolic Math Toolbox и скорее всего без возможности символьного интегрирования) и допускала бы компиляцию MATLAB-приложений, использующих формальные математические преобразования, в независимо исполняемые приложения.

Литература

1. Symbolic Math Toolbox User's Guide.— MA.: MathWorks, Inc., 2002.— 262 p.
2. *Wolfram S.* The Mathematica Book, Fifth Edition.— Wolfram Media, Inc., 2003.— 1488 p.
3. *Monagan M. B., Geddes K. O., Heal K. M., Labahn G., Vorkoetter S. M., McCarron J., DeMarco P.* Maple 9 Advanced Programming Guide.— Waterloo Maple Inc., 2003.— 456 p.

УДК 338.27

ДЕЛЬТА-НОРМАЛЬНЫЙ МЕТОД РАСЧЕТА ПОКАЗАТЕЛЕЙ VAR ФИНАНСОВЫХ ИНСТРУМЕНТОВ

Сергеев С. А.
Бийский технологический институт (филиал) АлтГТУ, Бийск,
e-mail:netserg@mail.biysk.ru

В финансовых и других областях не редко встречаются ситуации неопределенности. Мерой неопределенности в финансовом риск-менеджменте выступает такая категория как риск, которая может быть выражена с помощью количественных показателей. Под риском преимущественно понимается возможность потери части ресурсов, недополучения доходов или появление дополнительных незапланированных расходов в результате осуществления предпринимательской либо спекулятивной деятельности. Одной из мерой риска стал такой показатель как VaR (Value-at-risk) [1,3].

VaR — это выраженная в денежных единицах (базовой валюте) оценка величины, которую не превысят ожидаемые в течение данного периода времени потери с заданной вероятностью.

Величина VaR для портфеля заданной структуры — это наибольший ожидаемый убыток, обусловленный колебанием цен на финансовых рынках.

В основе расчета VaR дельта-нормальным методом лежит предположение о нормальном законе распределения логарифмических доходностей факторов рыночного риска, характеризуемых в данном случае ценами финансовых инструментов.

$$r_t = \ln\left(P_t / P_{t-1}\right) \sim N\left(\mu, \sigma^2\right).$$

Для отдельной позиции портфеля финансовых инструментов, подверженных только одному фактору риска величину VaR принято рассчитывать по следующей формуле:

$$VaR = k_{1-\alpha} V \sigma_t \sqrt{T},$$

где $\left(1-\alpha\right)$ — доверительный интервал, $k_{1-\alpha}$ — квантиль нормального распределения, соответствующая заданному доверительному интервалу; V — текущая стоимость портфеля, T — временной горизонт.

Ключевыми параметрами расчета VaR являются доверительный интервал и временной горизонт.

Доверительный интервал — это вероятность того, что потери данного портфеля не превысят рассчитываемое значение VaR. Он выбирается исходя из предпочтений по риску, требований регулирующих органов и

корпоративной практики (чаще всего используют его равным 95%.) Например, компания Microsoft в своей финансовой отчетности использует VaR равный 97,5%. Базельский комитет по банковскому надзору рекомендует уровень равный 99%.

Временной горизонт — период времени, для которого производится расчет. Он обычно выбирается исходя из стратегии управления портфелем или минимального срока, в течение которого можно реализовать портфель на рынке без существенного ущерба. Например, компания Nokia использует временной горизонт равный неделе, компания Microsoft для вычисления своего рыночного риска применяет его равным 20 дням. Базельский комитет по банковскому надзору рекомендует уровень равный 10 дням.

Волатильность — мера изменчивости доходности финансового инструмента или портфеля. В качестве показателя волатильности используется стандартное отклонение случайной величины, характеризующей доходность финансового инструмента. Рассмотрим два метода оценки волатильности: метод простой скользящей средней и метод экспоненциально взвешенной скользящей средней VaR.

Метод простой скользящей средней (equally weighted moving average) — метод расчета волатильности через выборочное стандартное отклонение, в котором каждому наблюдаемому значению отклонения доходности от средневзвешенной доходности приписывается одинаковый вес:

$$\sigma = \sqrt{\frac{1}{(n-1)}\sum_{i=1}^{n}(r_i - \overline{r})^2},$$

где $\overline{r} = \frac{1}{n}\sum_{i=1}^{n} r_i$.

Рассмотрим применение данного метода на примере расчета VaR для индексов РТС, NASDAQ, S&P 500.

На рис. 1 изображен временной ряд значений индексов РТС.

На рис. 2 представлены значения VaR для индексов РТС, рассчитанные с применением метода простой скользящей средней при доверительном интервале равном 99% и временном горизонте равном 1 дню.

Очевидно, что ряд значений изменений индекса РТС превосходит установленный VaR (рис. 3). Также следует отметить, что данный метод на спокойном рынке завышает значения VaR из-за того, что на стрессовых участках волатильность увеличивается и в дальнейшем на более спокойных — продолжает по-прежнему сохранять высокие значения.

Количество случаев превышения изменения индекса РТС над VaR возрастает во время неспокойного рынка (рис. 4.), что свидетельствует о том, что применение данной модели вычисления меры риска VaR ограничено для использования на таком еще не устоявшемся рынке как Россия.

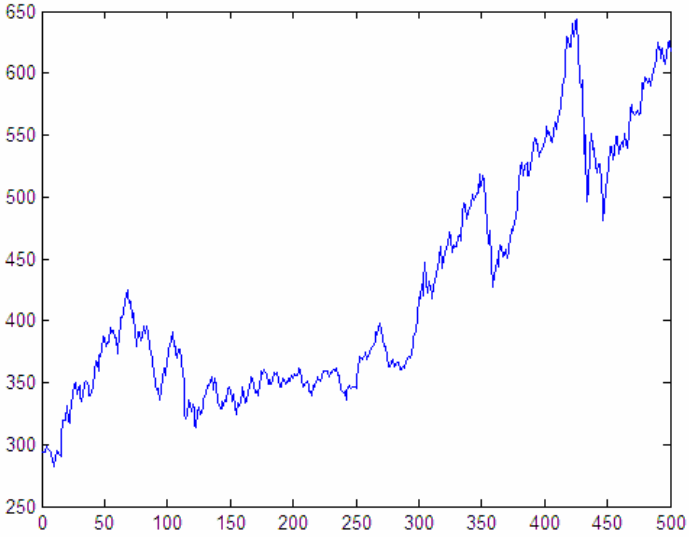


Рис. 1. Временной ряд значений индексов РТС.

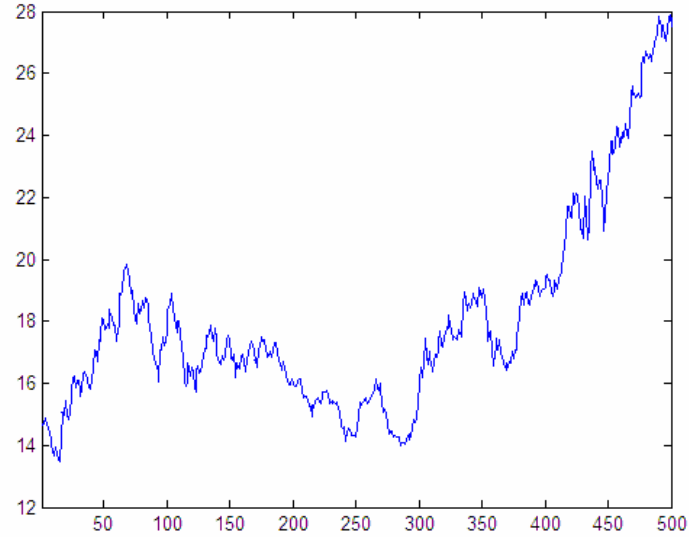


Рис. 2. Значения VaR для индексов РТС.

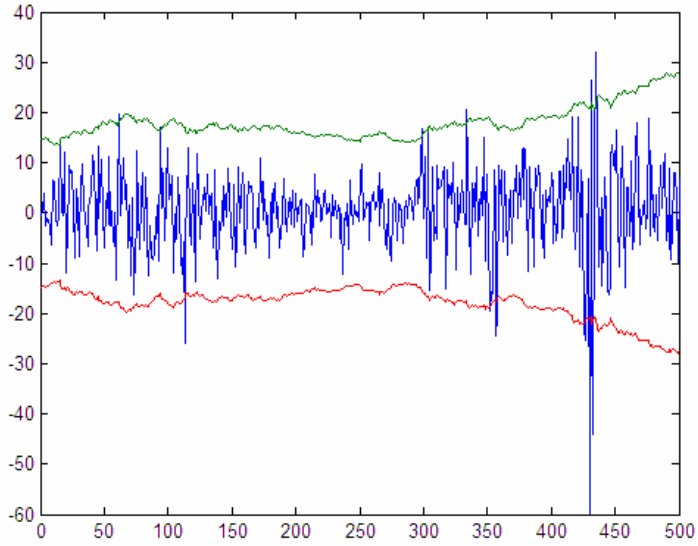


Рис. 3. Прирост индекса РТС и уровни VaR.

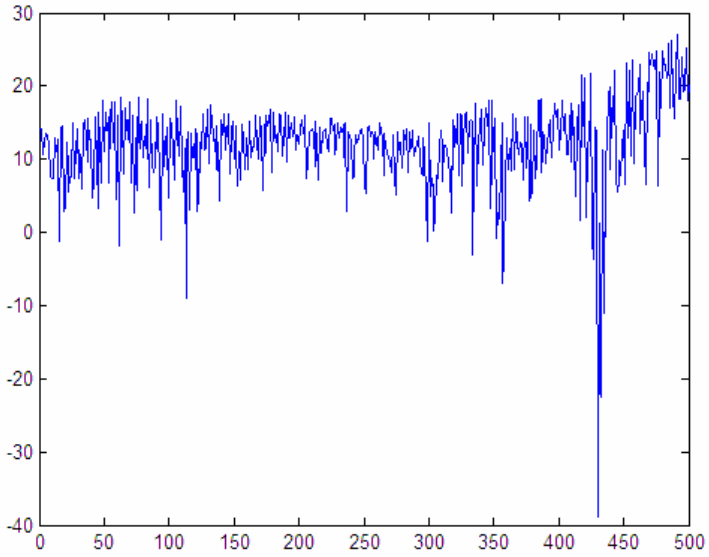


Рис. 4. Отклонение от VaR.

На рис. 5 изображен временной ряд значений индексов NASDAQ, характеризующий динамику рынка высокотехнологичных компаний.

На рис. 6 представлены значения VaR для индексов NASDAQ. У данного рынка наблюдается своя характерная особенность — превышения изменений индекса NASDAQ над VaR не является в данном конкретном частном случае опасным для участника рынка, так как данное явление наблюдается в верхней части рис. 7. Это означает, что участники рынка не несут потери, а наоборот получают «сверхприбыль».

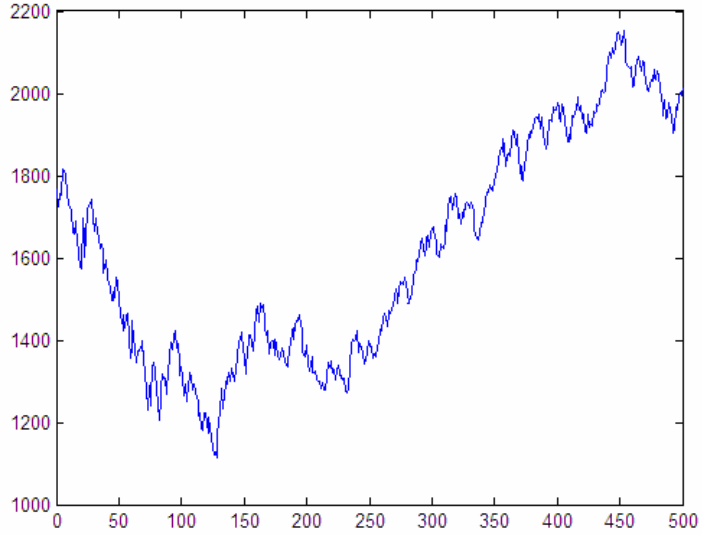


Рис. 5. Индекс NASDAQ.

Количество случаев превышения изменения индекса NASDAQ над VaR (рис. 8) относительно мало по сравнению с рассмотренным выше случаем с индексом PTC, что свидетельствует о более устойчивом, а значит и более предсказуемом рынке. Отсутствие случаев превышения отрицательного изменения индекса NASDAQ над VaR указывает на то, что за последние два года применение данной методики расчета VaR на данном рынке не привело бы к неожиданным потерям участниками рынка.

На рис. 9 изображен временной ряд значений индексов S&P500.

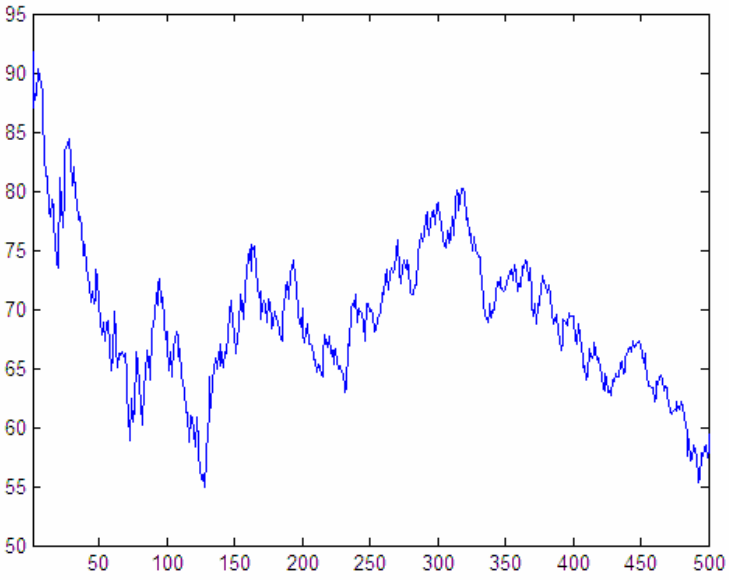


Рис. 6. Значения VaR для индексов NASDAQ.

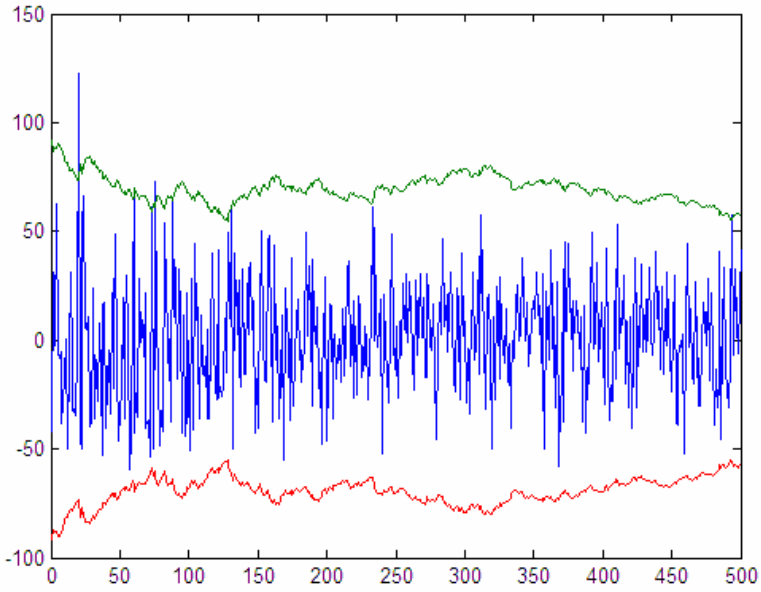


Рис. 7. Прирост индекса NASDAQ и уровни VaR.

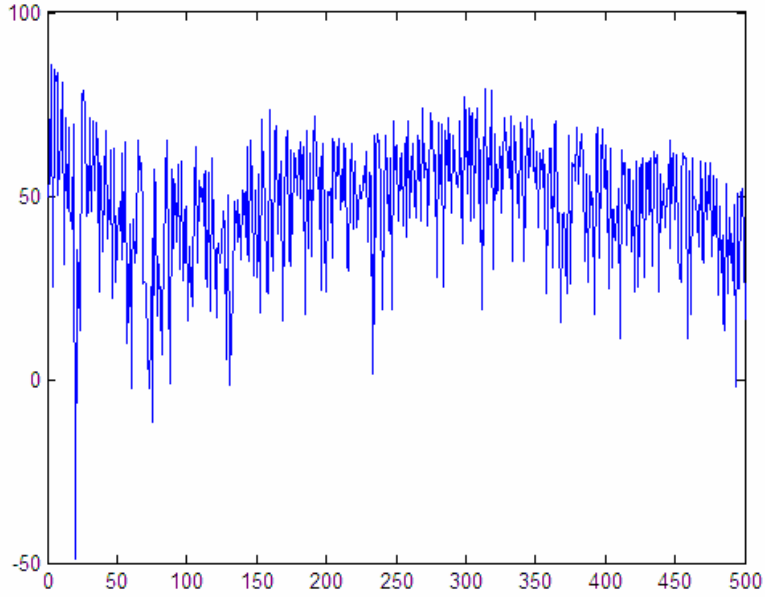


Рис. 8. Отклонение от VaR.

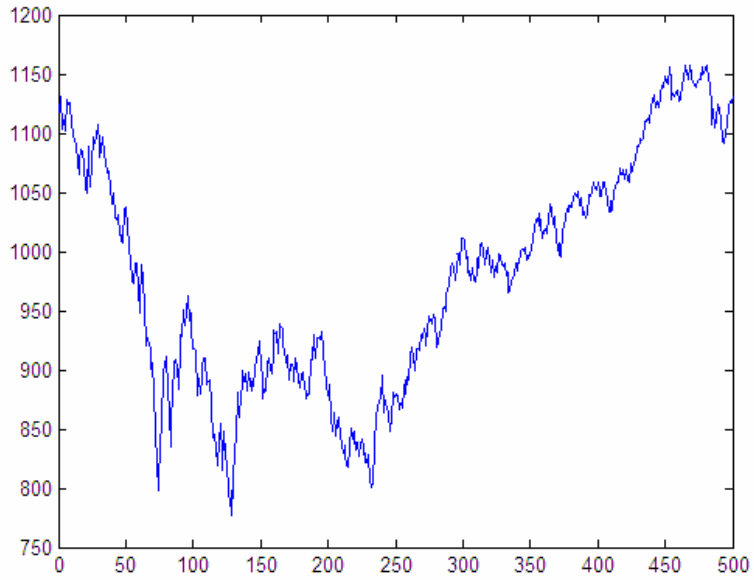


Рис. 9. Индекс S&P 500.

На рис. 10 представлены значения VaR для индексов S&P500. Как видно значения VaR на данном участке изменяется практически в два раза и к настоящему времени близок к своему минимуму, что свидетельствует о том, что в недавнем прошлом рынок имел относительно спокойную фазу. Данное утверждение подтверждается рис.11, на котором изображены прирост индекса S&P 500 и уровни VaR. Как видно из графика, в связи с нестабильностью рынка в прошлом на спокойном участке наблюдается завышение значения VaR.

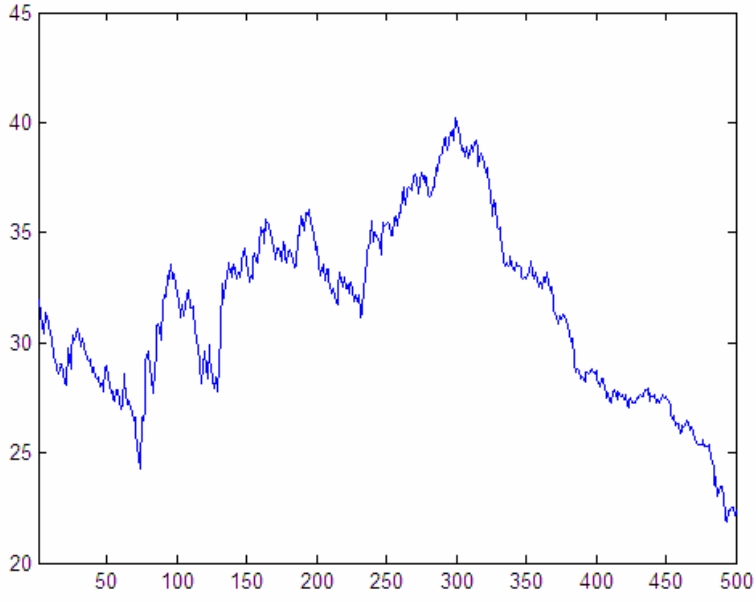


Рис. 10. Значения VaR для индекса S&P 500.

Количество случаев превышения изменения индекса S&P500 над VaR существенно велико на начальном этапе (рис. 12). Однако эта зона сменяется зоной, в которой превышения не наблюдается, что объясняется снижением волатильности рынка и завышением значения VaR, которое уже отмечалось выше.

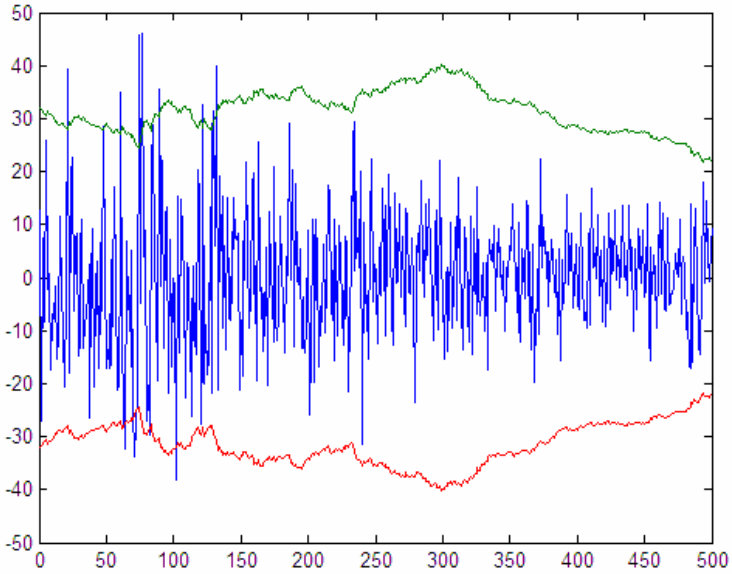


Рис. 11. Прирост индекса S&P 500 и уровни VaR.

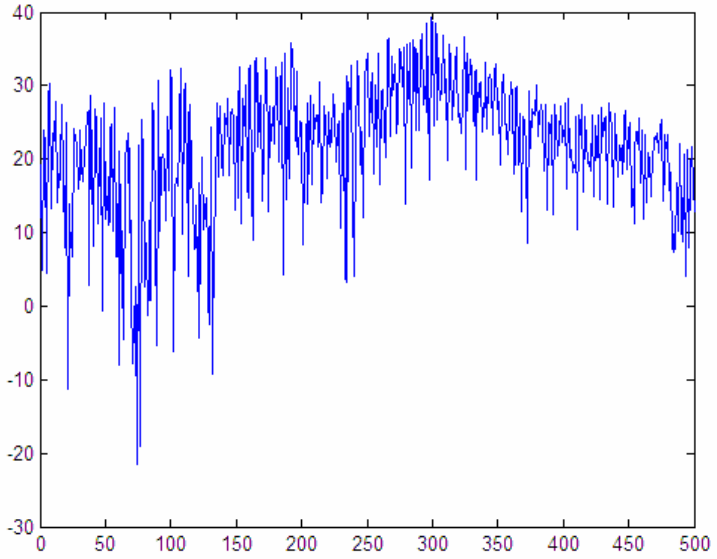


Рис. 12. Отклонение от VaR.

Метод экспоненциально взвешенной скользящей средней (exponentially weighted moving average) — метод расчета волатильности, аналогичный методу, в котором более ранние наблюдения имеют более высокий

вес. Данный метод позволяет учитывать динамику изменения волатильности:

$$\sigma_t = \sqrt{\frac{\sum_{i=1}^{n} \lambda^{i-1} (r_{t-i} - \overline{r})^2}{\sum_{i=1}^{n} \lambda^{i-1}}},$$

где λ — параметр сглаживания.

Также для расчета VaR возможно применение метода GARCH (обобщенная авторегрессионая условная гетероскедастичность) - метод оценки волатильности на основе предыдущих оценок волатильности и наблюдаемых значений отклонения доходности от средневзвешенной доходности [2]:

$$\sigma_t^2 = \alpha + \sum_{i-1}^{P} \beta_i \cdot \sigma^2{}_{t-i} + \sum_{j=1}^{Q} \gamma_j \cdot r_{i-j}^2.$$

Гетероскедастичность — дисперсия остатков не является постоянной.

Авторегрессионный процесс — процесс, при котором значение ряда находится в линейной зависимости от предыдущих значений.

Условная дисперсия — это дисперсия случайной переменной, обусловленная информацией о других случайных переменных.

Рассмотрим модель GARCH(1,1), которая описывается следующим уравнением:

$$\sigma_t^2 = \alpha + \beta \cdot \sigma^2{}_{t-1} + \gamma \cdot r_{i-1}^2.$$

Покажем применение данного метода также на примере расчета VaR для индексов РТС, NASDAQ, S&P 500.

На рис. 13 показаны значения VaR для индекса РТС. Кривая VaR имеет характерные пики, которые образованы вследствие резких колебаний рынка.

Очевидно, что на стрессовых участках рынка метод, основанный на GARCH волатильности более чувствителен к резкому изменению индекса РТС (рис. 14).

Количество случаев превышения изменения индекса РТС над VaR все равно достаточно велико (рис. 15), что свидетельствует о том, что применение данной модели не снимает проблему более точного вычисления меры риска VaR и ограничено для использования на таком нестабильном рынке как Россия.

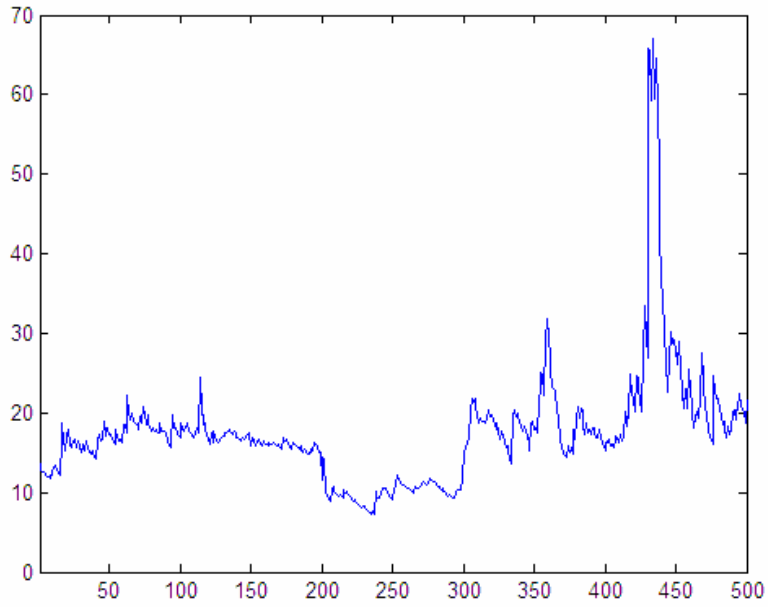


Рис. 13. Значения VaR для индекса РТС.

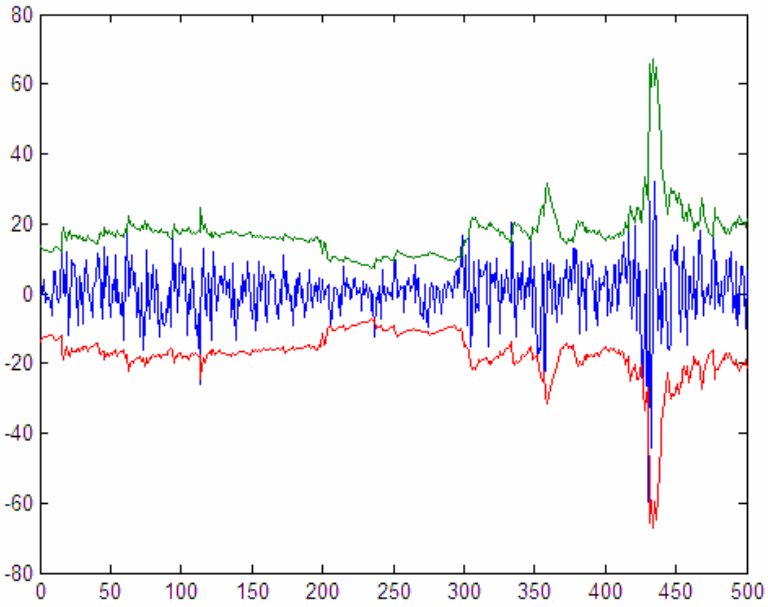


Рис. 14. Прирост индекса РТС и уровни VaR.

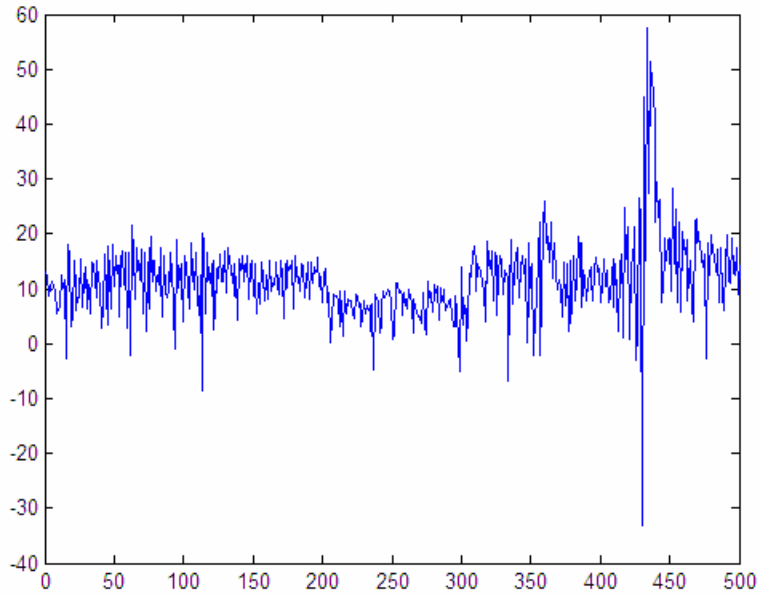


Рис. 15. Отклонение от VaR.

На рис. 16 показаны значения VaR для индекса NASDAQ. Кривая VaR также имеет пики, которые образованы вследствие колебаний рынка.

На рис. 17 наблюдается единственный случай превышения отрицательного изменения индекса NASDAQ над VaR, в то время как все остальные находятся в положительной зоне.

Самые максимальные случаи превышения изменения индекса NASDAQ над VaR (рис. 18) приходятся на резкие взлеты рынка.

На рис. 19 представлены значения VaR для индексов S&P500. Как видно значения VaR на данном участке изменяется практически в четыре раза из-за резких всплесков в прошлом и имеет тенденцию к снижению. Данное утверждение подтверждается рис. 20, на котором изображены прирост индекса S&P 500 и уровни VaR. Как видно из графика, значения VaR на спокойных участках постепенно уменьшается, но это не мешает отлавливать случаи относительных резких изменений.

Количество случаев превышения изменения индекса S&P500 над VaR велико (рис. 21), однако в основном находится около значения нуля.

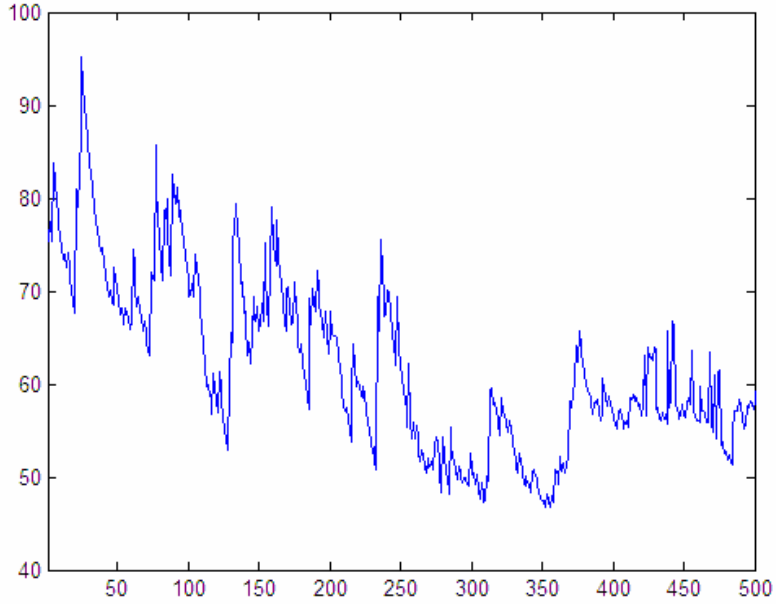


Рис. 16. Значения VaR для индекса NASDAQ.

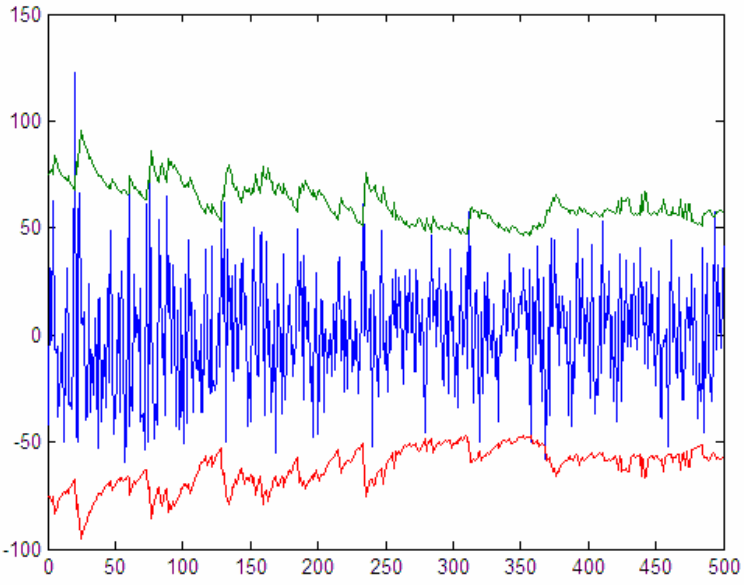


Рис. 17. Прирост индекса NASDAQ и уровни VaR.

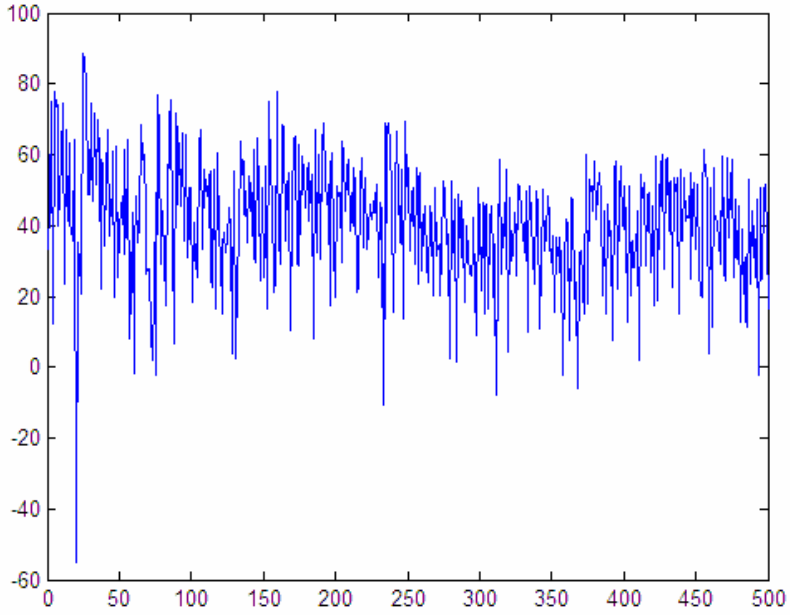


Рис. 18. Отклонение от VaR.

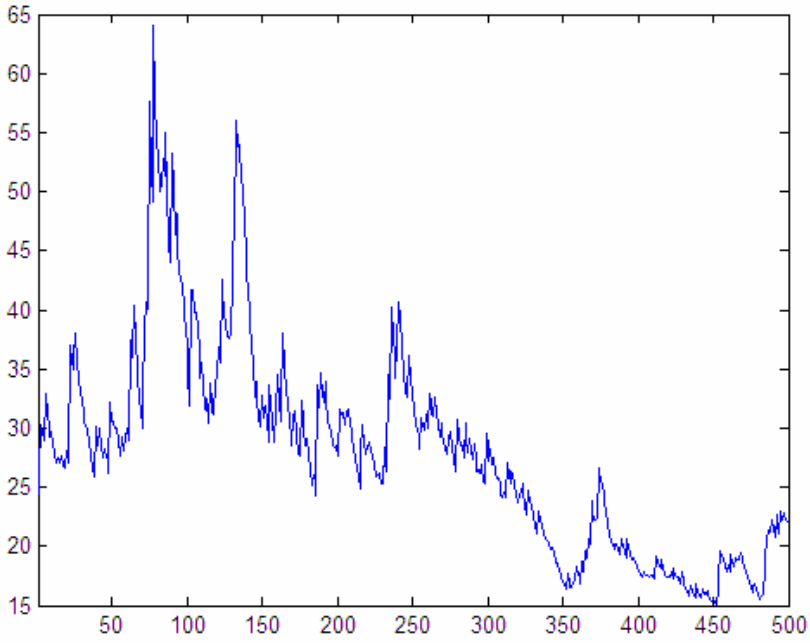


Рис. 19. Значения VaR для индекса S&P 500.

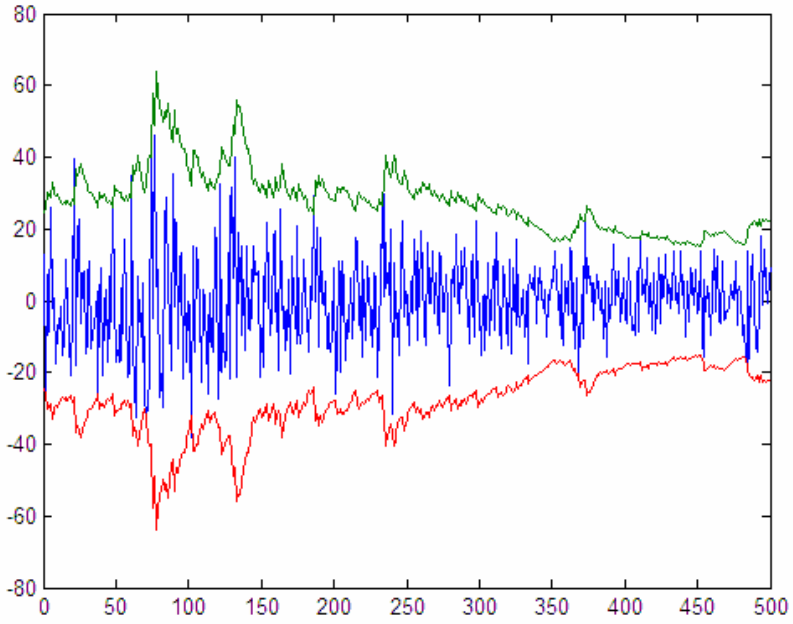


Рис. 20. Прирост индекса S&P 500 и уровни VaR.

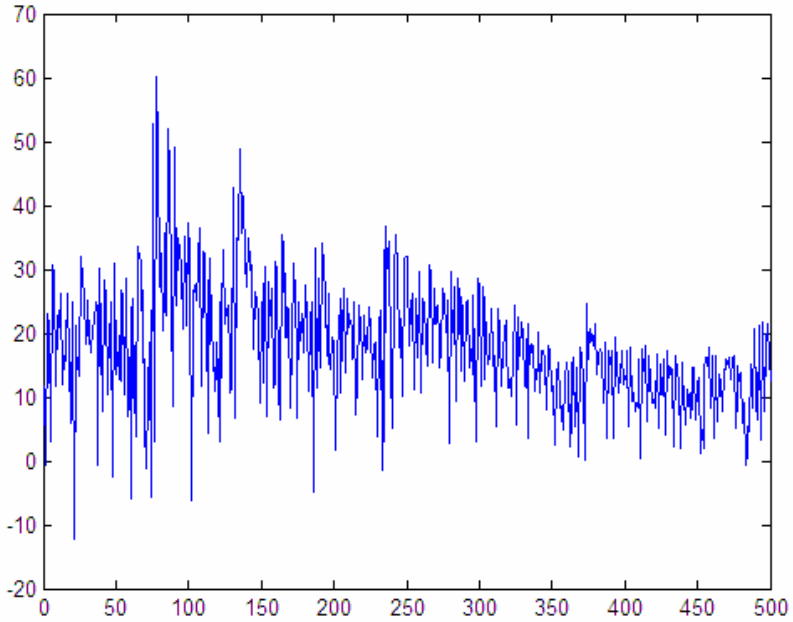


Рис. 21. Отклонение от VaR.

Для расчета показателя VaR портфеля для финансовых инструментов используют следующую формулу

$$VaR_{порт} = k_{1-\alpha}\sqrt{\mathrm{X}^T \Omega \mathrm{X} T},$$

где X — вектор-столбец финансовых инструментов в денежном выражении, составляющих портфель, Ω — ковариационная матрица изменений доходностей финансовых инструментов.

Расчет VaR реализован в пакете MATLAB. Котировки акций могут быть представлены в формате Excel.

Достоинства дельта-нормального метода:

- простота реализации;
- небольшие затраты на сбор первичны данных;
- быстрота вычислений;
- приемлемая точность оценки VaR.

Недостатки дельта-нормального метода:

- не применим к нелинейным инструментам, таким как опционы;
- VaR оказывается иногда завышенным или заниженным, так как для финансовых инструментов характерны «толстые хвосты», которые характеризуют отличие плотности вероятности от нормального распределения;
- игнорирование рисков одиночных событий;
- не учитываются какие-либо иные виды риска, кроме риска дельты.

Для расчета VaR применяются еще дельта-гамма-вега приближение, метод исторического моделирования, метод Монте-Карло. Данные методы также имеют свои «плюсы» и «минусы».

Литература

1. Энциклопедия финансового риск-менеджмента / Под ред. А. А. Лобанова и А. В. Чугунова.— М.: Альпина Паблишер, 2003.— 786 с.
2. *Уотшем Т. Дж., Паррамоу К.* Количественные методы в финансах / Пер. с англ. под ред. М. Р. Ефимовой.— М.: Финансы, ЮНИТИ, 1999.— 527 с.
3. *Morgan J. P.* RiskMetrics — Technical Document. Fourth Edition.— Reuters. NY, 1996.

УДК 004

ВИЗУАЛИЗАЦИЯ ТЕПЛОВЫХ ПОЛЕЙ ПЕЧАТНЫХ ПЛАТ С ПОМОЩЬЮ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ИНТЕЛЛЕКТУАЛЬНЫХ СЕНСОРОВ В ПАКЕТЕ FEMLAB

Сидоров О. В.
Московский государственный технический университет «МАМИ», Москва,
e-mail:sid_ov@mail.ru

В настоящее время все большее число операций управления сложными процессами возлагается на автоматические комплексы, представляющие собой большую совокупность печатных плат. Выход из строя таких комплексов может привести к техногенным катастрофам. Как правило, выходу из строя комплекса предшествует аномальный тепловой режим отдельных узлов и элементов печатных плат комплекса. Возможность диагностики и визуализации тепловых полей в реальном масштабе времени позволяет на раннем этапе возникновения нештатных ситуаций выявлять неисправности и проводить профилактические работы. Такой контроль осуществляется распределенной системой интеллектуальных сенсоров в виде пленочного композитного материала (рис1.), встроенного в печатную плату преобразующий параметр теплового поля в данной точке в электрический сигнал.

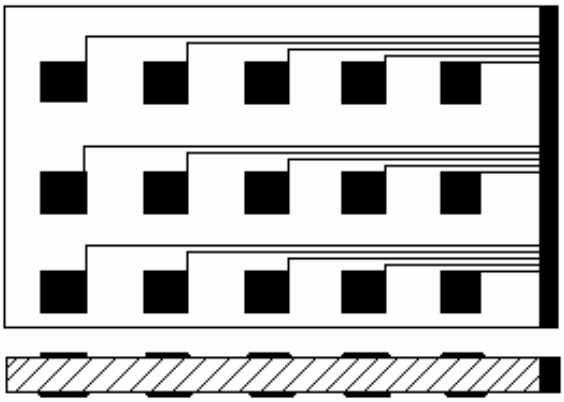


Рис. 1.

В ходе исследований на основе пакета Femlab фирмы Comsol и пакета Simulink системы MATLAB построен алгоритм обработки сигналов для цветовой визуализации тепловых динамических режимов платы с помощью встроенной распределенной системы интеллектуальных сенсоров. Необходимые для этого данные были получены в других пакетах.

Так работа электронной схемы моделировалась в пакете Multisim 2001. Полученные данные экспортировались в пакет AutoTherm корпора-

ции Mentor Graphics для дальнейшего моделирования динамических тепловых режимов самой печатной платы с находящимися на ней электронными компонентами. Здесь же, в частности, и визуализировалось динамическое распределение теплового поля платы. Моделирование теплового режима в окружающей среде при наличии конвекции и принудительных воздушных потоков производилось с помощью пакета FLOTHERM компании Flomerics Ltd. Массивы данных из этих приложений импортировались в указанное выше разработанное приложение Femlab.

Кроме того, моделирование позволило установить, что наиболее адекватное отображения тепловых режимов получается с помощью сенсоров квадратной формы.

Работа выполнена при поддержке РФФИ (грант 02-07-08006)

УДК 517

АЛГОРИТМ РЕШЕНИЯ СИСТЕМ УРАВНЕНИЙ КОЛМОГОРОВА (ОЦЕНКА КАЧЕСТВА СИСТЕМЫ ТЕХНИЧЕСКОГО ОБСЛУЖИВАНИЯ)

Сорокин А. С.
Кузбасская государственная педагогическая академия, Новокузнецк,
e-mail:andsor@mail.ru

Известно, что всякая марковская модель системы технического обслуживания описывается системой дифференциальных уравнений [1]:

$$\frac{dP_k(t)}{dt} = \sum_{i=1}^{n} \lambda_{ki} P_i(t), \quad k = \overline{1,n}. \qquad (*)$$

В системе (*) исключая последовательно функции $P_i(t)$, $i = \overline{2,n}$, получаем уравнение резольвенты для функции $P_1(t)$:

$$\sum_{k=0}^{n} a_k \frac{d^k P_1(t)}{dt^k} = f(t). \qquad (**)$$

Тогда имеет место структурная формула решения уравнения (**) [2]:

$$P_1(x) = \int_{x_0}^{x} \sum_{i=1}^{n} \Big(\prod_{j=1, j\neq i}^{n} (r_i - r_j) \Big)^{-1} \exp(r_i(x-t)) f(t) dt. \qquad (***)$$

Проиллюстрируем этот алгоритм на примере, взятом из статьи Рыбалко В. В. [3].

Система дифференциальных уравнений (уравнений Колмогорова) имеет вид:

$$\begin{aligned}
\frac{dP_1(t)}{dt} &= -P_1(t)(\lambda_{14} + \lambda_{12}) + P_3(t)\mu_{31} + P_4(t)\mu_{41} + P_5(t)\mu_{51} + P_6(t)\mu_{61};\\
\frac{dP_2(t)}{dt} &= -P_2(t)(\lambda_{23} + \lambda_{24} + \lambda_{26}) + P_1(t)\lambda_{12};\\
\frac{dP_3(t)}{dt} &= -P_3(t)(\lambda_{35} + \mu_{31}) + P_2(t)\lambda_{23};\\
\frac{dP_4(t)}{dt} &= -P_4(t)(\lambda_{45} + \mu_{41}) + P_1(t)\lambda_{41} + P_2(t)\lambda_{24};\\
\frac{dP_5(t)}{dt} &= -P_5(t)\mu_{51} + P_3(t)\lambda_{35} + P_4(t)\lambda_{45};\\
\frac{dP_6(t)}{dt} &= -P_6(t)\mu_{61} + P_2(t)\lambda_{26}.
\end{aligned} \qquad (1)$$

Начальные условия: $P_1(0) = 1, \quad P_k(0) = 0, \quad k = 2,\ldots,6.$

Значения параметров приняты:

$$\lambda_1 = 0.012; \quad \lambda_{23} = 0.2; \quad \lambda_{14} = 0.02; \quad \lambda_{26} = 0.004; \quad \lambda_{24} = 0.2; \quad \lambda_{45} = 0.2; \quad \lambda_{35} = 0.0056;$$

$$\mu_{61} = 0.004;\ \ \mu_{31} = 0.05;\ \ \mu_{51} = 0.05;\ \ \mu_{41} = 0.02.$$

При этих значениях система (1) принимает вид:

$$\begin{gathered}
\frac{dP_1(t)}{dt} = -\tfrac{1}{25}P_1(t) + \tfrac{1}{20}P_3(t) + \tfrac{1}{50}P_4(t) + \tfrac{1}{20}P_5(t) + \tfrac{1}{250}P_6(t);\\
\frac{dP_2(t)}{dt} = -\tfrac{101}{250}P_2(t) + \tfrac{1}{50}P_1(t);\\
\frac{dP_3(t)}{dt} = -\tfrac{139}{2500}P_3(t) + \tfrac{1}{5}P_2(t);\\
\frac{dP_4(t)}{dt} = -\tfrac{11}{50}P_4(t) + \tfrac{1}{50}P_1(t) + \tfrac{1}{5}P_2(t);\\
\frac{dP_5(t)}{dt} = -\tfrac{1}{20}P_5(t) + \tfrac{7}{1250}P_3(t) + \tfrac{1}{5}P_4(t);\\
\frac{dP_6(t)}{dt} = -\tfrac{1}{250}P_6(t) + \tfrac{1}{250}P_2(t),\\
\sum_{i=1}^{6} P_i(t) = 1.
\end{gathered} \tag{2}$$

Решая систему (2), получаем базис Гребнера:

$$\begin{gathered}
\frac{d^5P_1(t)}{dt^5} = -\tfrac{1}{25}\frac{d^4P_1(t)}{dt^4} + \frac{d^3P_1(t)}{dt^3} + \tfrac{613}{1562500}\frac{d^2P_1(t)}{dt^2} - \tfrac{26463}{195312500}\frac{dP_1(t)}{dt} + \tfrac{16576731}{390625}10^{-6}P_1(t) -\\
-\tfrac{4790763707}{78125}10^{-8}P_2(t) + \tfrac{1}{32}10^{-5}P_3(t) - \tfrac{226447}{25}10^{-8}P_4(t) + \tfrac{1}{32}10^{-5}P_5(t) + \tfrac{1}{9765625}10^{-5}P_6(t);\\
\frac{d^4P_1(t)}{dt^4} = -\tfrac{1}{25}\frac{d^3P_1(t)}{dt^3} + \tfrac{1}{2500}\frac{d^2P_1(t)}{dt^2} + \tfrac{613}{15625}10^{-2}\frac{dP_1(t)}{dt} - \tfrac{26463}{1953125}10^{-2}P_1(t) +\\
+\tfrac{26809087}{15625}10^{-6}P_2(t) - \tfrac{1}{16}10^{-4}P_3(t) + \tfrac{10151}{25}10^{-6}P_4(t) - \tfrac{1}{16}10^{-4}P_5(t) - \tfrac{1}{390625}10^{-4}P_6(t);\\
\frac{d^3P_1(t)}{dt^3} = \tfrac{1}{25}\frac{d^2P_1(t)}{dt^2} + \tfrac{1}{2500}\frac{dP_1(t)}{dt} + \tfrac{613}{15625}10^{-2}P_1(t) - \tfrac{157579}{3125}10^{-4}P_2(t) + \tfrac{1}{8000}P_3(t) -\\
-\tfrac{433}{25}10^{-4}P_4(t) + \tfrac{1}{8000}P_5(t) + \tfrac{1}{156250}10^{-3}P_6(t);
\end{gathered} \tag{3}$$

$$\frac{d^2P_1(t)}{dt^2} = -\tfrac{1}{25}\frac{dP_1(t)}{dt} + \tfrac{1}{2500}P_1(t) + \tfrac{219}{15625}P_2(t) - \tfrac{1}{400}P_3(t) + \tfrac{7}{1250}P_4(t) - \tfrac{1}{400}P_5(t) - \tfrac{1}{62500}P_6(t);$$

$$\frac{dP_1(t)}{dt} = -\tfrac{1}{25}P_1(t) + \tfrac{1}{20}P_3(t) + \tfrac{1}{50}P_4(t) + \tfrac{1}{20}P_5(t) + \tfrac{1}{250}P_6(t);$$

$$\sum_{i=1}^{6} P_i(t) = 1.$$

Исключим из системы (3), функции $P_2(t)$, $P_3(t)$, $P_4(t)$, $P_5(t)$, $P_6(t)$, получим уравнение резольвенты для функции $P_1(t)$:

$$\frac{d^4P_1(t)}{dt^4} + \tfrac{359}{500}\frac{d^3P_1(t)}{dt^3} + \tfrac{18687}{125000}\frac{d^2P_1(t)}{dt^2} + \tfrac{14331}{1562500}\frac{dP_1(t)}{dt} + \tfrac{549}{15625000}P_1(t) - \tfrac{1111}{625}10^{-5} = 0. \tag{4}$$

Характеристическими числами (4) являются:

$$r_1 = —0.0040994088,\ r_2 = —0.2093579663,$$

$$r_3 = —0.1016014637,\ r_4 = —0.4029411611.$$

Тогда с помощью структурной формулы (***) получаем явное аналитическое решение системы (2) в виде:

$$P_1(t) = 0.01203960153\exp(r_1t) - 0.07562139618\exp(r_2t) +$$
$$+0.560571225\exp(r_3t) - 0.002909284573\exp(r_4t) + 0.5059198542,$$
$$P_2(t) = 0.0006021297187\exp(r_1t) - 0.007770304751\exp(r_2t) +$$
$$+0.03707499613\exp(r_3t) - 0.05495235853\exp(r_4t) + 0.02504553734,$$
$$P_3(t) + P_5(t) = 0.009913620227\exp(r_1t) + 0.3713891146\exp(r_2t) -$$
$$-0.7534464851\exp(r_3t) - 0.003083934132\exp(r_4t) + 0.3752276867,$$
$$P_4(t) = 0.001673075428\exp(r_1t) - 0.2881487662\exp(r_2t) +$$
$$+0.1573197126\exp(r_3t) + 0.06039459533\exp(r_4t) + 0.06876138433,$$
$$P_6(t) = -0.0242284269\exp(r_1t) + 0.0001513533208\exp(r_2t) -$$
$$-0.001519452125\exp(r_3t) + 0.0005509817391\exp(r_4t) + 0.02504553734.$$

На рис. 1 показано аналитическое решение $P_1(t)$. Точками указаны данные, полученные В. В. Рыбалко [3] с помощью пакета Mathcad.

На рис. 2 показано аналитическое решение $P_2(t)$. Точками указаны данные, полученные В.В. Рыбалко [3] с помощью пакета Mathcad.

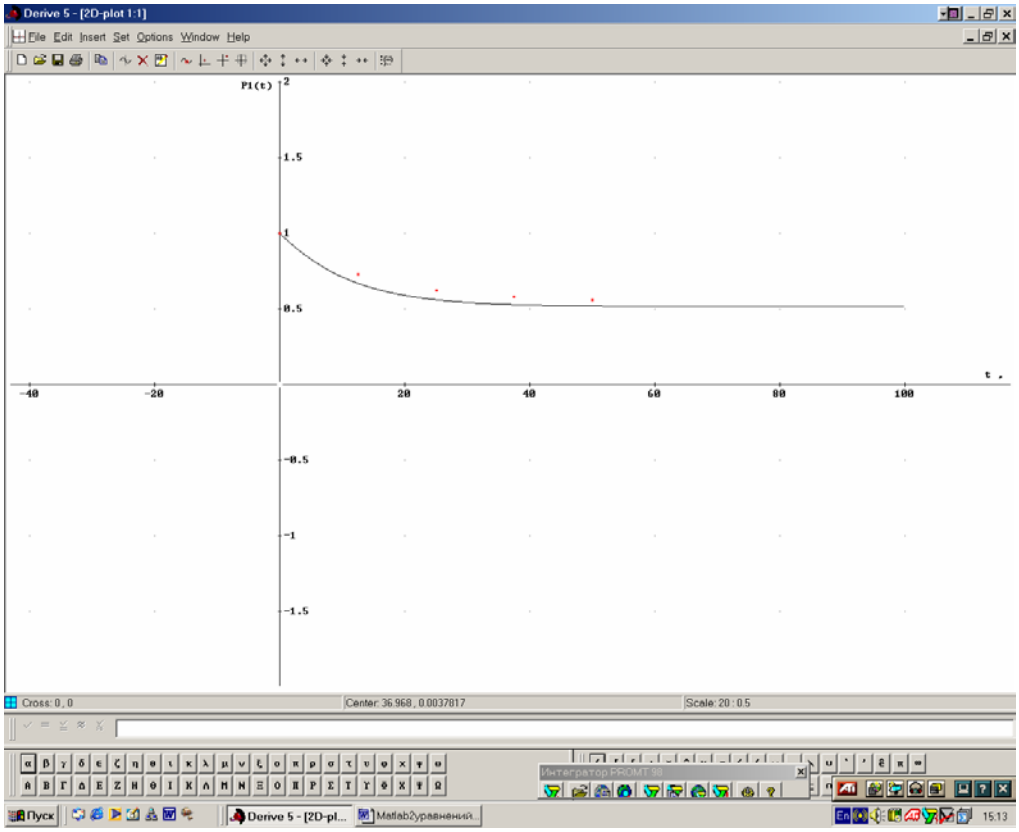


Рис. 1. Вероятность состояния $P_1(t)$.

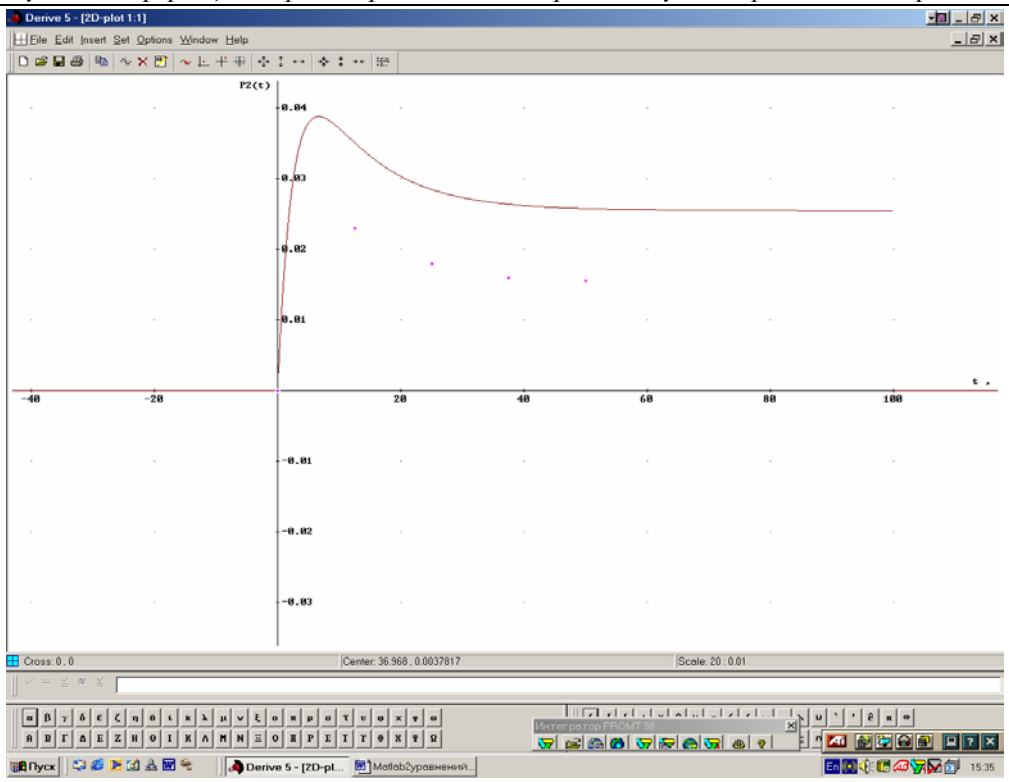


Рис. 2. Вероятность состояния $P_2(t)$.

На рис. 3 показано аналитическое решение $P_3(t)+P_5(t)$. Точками указаны данные, полученные В. В. Рыбалко [3] с помощью пакета Mathcad.

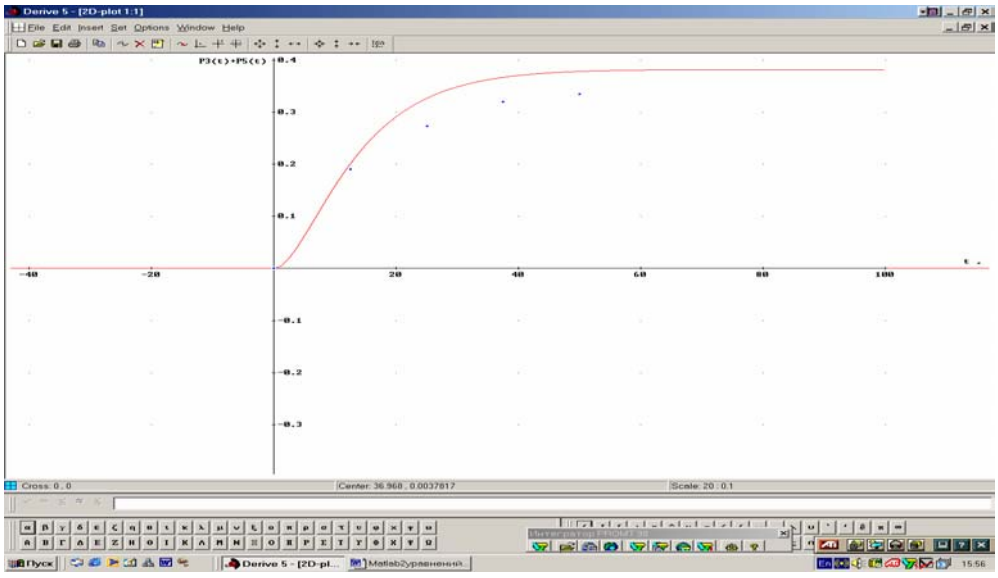


Рис. 3. Вероятность состояния $P_3(t)+P_5(t)$.

На рис. 4 показано аналитическое решение $P_4(t)$. Точками указаны данные, полученные В. В. Рыбалко [3] с помощью пакета Mathcad.

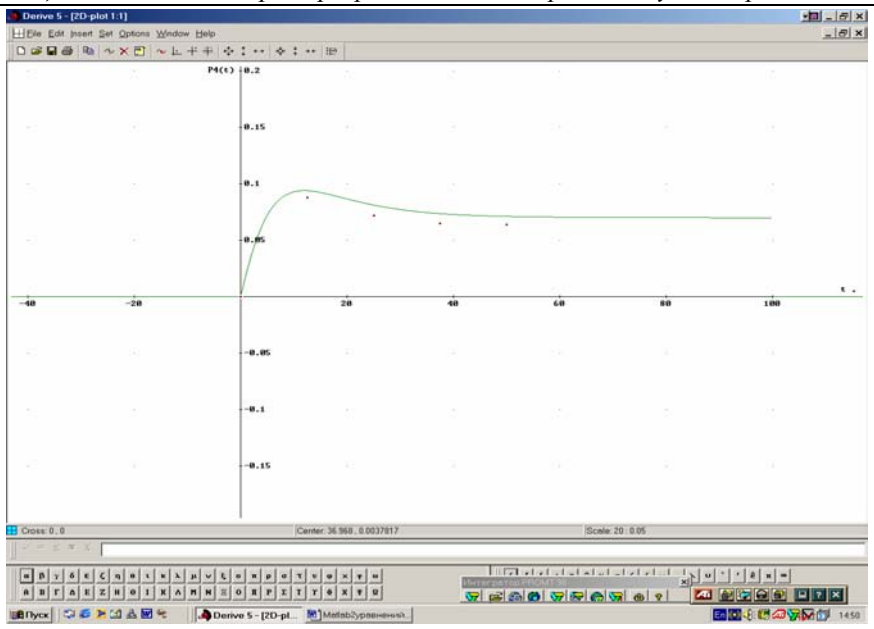


Рис. 4. Вероятность состояния $P_4(t)$.

На рис. 5 показано аналитическое решение $P_6(t)$. Точками указаны данные, полученные В. В. Рыбалко [3] с помощью пакета Mathcad.

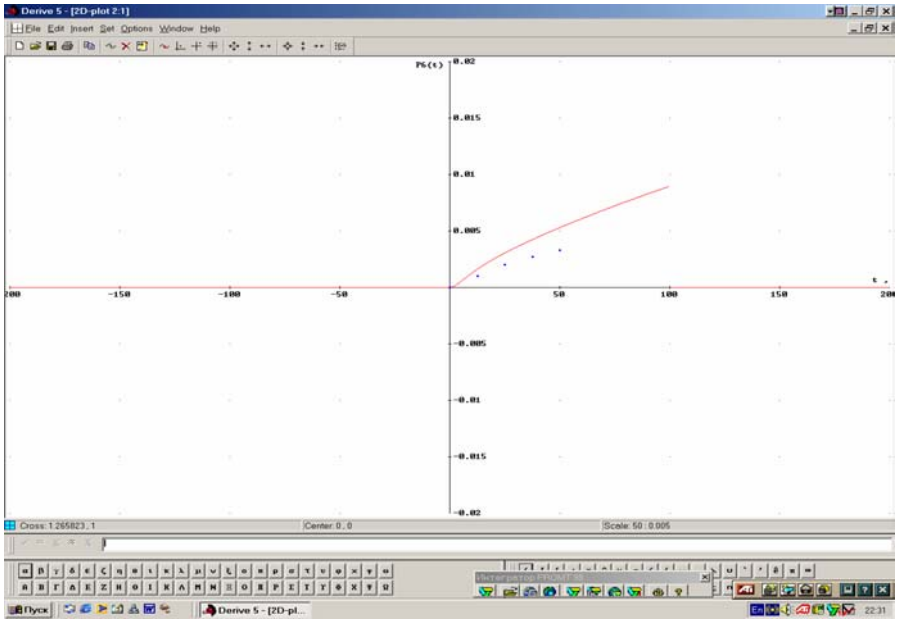


Рис. 5. Вероятность состояния $P_6(t)$.

Известно, что интеграл от вероятности работоспособного состояния $P_1(t)$, взятый по интервалу от нуля до времени, соответствующего ее стационарному значению, дает среднее время безотказной работы системы:

$$T_{cp}(t) = -2.936911666\exp(r_1 t) + 0.3612062035\exp(r_2 t) - 5.517353831\exp(r_3 t) + \\ +0.007220122573\exp(r_4 t) + 0.5059198541t + 8.085839171. \quad (5)$$

На рис. 6 показано аналитическое решение $T_{cp}(t)$.

Функционал, характеризующий качество функционирования системы, вычисляется по формуле:

$$J(t) = \frac{Q(t)}{R(t)}, \tag{6}$$

где

$$Q(t) = -2.936911666\exp(r_1 t) + 0.3612062035\exp(r_2 t) - 5.517353831\exp(r_3 t) +$$
$$+0.007220122573\exp(r_4 t) + 0.5059198541t + 8.085839171,$$

причем

$$R(t) = 2.936911666\exp(r_1 t) - 0.3612062035\exp(r_2 t) + 5.517353831\exp(r_3 t) -$$
$$-0.007220122573\exp(r_4 t) + 0.4940801458t - 8.085839171.$$

На рис. 7 показано аналитическое решение $J(t)$.

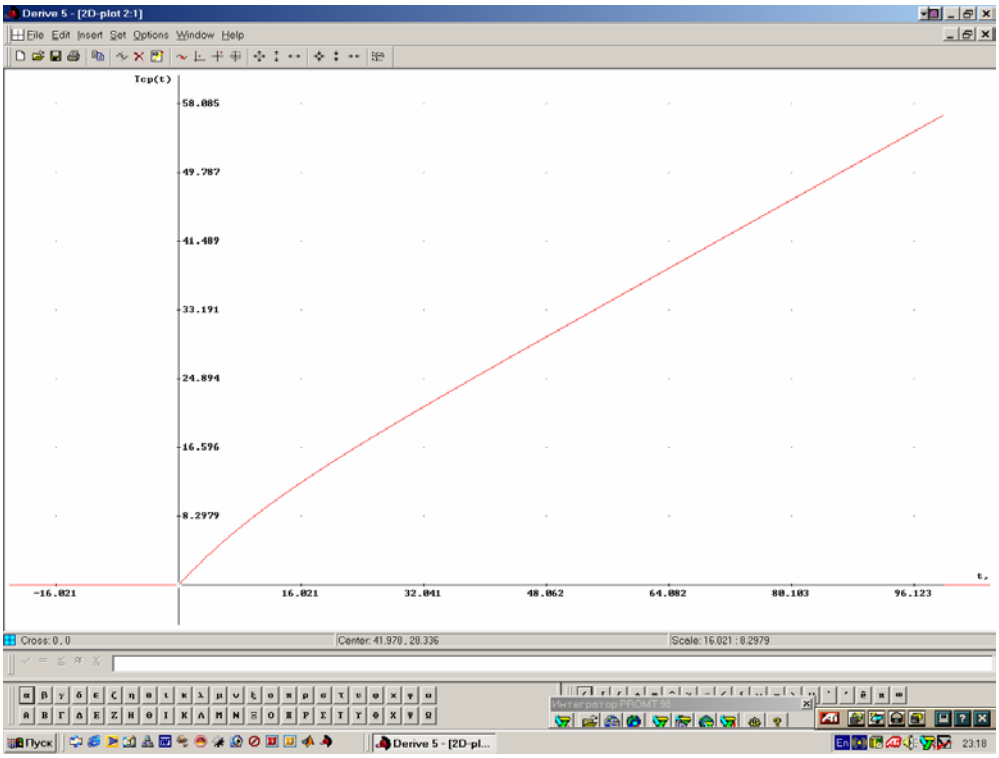


Рис. 6. Среднее время безотказной работы $T_{cp}(t)$.

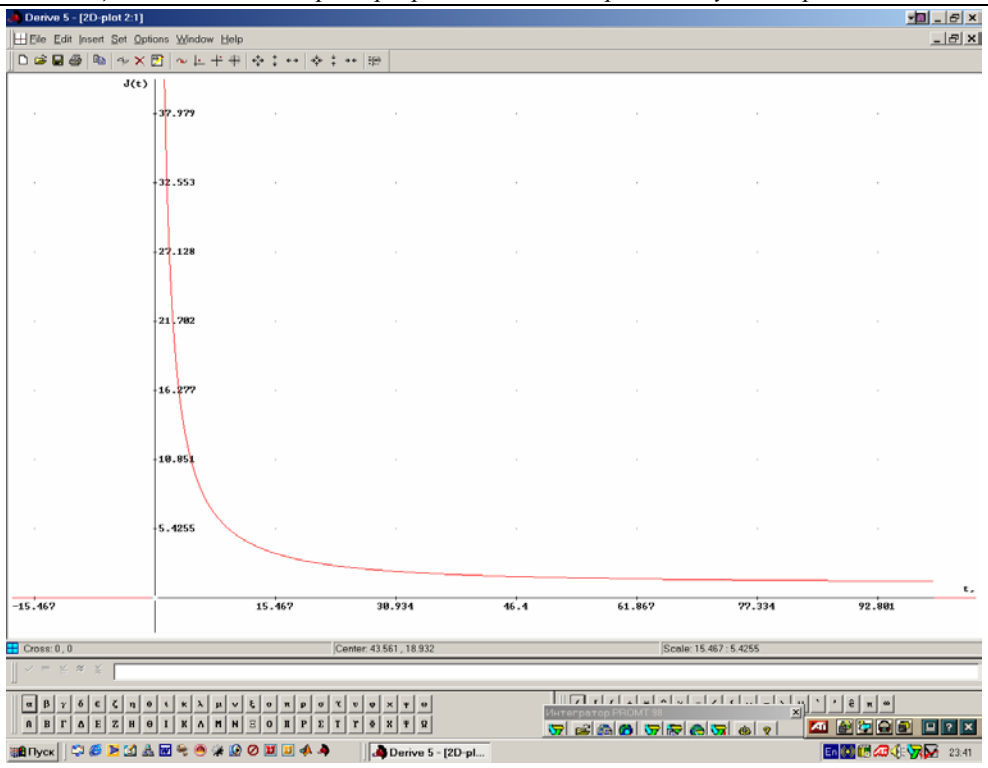


Рис. 7. Качество функционирования системы $J(t)$.

Отметим, что функционал (6), характеризующий качество функционирования системы, с помощью (5) преобразуется к виду:

$$J(t) = \frac{T_{cp}(t)}{t - T_{cp}(t)}. \qquad (7)$$

Представление (7) зависит как от среднего времени безотказной работы $T_{cp}(t)$, так и от времени работы t.

На рис. 8 показано аналитическое решение $J(t)$ в зависимости от среднего времени безотказной работы $T_{cp}(t)$ и от времени работы t.

Преобразуем функционал (7) к виду:

$$J(t) = \frac{\tau_{cp}(t)}{1 - \tau_{cp}(t)}, \qquad (8)$$

где

$$\tau_{cp}(t) = \frac{T_{cp}(t)}{t}.$$

На рис. 9 показано аналитическое решение $J(t)$ в зависимости от относительного среднего времени безотказной работы $T_{cp}(t)/t$.

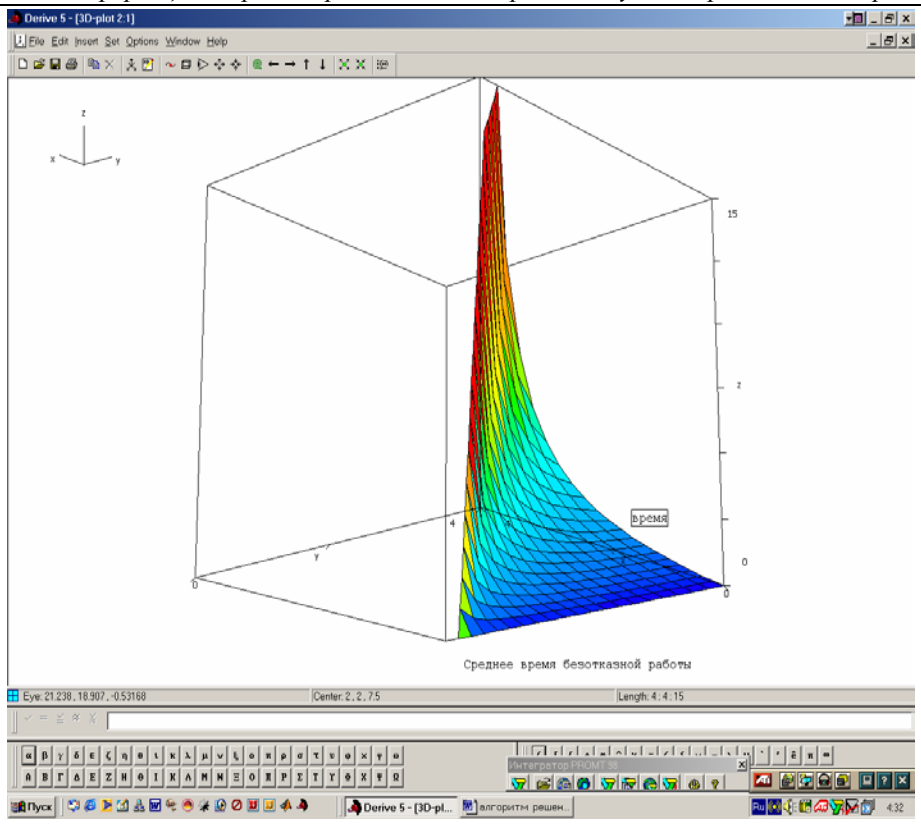


Рис. 8. Качество функционирования системы $J(t)$ как функционала среднего времени безотказной работы $T_{cp}(t)$ и времени работы t.

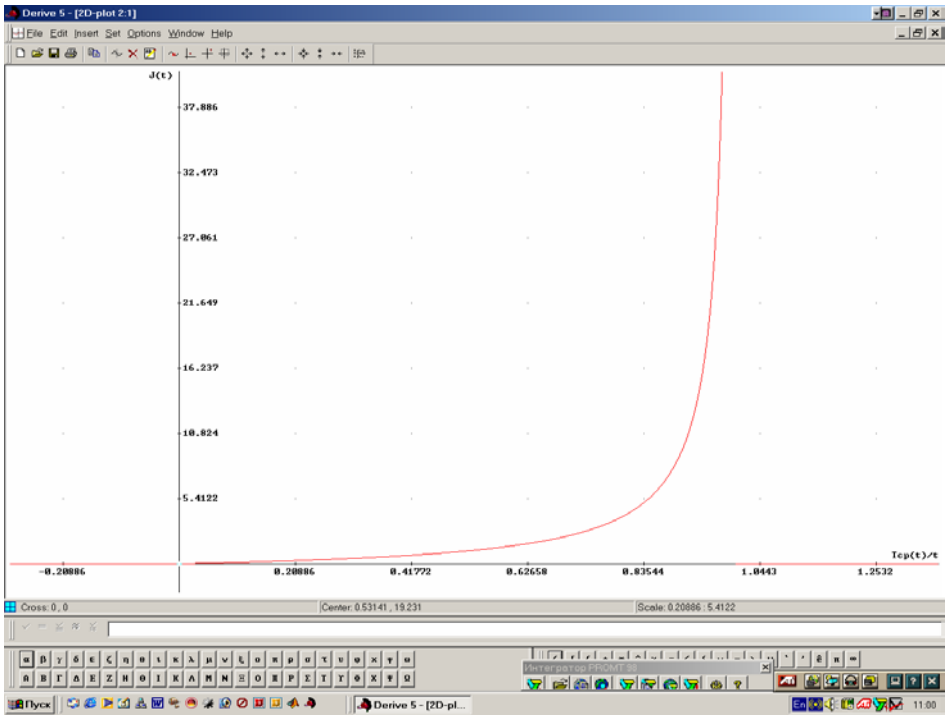


Рис. 9. Качество функционирования системы $J(t)$, в зависимости от относительного среднего времени безотказной работы $T_{cp}(t)/t$.

Таким образом, на приведенном примере показано, что система уравнений Колмогорова, характеризующая состояние объекта, имеет всегда единственное аналитическое решение и позволяет производить оценку качества системы технического обслуживания объекта.

Отметим, что предложенный эффективный алгоритм расчета относительного времени пребывания системы технического обслуживания в работоспособном состоянии не требует составления отдельной программы и легко реализуется в любом символьном математическом пакете.

Все символьные выкладки данной работы выполнены в пакете DERIVE 5.05. Заметим, что приведенный эффективный алгоритм легко может быть реализован также и в пакете MAPLE 9.01.

Литература

1. *Вентцель Е. С.* Теория вероятностей.— М. 1964.— 576 с.
2. *Сорокин А. С.* Структурные формулы некоторых классов аналитических функций в конечносвязной области // Матем. сб.— 1997.— Т. 188.— №12.— С.107–134.
3. *Рыбалко В. В.* Оценка качества системы технического обслуживания энергетических объектов // Exponenta Pro. Математика в приложениях.— 2003.— №3.— С.58–61.

УДК 519.6:519.85

ПЛП-ПОИСК И ЕГО РЕАЛИЗАЦИЯ В СРЕДЕ MATLAB

Статников И. Н, Фирсов Г. И.
Институт машиноведения РАН им. А.А.Благонравова, Москва,
e-mail: firsovgi@mtu-net.ru

Кажется очевидным, что наиболее привлекательными становятся такие методы исследования и решения задач оптимизации, которые, при условии наличия адекватной математической модели, требуют минимума априорной информации о решаемой задаче, более того, позволяют по ходу решения получать такую информацию легко и просто. К таким относится метод Монте-Карло и его различные модификации [1]. В основе использования метода Монте-Карло и его модификаций лежат принципы случайного поиска решения задачи, что и делает такой подход универсальным. Но платой за такую универсальность является определенная «слепота», и это приводит к громадным объемам вычислений даже для современных вычислительных машин, тем более, что имеет место рост размерности решаемых задач (растет число фазовых координат и число конструктивных (оптимизируемых) параметров, растет число критериев качества, характеризующих систему).

Как представляется, в значительной степени эту потребность реализует метод планируемого ЛП-поиска (ПЛП-поиска) благодаря синтезу в нем идеи дискретного квазиравномерного по вероятности зондирования J — мерного пространства варьируемых параметров α_j (j=1,…,J) и методологии планируемого математического эксперимента [2,3]. Сочетание таких идей в алгоритме ПЛП-поиска позволило, с одной стороны, осуществлять глобальный квазиравномерный просмотр заданной области варьируемых параметров, а, с другой стороны, применить многие формальные оценки из математической статистики.

Рассмотрим алгоритм ПЛП-поиска и формализованную постановку решаемой задачи при его использовании. Отметим, что успешность применения ПЛП-поиска обуславливается тем, что этот метод предназначен, в основном, для применения на предварительном этапе решения задачи, когда полученная информация позволяет принять решение об использовании других методов оптимизации (но значительно эффективнее), или об окончании решения (такое тоже возможно). В основание метода положена рандомизация расположения в области $G(\vec{\alpha})$ векторов $\vec{\alpha}$, рассчитываемых по $ЛП_\tau$-сеткам [4], и которая оказывается возможной благодаря тому, что весь вычислительный эксперимент проводится сериями. В ПЛП-поиске на сегодняшний день можно варьировать одновременно значения до 51-го па-

раметров (J = 51). Для рандомизации (случайного смешения уровней варьируемых параметров α_{ijh}) дискретного обзора $G(\vec{\alpha})$ могут быть использованы многие существующие таблицы равномерно распределенных по вероятности целых чисел. В целях экономии памяти ЭВМ в ПЛП-поиске алгоритм рандомизации построен на использовании датчика псевдослучайных чисел q ($0 < q < 1$) из [5]. Рандомизация состоит в том, что для каждой h - ой серии экспериментов (h=1,…, $H(i, j)$), где $H(i, j)$ - объем выборки из элементов Φ_{ijh} для одного критерия, вычисляется свой вектор случайный номеров строк $\vec{j} = (j_{1h}, j_{2h}, ..., j_{\beta h})$ в таблице направляющих числителей (ТНЧ) по формуле:

$$j_{\beta h} = [\mathrm{R} \times \mathrm{q}] + 1, \qquad (1)$$

а значения α_{ij} в h-ой серии рассчитываются с помощью линейного преобразования

$$\alpha_{ijh} = \alpha_{j^*} + q_{ihj_{\beta h}} \times \Delta\alpha_j,$$

где $\Delta\alpha_j = \alpha_{j^{**}} - \alpha_{j^*}$, $\alpha_{j^{**}}, \alpha_{j^*}$ — соответственно верхние и нижние границы области $G(\vec{\alpha})$; β = 1, …, J; R - любое целое число (в ПЛП-поиске R = 51); j — фиксированный номер варьируемого параметра; i = =1,…, M (j) — номер уровня j-го параметра в h-й серии; M(j) — число уровней, на которое разбивается j-й параметр; в общем случае $j_{\beta h} \neq j$ (в чем и состоит одна из целей рандомизации). Можно показать с помощью критерия Романовского [6], что числа $j_{\beta h}$, вырабатываемые по формуле (1), оказываются совокупностью равномерно распределенных по вероятности целых чисел. Обратим внимание, что M(j) и есть количество экспериментов, реализуемых в одной серии. И если M (j) = M = const и $H(i, j)$ = H = const, то в этом случае параметры *N0*, *M* и H связаны простым соотношением:

$$N0 = M \times H, \qquad (2)$$

где *N0* — общее число вычислительных экспериментов (ВЭ), при этом длина выборки из Φ_{ijh} в точности равна H. Но в общем случае, когда $M(j)$ = var, то и $H(i, j)$ = var, и тогда формула (2) для одного критерия примет такой вид:

$$N0 = \sum_{i=1}^{M(j)} H(i, j).$$

Для проведения однофакторного дисперсионного анализа [7] по всем параметрам для каждого критерия производится сортировка результатов вычислений, полученных при вычисления в точках матрицы планируемых экспериментов (МПЭ). В результате сортировки для одного критерия будет получено J матриц, состоящих из элементов Φ_{ijh} а для K критериев

будет получено $J \times K$ матриц, состоящих из элементов Φ_{ijhk}, где k — номер критерия. Этот анализ позволяет принять (или отвергнуть) с требуемой вероятностью $P = 1 - \alpha$, где α — заданный уровень значимости, следующую нулевую гипотезу: средние значения $\bar{\Phi}_{ijk}$ не существенно (случайно) отличаются от общего среднего значения k-го критерия $\bar{\Phi}_{0k}$. Если принят положительный ответ (гипотеза принята), то допускается на следующем этапе решения задачи несущественно влияющий параметр α_j не варьировать, а зафиксировать одно из его значений, например, $\alpha_j = \alpha_{ij}$ для такого i, где $\bar{\Phi}_{ijk}$ имеет наилучшее значение в смысле искомого экстремума.

Иначе говоря, инструментальная основа ПЛП-поиска состоит в следующем [8].

В пространстве варьируемых (исследуемых) параметров α_j ($j=1,...,J$) строится матрица планируемых экспериментов (МПЭ) размером ($N0 * J$),где $N0$ — число строк этой матрицы (число вычислительных экспериментов), а J — число столбцов. При этом весь численный эксперимент проводится сериями, в каждой из которых реализуется алгоритм ЛП-поиска [5]. При этом каждая серия вычислительных экспериментов рассчитывается по совокупности J строк из таблицы направляющих числителей $\{V_{jm}\}$, где $j=1,...,51$, $m=1,...,20$ [5]. Номера строк для каждой серии берутся из совокупности равномерно распределенных по вероятности целых чисел, вырабатываемой в ПЛП-поиске с помощью того же датчика псевдослучайных чисел qu ($0 < qu < 1$), что используется и при выработке случайных значений α_{ij} где $i=1,...,N0$. При реализации ЛП-поиска в каждой серии каждый варьируемый параметр в диапазоне своего изменения разбивается на M сечений. В ПЛП-поиске предусмотрены варианты, когда $M=2l$, $M=2l+1$, $M= 2^l$ ($l=2,3,4,...$). В свою очередь также реализутся для каждого из трех перечисленных случаев варианты при $M=const$ и $M=var$.

В сформированной указанным способом МПЭ производятся вычисления по каждому критерию в каждой строке матрицы. Далее определенным выше образом производится сортировка результатов вычислений, привязанная к МПЭ, и появляются возможности использования оценок математической статистики для определения статистического влияния каждого варьируемого параметра на каждый вычислявшийся критерий.

При всей сложности самой программной реализации ПЛП-поиска, в этой программе есть два объекта, которые являются ее кирпичиками. Это датчик выработки вектора *wn* случайных номеров строк (ddrns (J,np1); Приложение I) и датчик выработки псевдослучайных чисел qu (sobol (J,NP); Приложение II). Для программной реализации этих датчиков очень удобной оказалась среда MATLAB (версии 5.2 и выше) [9, 10]. Дело в том, что главной идеей получения псевдослучайного числа qu является перевод

десятичной записи номера очередного вычислительного эксперимента *NP* в двоичную; на этой основе выбираются номера столбцов из фиксированной строки таблицы направляющих числителей; стоящие в них целые числа из десятичной записи переводятся в двоичную; полученные двоичные числа складываются поразрядно по правилу логического сложения двоичных, а результат переводится в десятичную запись, дающую искомое число. Очевидно, что перечисленная совокупность действий по получению требуемого числа qu_{ij} есть не что иное как совокупность операций с двоичными разрядами записи номера *NP*. Эти операции легко реализовались в sobol (J,NP) благодаря использованию операторов поразрядной обработки MATLAB [11]:

bitand (a,b) — производит операцию поразрядного И для двух целых неотрицательных чисел a и b;

bitshift (a,k,n) — производит поразрядный сдвиг n младших разрядов неотрицательного a на k позиций (вправо, если k<0);

bitxor (a,b) — выполняет операцию поразрядного исключающего ИЛИ для неотрицательных целых a и b.

В программной реализации ПЛП-поиска в среде MATLAB [9, 10] очень легко реализуется графический интерфейс, позволяющий визуально в дополнение к статистической информации судить о границах влияния варьируемых параметров на критерии качества.

Отметим, что в ПЛП-поиске представляется возможным варьировать до 51 параметра и разбивать варьируемые параметры на число сечений до 1024, хотя реально употребляются от 4 до 256 сечений.

Теперь опишем типовую формализованную постановку задачи, для которой будет полезным использование ПЛП-поиска. Пусть задана математическая модель (ММ) исследуемой динамической системы в виде

$$L(\bar{y}(\bar{\alpha},t),\bar{\alpha}) = 0, \tag{3}$$

$$\bar{\varphi}(\bar{\alpha}) \le 0, \tag{4}$$

где L — оператор, воздействующий на систему уравнений (3) (линейный или нелинейный); $\bar{y}(\bar{\alpha},t)$ — вектор фазовых координат системы; $\bar{\alpha} = (\alpha_1,...,\alpha_J)$ — вектор коэффициентов системы (3) и (4). Далее задана исходная область $G(\vec{\alpha})$ изменения параметров в виде J-мерного параллелепипеда:

$$\alpha_j \in \left|\alpha_{j^*},\alpha_{j^{**}}\right|,\ j = 1,...,J. \tag{7}$$

И пусть задано множество критериев качества системы (в явном или неявном виде):

$$\{\Phi_k = \Phi_k(\alpha_1,...,\alpha_J),\ \bar{\alpha} \in G(\bar{\alpha}), k = 1,...,K\}. \tag{8}$$

На основе проведенного численного анализа (ПЛП-поиска) возможно:

а) определение существенных (релевантных) параметров $\alpha_m (m \le J)$ в смысле их влияния на значения каждого критерия $\Phi_k(\bar{\alpha})$; иначе говоря, статистическим путем оценить изменения производных $\partial\Phi_k(\bar{\alpha})/\partial\alpha_j$ при $\alpha_j \in (\alpha_{j^*}, \alpha_{j^{**}})$;

б) определение областей концентрации $G_k(\bar{\alpha})$ наилучших решений по каждому критерию $\Phi_k(\bar{\alpha})$; речь идет о том, чтобы основываясь на заданной метрике $\rho(\Phi_k(\bar{\alpha}), \Phi_k^+)$, где Φ_k^+ — экстремальное значение k-го критерия качества, заранее известное или определяемое по ходу проведения вычислительного эксперимента, отыскать область $G_k(\bar{\alpha})$, удовлетворяющую одновременно двум условиям: с вероятностью $P \ge Pз$ найти множество $n \subseteq N$ таких точек $\bar{\alpha}_u (u = 1,...,n)$, что в каждой из этих точек выполнится неравенство

$$\rho(\Phi_k(\bar{\alpha}), \Phi_k^+) \le \varepsilon_k$$

и при этом

$$(n / N) \ge 1 - \delta$$

Здесь $\bar{\alpha}_k \in G_k(\bar{\alpha}), 0 < \delta << 1, 0 < \varepsilon_k << 1$, $Pз$ — заданная вероятность;

в) на основе выделенных существенных параметров и областей построить регрессионные зависимости

$$\bar{\Phi}_k = \varphi_k(\alpha_1,...,\alpha_m).$$

г) выделить в K-мерном пространстве критериев множество точек Парето (или, если возможно, построить поверхность Парето); в случае задания какой-либо схемы компромисса выделить область $G_0(\bar{\alpha}) \subseteq G(\bar{\alpha})$, содержащую концентрацию компромиссных решений.

Одним из определяющих условий, повышающих эффективность применения поисковых методов при решении задач оптимального проектирования, является классификация всех параметров по степени их влияния на значения функций цели, т. е. выделение существенных параметров.

Описанный выше подход был применен при решении совместно с Я. М. Зархом задачи оптимального проектирования резонансного преобразователя (РП) для судовых валопроводов, используемых в целях изменения (снижения) уровня продольных колебаний механической системы, так как с его помощью можно воздействовать на собственные частоты системы $\omega_к$. Известно [12], что продольные колебания судовых валопроводов часто становятся причиной поломок в двигательных комплексах и повышенной вибрации корпуса судна. Данное обстоятельство связано с постоянным ростом скоростей хода судов, приводящих к увеличению неравномерности потока в диске гребных винтов и повышению переменных составляющих сил упора. Одним из перспективных путей предупреждения резонансных режимов продольных колебаний валопроводов во всем диапазоне их чисел

оборотов является использование резонансного преобразователя, представляющего собой поршень и гидравлический цилиндр. Полость цилиндра соединена с дополнительным резервуаром, и оба они заполнены маслом, находящимся под давлением, которое регулируется таким образом, чтобы уравновесить силу упора гребного винта, которая полностью передается на поршень. Таким образом, между валопроводом и корпусом главного упорного подшипника создается масляная полость, с помощью которой изменяют упруго-инерционные характеристики всей системы.

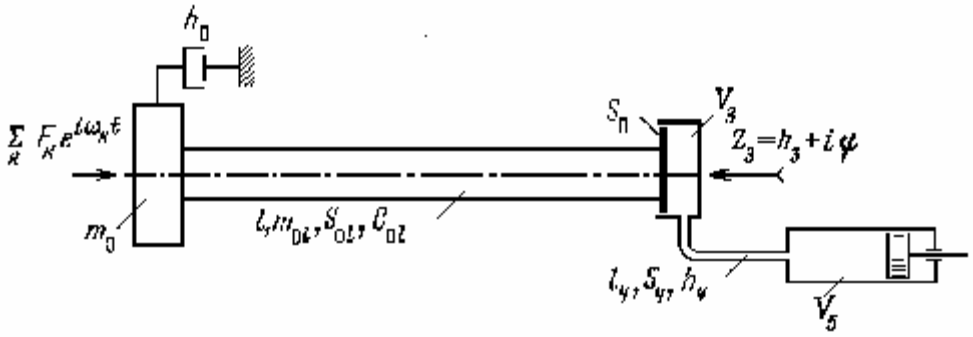


Рис. 1.

Динамическая модель системы валопровод-резонансный преобразователь-корпус судна, согласно [13, 14], представлена на рис. 1, где гребной винт моделируется массой m_0, к которой приложена сила активного сопротивления с коэффициентом сопротивления h_0, учитывающим потери как непосредственно на винте, так и на валопроводе; валопровод представлен протяженным весомым стержнем с параметрами m_{0l}, l, S_{0l}, C_{0l} соответствующими массе, длине, площади поперечного сечения и продольной жесткости реального валопровода; главный упорный подшипник (ГУП) задается его импедансом $Z_3 = h_3 + i\psi$, учитывающим полное сопротивление как собственно ГУП, так и корпуса судна; резонансный преобразователь показан как гидросистема, имеющая суммарную площадь плунжеров $S_П$, емкость полости внутри ГУП V_3 с длиной l_3, длину и проходное сечение соединительных трубопроводов l_4 и S_4 и емкость вынесенного резервуара V_5 с площадью сечения S_{45} и длиной l_5. Гидросистема заполнена маслом, имеющим плотность ρ и коэффициент динамической вязкости μ.

В работах [13, 14] дана динамическая модель этой системы и выведены математические зависимости между амплитудами колебаний главного упорного подшипника (ГУП), величинами возбуждающих сил, параметрами РП и ее собственными частотами. Эти частоты являются корнями уравнения

$$i\omega Z_П G_l(\omega) - Es_{0l}\beta G_l'(\omega) = 0,$$

где $1/\ Z_{\text{П}} = 1/\ Z_{\text{РП}} + 1/\ Z_3$, $G_l = \cos\beta l + a\sin\beta l$; $G_l^{'} = \sin\beta l - a\cos\beta l$; $a = \alpha_2 - i\beta_2$; $\alpha_2 = -m_0\omega^2 / ES_{0l}\beta$; $\beta_2 = h_0\omega / ES_{0l}\beta$; $\beta = \omega\sqrt{m_{0l} / ES_{0l}l}$; причем $G_l, G_l^{'}, E, S_{0l}, \beta, Z_3$ не зависят от параметров РП.

Входной импеданс резонансного преобразователя $Z_{РП}$ связан с его геометрическими параметрами следующим образом:

$$1/\ Z_{РП} = \frac{\alpha_4\alpha_5\alpha_6^2 - \omega\alpha_1\alpha_2(\omega\rho\alpha_4\alpha_5\alpha_6 - \alpha_1^2 G)}{i\alpha_1^2 G(\omega\rho\alpha_3\alpha_4\alpha_5\alpha_6 - \alpha_1^2 G)},$$

где $\alpha_1 = S_{\text{П}}$ - суммарная площадь плунжеров в системе, $\alpha_2 = l$ - длина полости внутри ГУП, $\alpha_3 = l_4$ и $\alpha_6 = S_4$ - длина и проходное сечение соединительных трубопроводов, $\alpha_4 = S_5$ и $\alpha_5 = l_5$ - площадь сечения и длина вынесенного резервуара, G - объемный модуль упругости для жидкости, ρ - плотность жидкости, заполняющей гидросистему РП.

Расчет и проектирование РП сводится к выбору его геометрических параметров α_j (j = 1, 2, 3, 4, 5, 6) при выбранной конструктивной схеме. Каждое такое сочетание параметров определяет степень эффективности управления резонансными свойствами системы с помощью РП. Количественно возможность такого управления оценивается по величине модуля производных $\left|d\omega_k / d\alpha_j\right|$. Чем больше модуль производной, тем эффективнее можно влиять на резонансные свойства системы. Формально такую функцию цели запишем как

$$\Phi_k(\alpha) = \max\left|d\omega_k / d\alpha_j\right|,$$

где $\bar{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ — вектор геометрических параметров РП.

При решении задачи по изложенной выше методике осуществлялась рандомизация всего процесса поиска оптимальных сочетаний параметров α_j. Таким способом строилась матрица планирования экспериментов, часть которой дана в табл. 1. Параметры матрицы планирования выбраны следующими: N = 218 — общее число всех машинных экспериментов на ЭЦВМ, r = 6 — размерность исходного пространства варьируемых параметров, N_l = 16 — число уровней, на которые разбивался каждый параметр α_j;, M_g - число серий экспериментов или число реализаций на каждом уровне параметра α_j (g = 1, 2,..., N_l); общее число экспериментов $N = \sum_{g=1}^{N_1} M_g$.

Область поиска оптимальных сочетаний параметров представляла гиперпараллелепипед, заданный следующей системой параметрических ограничений:

$$970 \le \alpha_1 \le 2300;\ 40 \le \alpha_2 \le 515;\ 10 \le \alpha_3 \le 260,$$
$$1{,}8 \le \alpha_4 \le 7{,}1;\ 300 \le \alpha_5 \le 1200;\ 1{,}5 \le \alpha_6 \le 22.$$

Таблица 1.

N	α_j					
	α_1	α_2	α_3	α_4	α_5	α_6
1	1969	5,28	260,94	496,87	183,91	11,56
2	1781	4,03	148,44	721,88	242,66	7,06
3	1843	5,91	429,69	1059,38	87,03	4,81
4	1218	6,53	373,44	1171,88	125,16	16,06
5	1343	3,41	204,69	609,88	154,53	13,81
6	1093	4,66	92,19	384,38	213,28	9,31
7	1281	4,34	232,81	440,63	110,47	5,94
8	1781	6,84	457,81	890,63	227,97	14,94
9	1031	5,59	120,31	665,62	51,72	19,44
l0	1906	4,97	176,56	553,12	81,09	3,69
11	1406	2,47	401,56	1003,12	198,59	12,69
12	1656	3,72	64,06	328,13	226,34	17,19
l3	1031	3,72	401,56	496,87	81,09	13,81
14	1531	6,22	176,56	946,87	198,59	4,81
15	1281	2,47	64,06	721,88	139,84	18,31
16	1156	4,34	457,81	843,38	169,22	16,06
17	1781	4,97	289,06	1171,88	22,34	9,31
18	1656	6,84	232,81	384,38	51,72	7,06
• • • •	• • • • •	• • • •	• • • • •	• • • • •	• • • • •	• • • •
214	1906	4,34	457,81	328,13	95,78	14,94
215	1406	6,84	232,81	778,12	213,28	5,94
216	1531	2,47	64,06	890,63	66,41	17,19
217	1031	4,97	289,06	440,63	183,91	8,19
218	1781	6,22	176,56	665,62	125,16	3,69

После всех машинных экспериментов каждый параметр α_j (фактор), в том числе и все парные произведения, был подвергнут дисперсионному анализу с целью определения его статистической значимости [6]. При этом подсчитывается среднее значение $\tilde{\Phi}(\alpha)$ для всей совокупности экспериментов

$$\tilde{\Phi}(\alpha) = \frac{1}{N}\left|\sum_{g=1}^{N_1}\sum_{h=1}^{M_g}\Phi_{gh}(\alpha)\right|.$$

Для каждого фактора α_j подсчитываются средние значения $\tilde{\Phi}_g$ на каждом *g*-м уровне, вычисляются оценки дисперсий между уровнями и внутри них. По полученным оценкам дисперсий рассчитывается критерий статистической значимости Фишера *F*. Найденное значение сопоставляется с теоретическим критерием F_T, из табл. XVIII работы [6] при соответствующих степенях свободы $\nu_1 = N_1$ и $\nu_2 = N - N_1$ и 5%-ном уровне значимости критерия *F*. Для проведения подобного анализа над 15 парными сочетаниями факторов ($C_6^2 = 15$) в каждом эксперименте подсчитываются зна-

чения всех произведений типа $\alpha_i\alpha_j$, где $i \neq j$. Затем полученные значения для каждого типа произведения при i = const и j = const группируются в интервалы так, чтобы в каждом интервале $M_g \geq 10$. В дальнейшем каждый тип парного сочетания подвергался дисперсионному анализу. По результатам анализа выявлены статистически значимые линейные и парные эффекты отдельных факторов и их парных сочетаний (табл. 2) при уровне значимости P = 0,05.

Таблица 2.

Фактор	ν_1	ν_2	F	F_T
α_5	15	202	33,46	1,70
α_4	15	202	1,87	1,70
$\alpha_2\alpha_5$	9	208	29,83	1,89
$\alpha_1\alpha_5$	12	205	29,40	1,78
$\alpha_4\alpha_5$	10	207	19,20	1,89
$\alpha_5\alpha_6$	11	206	17,90	1,80
$\alpha_3\alpha_5$	11	206	9,00	1,80

По результатам дисперсионного анализа и данным матрицы планирования экспериментов, пользуясь, например, методом наименьших квадратов, можно построить корреляционную зависимость $\tilde{\Phi}(\alpha)$ в виде полинома, содержащего линейные члены и парные сочетания табл. 2. Основываясь на результатах табл. 2, можно также построить функцию, аппроксимирующую поверхность заданной функции цели $\Phi(\alpha)$. В этом случае построенная зависимость будет носить более простой и достоверный характер по сравнению с аналогичным выражением, построенным для исходной размерности пространства исследуемых параметров, по следующим причинам: 1) размерность пространства поиска значительно сокращена (например, в данной задаче от r = 6 можно перейти к r = 2); 2) учитываются наиболее существенные парные взаимодействия типа $\alpha_i\alpha_j$; 3) с учетом первой и второй причин аппроксимация будет производиться на более «гладких» участках поверхности функции цели.

Для дальнейшего поиска сочетаний параметров, дающих максимальное значение $\Phi(\alpha)$, применялся следующий прием. На основании критерия Дункана [15] выявлены наиболее существенные в статистическом смысле интервалы у всех парных сочетаний из табл. 2 и составлена система неравенств

$$\begin{gathered} 48 \leq \alpha_2\alpha_5 \leq 333,\ 23042 \leq \alpha_1\alpha_5 \leq 130030, \\ 7331 \leq \alpha_4\alpha_5 \leq 61562,\ 57 \leq \alpha_5\alpha_6 \leq 1154,\ 1431 \leq \alpha_3\alpha_5 \leq 27317. \end{gathered} \tag{5}$$

Далее были проведены три серии экспериментов.

Серия 1. Выбран набор постоянных значений параметров α_1 = 1406,25, α_2 = 3,41, α_3 = 401,53, α_4 = 360 и α_6 = 4,81. С учетом системы не-

равенств (5) назначен диапазон варьирования $22 \le \alpha_5 \le 100$. При этом были несколько завышены верхние границы неравенств у $\alpha_1\alpha_5$ и $\alpha_3\alpha_5$. Затем применили одномерный ЛП-поиск по α_5. Усредненные результаты приведены в табл. 3.

Серия 2. Оставлен тот же набор постоянных значений α_1, α_2, α_3 и α_6,. С учетом системы (5) область поиска максимальных значений $\Phi(\alpha)$ по α_4 и α_5, определялась следующими диапазонами их варьирования:

$$905 \le \alpha_4 \le 1060;\ 21 \le \alpha_5 \le 58.$$

При этом соблюдались все неравенства в системе (5). Усредненные результаты исследования с применением ЛП-поиска в области (α_4, α_5) даны в табл. 3.

Таблица 3.

N	Среднее значение функции цели $\tilde{\Phi}(\bar{\alpha})$			
	ЛП-поиск в исходной области	Планирование экспериментов		
		1-я серия	2-я серия	3-я серия
16	2,70	4,51	8,82	7,22
32	2,27	4,56	9,07	7,80

Серия 3. Был выбран другой произвольный набор постоянных значений параметров α_1 = 1346,75, α_2 = 5,59, α_3 = 457,81 и α_6 = 2,56. Область поиска на плоскости (α_4, α_5) оставлена та же, что и во второй серии. Применялся метод ЛП-поиска. Усредненные результаты даны в табл. 3.

Кроме трех серий, поиск оптимального набора параметров производился ЛП-поиском в исходной области, заданной системой неравенств (5). Усредненные результаты также показаны в табл. 3.

В этой же исходной~ области поиска проведено 200 машинных экспериментов ЛП-поиском. При этом получено $\tilde{\Phi}(\alpha)$ = 2,62, что несущественно отличалось от среднего значения этой функции, полученного при планировании экспериментов.

Данные табл. 3 показывают, сколь значительно повышается эффективность дальнейшего применения поисковых методов, в том числе и ЛП-поиска, если на начальном этапе решения задачи оптимального проектирования использовать описанный прием определения существенных и несущественных параметров. В частности, в результате использования предварительно спланированных экспериментов удалось существенно снизить размерность пространства поиска оптимальных решений при одновременном отыскании в среднем более высоких значений функции цели.

Изложенный подход применялся при решении целого ряда конкретных задач исследования, оптимизации и идентификации различных механических систем, в частности для синтеза шарнирно-рычажного четырехзвенного механизма съемного гребня чесальной машины с разгружателем

и проектирования зубчато-рычажного механизма с остановкой, широко использующегося в эмалировочных автоматах, в автоматах пищевой и полиграфической промышленности, в револьверных подачах прессов и т. д. С помощью рандомизации области изменения параметров получены оптимальные значения инерционно-жесткостных параметров динамической системы с 23 степенями свободы, описывающей двухступенчатый планетарный судовой редуктор. Показана принципиальная возможность отстройки собственных частот от заданных рабочих диапазонов для конкретных моделей редукторов. Решены также задачи выбора рациональных параметров системы шумозащиты пневморапирного ткацкого станка типа АТПР, синтеза колебательной системы швейной машины по критериям минимальных динамических нагрузок при ограничениях на относительные перемещения исполнительных органов, минимизации динамических нагрузок в элементах трансмиссии главного привода рабочей клети широкополосного прокатного стана 200 НЛМЗ, идентификации упругих и прочностных характеристик композитного материала цилиндрической оболочки.

Литература

1. *Бусленко Н. П., Голенко Д. И, Соболь И. М. и др*. Метод статистических испытаний (метод Монте-Карло).— М.: Физматгиз,1962.— 322 с.
2. *Статников И. Н*. О структурировании пространства исследуемых параметров в задачах проектирования машин и механизмов // Проблемы машиностроения и надежности машин.— 2000.— № 5.— С.11–17.
3. *Статников И. Н., Фирсов Г. И*. ПЛП-поиск — эвристический метод решения прикладных задач оптимизации // Практика применения научного программного обеспечения в образовании и научных исследованиях. СПб.: СПбГПУ, 2003.— С.54–67.
4. *Соболь И. М.* Многомерные квадратурные формулы и функции Хаара.— М.: Наука, Гл. ред. физ.-мат. лит., 1969.— 288 с.
5. *Соболь И. М., Статников Р. Б.* Выбор оптимальных параметров в задаче со многими критериями.— М.: Наука, 1981.— 110 с.
6. *Митропольский А. К.* Техника статистических вычислений.— М.: Наука, Гл. ред. физ-мат. лит. 1971.— 576 с.
7. *Шеффе Г.* Дисперсионный анализ.— М.: Наука, Гл. ред. физ-мат. лит. 1980.— 512 с.
8. *Статников И. Н.* САПР, вычислительный эксперимент и ПЛП-поиск // Автоматизация экспериментов в динамике машин.— М.: Наука, 1987.— С.132–139.
9. *Потемкин В. Г.* Вычисления в среде MATLAB.— М.: ДИАЛОГ-МИФИ, 2004.— 720 с.
10. *Потемкин В. Г.* MATLAB 6: среда проектирования инженерных приложений.— М.: ДИАЛОГ-МИФИ, 2003.— 448 с.

11. *Кондрашов В. Е., Королев С. Б.* MATLAB как система программирования научно-технических расчетов.— М.: Мир, Институт стратегической стабильности Минатома РФ, 2002.— 350 с.
12. *Генкин М. Д., Тайчер С. Я.* К вопросу о возможности снижения уровня продольных резонансных колебаний судовых валопроводов // Виброакустическая активность механизмов с зубчатыми передачами.— М.: Наука, 1971.— С.136–145.
13. *Глазов Ю. Е., Зарх Я. М., Тайчер С. Я.* Методы подбора параметров резонансного преобразователя для судовых валопроводов // Сб.: Акустическая динамика машин и конструкций.— М.: Наука, 1973.— С.96–101.
14. *Генкин М. Д., Глазов Ю. Е., Зарх Я. М., Короткина И. С., Тайчер С. Я.* Исследование динамических свойств резонансного преобразователя // Сб.: Акустическая динамика машин и конструкций.— М.: Наука, 1973.— С.87–95.
15. *Хикс Ч.* Основные принципы планирования эксперимента.— М.: Мир, 1967.

Приложение 1.

```
function wn=ddrns(J,np1);
%DDRNS Датчик определения случайных номеров строк;
%J ЧИСЛО варьируемых параметров;
%NP1 Начальное значение векторов случайных номеров строк;
   NP=NP1;
for j1=1:J
  NP=NP+1;qu=sobol(J,NP);
 wn(j1)=round(50*qu(2))+1;
 end
```

Приложение 2.

```
function qu=sobol(J,NP);
%SOBOL Выработка вектора псевдослучайных чисел qu;
%J Число варьируемых параметров;
%NP Номер эксперимента;
v=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 3 5 15 17 51 85 255 257 771 1285 3855 4369 13107 21845 65535 65537 196611 327865 983055
1 1 7 11 13 61 67 79 465 721 823 4091 4125 4141 28723 45311 53505 250113 276231 326411
1 3 7 5 7 43 49 147 439 1013 727 987 5889 6915 16647 49925 116487 83243 116529 715667
1 1 5 3 15 51 125 141 177 759 267 1839 6929 16241 16565 17139 82207 50979 252717 851901
1 3 1 1 9 59 25 89 321 835 833 4033 3913 11643 18777 35225 102401 45059 36865 299009
1 1 3 7 31 47 109 173 181 949 471 2515 6211 2147 3169 35873 33841 99889 247315 1032727
1 3 3 9 9 57 43 43 225 113 1601 579 1731 11977 7241 63609 81003 15595 144417 685617
1 3 7 13 3 35 89 9 235 929 1341 3863 1347 4417 5087 12631 103445 152645 130127 775365
1 1 5 11 27 53 69 25 103 615 913 977 6197 14651 2507 27109 5205 91369 302231 172023
1 3 5 1 15 19 113 115 411 157 1725 3463 2817 9997 7451 12055 44877 24895 508255 574033
1 1 7 3 29 51 47 97 233 39 2021 2909 5459 2615 13329 35887 97323 83101 320901 810643
1 3 7 7 21 61 55 19 59 761 1905 3379 8119 13207 8965 9997 75591 226659 187499 628265
```

```
1 1 1 9 23 37 97 97 353 169 375 1349 5121 13313 19457 1033 62487 250917 234593 308321
1 3 3 5 19 33 3 197 329 983 893 3739 7669 2671 18391 31161 12111 259781 36159 232401
1 1 3 13 11 7 37 101 463 657 1599 347 2481 5201 3123 32253 78043 63447 508757 974837
1 1 7 13 25 5 83 255 385 647 415 387 7101 11469 11699 15865 49173 147489 81991 802875
1 3 5 11 7 11 103 29 111 581 605 2381 2677 14855 721 26903 100419 206167 241771 987201
1 1 1 3 13 39 27 203 475 505 819 2821 1405 12165 709 41543 57545 77163 357231 378135
1 3 1 15 17 63 13 65 451 833 975 1873 7423 5837 20481 12291 86017 12303 299025 774207
1 1 5 5 1 27 33 195 263 139 915 1959 725 5387 19285 5165 27985 69809 128325 164575
1 3 3 3 25 17 115 177 19 147 1715 1929 2465 12483 13057 28931 54019 21251 62233 248081
1 1 3 15 29 15 41 105 249 203 1223 2389 471 12945 32321 29377 127427 103759 472541 1008719
1 3 1 7 3 23 79 17 275 81 1367 3251 2887 1279 4865 64771 24321 42247 338691 599831
1 3 7 9 31 29 17 47 369 337 663 1149 1715 187 12285 53631 110851 4357 153347 671033
1 1 5 13 11 3 29 169 393 829 629 243 5595 8133 4929 10817 8261 189901 255947 734787
1 3 1 9 5 21 119 109 167 989 525 3609 5689 11819 15889 48083 67537 63993 336469 749285
1 3 3 1 23 13 75 149 333 375 469 1131 441 14471 12625 8881 34707 85105 479495 911133
1 3 3 11 27 31 73 15 473 365 981 1701 3169 7615 8405 41135 106823 107847 339031 977907
1 1 7 7 19 25 105 213 469 131 1667 143 4485 2981 12593 60913 15703 26967 507907 344073
1 3 5 5 21 9 7 135 101 215 1587 1339 6311 4081 28637 60935 94129 109273 475921 281389
1 1 1 15 5 49 59 253 21 733 1251 3497 3557 7223 13425 58577 69521 217151 424277 789985
1 1 1 1 1 33 65 191 451 451 451 2499 483 11843 28285 12029 86021 217093 348165 176165
1 3 5 15 17 19 21 155 229 447 481 1571 3781 10799 15893 959 19793 213491 377941 414943
1 1 7 11 13 29 3 175 247 177 721 983 3195 9277 15405 19637 87283 186143 343297 1041185
1 3 7 5 7 11 113 63 297 57 483 4021 5213 2031 4677 26607 20391 54345 259163 741087
1 1 5 3 15 19 61 47 403 471 1209 1625 5085 15371 19493 56445 26369 27399 521499 132383
1 3 1 1 9 27 89 7 497 979 1457 3217 185 6603 1129 36087 66817 98051 451841 175361
1 1 3 7 31 15 45 23 61 197 415 1163 7323 7563 25321 52563 37745 81777 235347 539895
1 3 3 9 9 25 107 39 361 251 1435 2977 1713 11617 14979 5455 68289 209987 346179 521289
1 3 7 13 3 3 25 55 215 517 725 3391 4021 4129 4099 12345 102733 21287 128115 20689
1 1 5 11 27 21 5 71 393 137 861 675 5875 12061 25469 47423 29505 124097 444613 430923
1 3 5 1 15 51 49 87 125 567 41 3093 5363 3471 17589 50131 33137 98739 361365 426737
1 1 7 3 29 19 111 103 285 1021 1619 1495 4977 15919 6731 43771 23313 151281 270519 11187
1 3 7 7 21 29 119 119 501 167 1579 3443 5441 1097 13483 58779 36561 116819 420599 998391
1 1 1 9 23 5 33 135 277 877 1701 557 1779 10369 15325 33331 118321 59665 498897 494137
1 3 3 5 19 1 67 153 199 929 869 675 6777 14343 18465 63615 43349 30799 322567 939017
1 1 3 13 11 39 101 169 301 269 1151 1489 287 8475 6929 46013 52785 75249 14035 507165
1 1 7 13 25 37 19 185 19 327 1897 2303 6919 16139 16677 34579 120981 239693 73299 863545
1 3 5 11 7 43 39 201 83 997 1679 3925 1517 305 21765 45827 91157 113679 204881 761911
1 1 1 3 13 7 91 217 351 91 1355 3705 1875 7621 4381 9079 94533 37261 431301 176455];
q=zeros(J,1);
i=1;
while (2^i-NP)<=0
  i=i+1;
  end
mr=i;j=0;
while j<J
  j=j+1;i=1:1:mr;
  b=bitshift(1,i-1);
  c=bitand(NP,b);
  x(mr-i+1)=c./(2.^(i-1));
   c(j,1)=0;i=0;
 while i<mr
       i=i+1;
 if x(i)==0
   v(j,mr-i+1)=x(i);
   c(j,i+1)=c(j,i);
```

```
    end,
b=bitshift(v(j,mr-i+1),i-1);
c(j,i)=bitxor(c(j,i),b);
c(j,i+1)=c(j,i);
    end
    q=c(j,i)/(2^mr);
 qu(j)=q;
end
```

УДК 519.3

АНАЛИТИЧЕСКОЕ ВЫЧИСЛЕНИЕ КВАЗИДИФФЕРЕНЦИАЛА В ПАКЕТЕ MATLAB

Тамасян Г. Ш.
Санкт-Петербургский государственный университет,
факультет прикладной математики — процессов управлений, Санкт-Петербург,
e-mail: grigoriytamasjan@mail.ru

Негладкий анализ — раздел современной математики, в которой изучаются недифференцируемые функции (см. [1]-[6]). Необходимость нахождения субдифференциала, супердифференциала, квазидифференциала и кодифференциала возникает в задачах негладкой оптимизации. Негладкие функции все чаще встречаются при математическом моделировании реальных процессов и практически важных задач, например, задачи распознавания образов, классификации, идентификация, технической и медицинской диагностики. Ниже обсуждается проблема создания программного обеспечения для аналитического вычисления квазидифференциала — одного из понятий негладкого анализа.

Введение

Пусть на выпуклом множестве $S \subset E^n$ задана конечная вещественная функция f, то есть в каждой точке $x \in S$ значение $f(x)$ - конечное вещественное число. В частности, S может совпадать с E^n.

Определение. Если существует конечный предел

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = \lim_{\Delta \to 0} \frac{f(x_0 + \Delta) - f(x_0)}{\Delta}, \qquad (1)$$

то говорят, что функция f дифференцируема в точке x_0. Предел (1) называется производной функции f в точке x_0. Функция называется дифференцируемой на S, если она дифференцируема в каждой точке $x \in S$.

Рассмотрим функцию $f : S \to E$, которая не является дифференцируемой на S, то есть не в каждой точке $x_0 \in S$ существует предел (1). Возьмем $g \in E^n$ (g называется направлением). Если существует конечный предел

$$f'(x_0, g) = \lim_{\alpha \downarrow 0} \frac{f(x_0 + \alpha g) - f(x_0)}{\alpha}, \qquad (2)$$

то говорят, что функция f дифференцируема в точке x_0 по направлению g, а значение $f'(x_0, g)$ называется производной функции f в точке x_0 по

направлению g. Здесь $\alpha \downarrow 0$ означает, что $\alpha \to 0,\ \alpha > 0$. Конечно, в (2) предполагается, что α достаточно мало, так что $x_0 + \alpha g \in S$. Говорят, что функция f дифференцируема по направлениям, если предел (2) существует и конечен при всех g.

Известно (см. [4–6]), что если $f(x)$ — выпуклая конечная функция, определенная на выпуклом открытом множестве $S \subset E^n$, то она дифференцируема по направлениям в любой точке $x_0 \in S$, причем

$$\frac{\partial f(x_0)}{\partial g} = \max_{v \in \partial f(x_0)} (v, g), \tag{3}$$

где $\partial f(x_0)$ — субдифференциал функции $f(x)$ в точке x_0. Множество $\partial f(x_0) \subset E^n$ является выпуклым компактом.

Таким образом, с каждой точкой x_0 у функции $f(x)$, являющейся выпуклой функцией или функцией максимума ($f(x) = \max\limits_{i \in \mathrm{l}:N} f_i(x)$, где $f_i(x)$ - выпуклые функции), связано множество $\partial f(x_0)$ - субдифференциал. При этом оказывается, что:

1. Производная по направлениям выражается с помощью $\partial f(x_0)$ по формуле (3).
2. Необходимое условие минимума на E^n имеет вид
$$0_n \in \partial f(x^*), \tag{4}$$
где x^* - точка минимума $f(x)$ на E^n.
3. Если $0 \notin \partial f(x_0)$, то направление
$$g(x_0) = \frac{-v(x_0)}{\|v(x_0)\|},$$
где $\|v(x_0)\| = \min\limits_{v \in \partial f(x_0)} \|v(x)\|$, является направлением наискорейшего спуска функции $f(x)$ в точке x_0.

Отсюда видно, что понятие субдифференциала позволяет исследовать функции, и в частности для решения задач оптимизации.

Пусть на открытом множестве $S \subset E^n$ задана конечная функция $f(x)$. Будем говорить, что $f(x)$ квазидифференцируема в точке $x_0 \in S$, если она дифференцируема в точке x_0 по любому направлению $g \in E^n$ и если существуют выпуклые компакты $\underline{\partial} f(x_0) \subset E^n$ и $\overline{\partial} f(x_0) \subset E^n$ такие, что

$$\frac{\partial f(x_0)}{\partial g} = \lim_{\alpha \to +0} \frac{f(x_0 + \alpha g) - f(x_0)}{\alpha} = \max_{v \in \underline{\partial} f(x_0)} (v, g) + \min_{w \in \overline{\partial} f(x_0)} (w, g). \tag{5}$$

Пару множеств $Df(x_0) = \left[\underline{\partial} f(x_0), \overline{\partial} f(x_0)\right]$ назовем квазидифференциалом

(КВД) функции $f(x)$ в точке x_0, а множества $\underline{\partial} f(x_0)$ и $\overline{\partial} f(x_0)$ - соответственно субдифференциалом и супердифференциалом функции $f(x)$ в точке x_0.

Из формулы (5) видно, что КВД определяется неоднозначно: пара множеств

$$A(x_0) = \left[\underline{\partial} f(x_0) + B, \overline{\partial} f(x_0) - B \right],$$

где $B \in E^n$ — произвольный выпуклый компакт, тоже является квазидифференциалом.

Для того чтобы квазидифференцируемая функция $f(x)$ достигала своего наименьшего (наибольшего) значения в точке x^*, необходимо, чтобы

$$-\overline{\partial} f(x^*) \subset \underline{\partial} f(x^*) \ (-\underline{\partial} f(x^*) \subset \overline{\partial} f(x^*)).$$

Для негладких функций можно сформулировать и достаточные условия (см. [5-6]). Рассмотрим основные элементы квазидифференциального исчисления, то есть исчисление квазидифференциалов. Нам понадобится ввести операции сложения пар выпуклых многогранников и умножение на вещественное число.

Пусть $D_1 = \left[A_1, B_1 \right]$, $D_2 = \left[A_2, B_2 \right]$ — пары выпуклых многогранников, где A_1, B_1, A_2 и B_2 - выпуклые многогранники. Положим

$$D_1 + D_2 = \left[A, B \right],$$

где $A = A_1 + A_2$, $B = B_1 + B_2$. Здесь, как обычно,

$$C_1 + C_2 = \left\{ c = c_1 + c_2 \middle| \ c_1 \in C_1, \ c_2 \in C_2 \right\}.$$

Если $C_1 = co\left\{ c_{1i} \middle| \ i \in I \right\}$, $C_2 = co\left\{ c_{2j} \middle| \ j \in J \right\}$, то

$$C_1 + C_2 = co\left\{ c = c_{1i} + c_{2j} \middle| \ i \in I, \ j \in J \right\}.$$

Пусть $D = \left[A, B \right]$, λ - вещественное число. Положим

$$\lambda D = \begin{cases} \left[\lambda A, \ \lambda B \right], & \lambda \geq 0, \\ \left[\lambda B, \ \lambda A \right], & \lambda < 0. \end{cases}$$

Основные формулы квазидифференциального исчисления.

1. Если $\varphi_1(x)$ и $\varphi_2(x)$ — квазидифференцируемые в точке x_0 функции и $D\varphi_1(x_0)$, $D\varphi_2(x_0)$ - их квазидифференциалы в этой точке, то и функции

$$f_1(x) = \lambda_1 \varphi_1(x) + \lambda_2 \varphi_2(x), \quad f_2(x) = \varphi_1(x) \varphi_2(x),$$

$$f_3(x) = \frac{\varphi_1(x)}{\varphi_2(x)} \ (\text{если } \varphi_2(x_0) \neq 0)$$

тоже являются квазидифференцируемые в точке x_0, при этом

$$Df_1(x_0) = \lambda_1 D\varphi_1(x_0) + \lambda_2 D\varphi_2(x_0), \quad Df_2(x_0) = D\varphi_1(x_0)\varphi_2(x_0) + \varphi_1(x_0) D\varphi_2(x_0),$$

$$Df_3(x_0) = \frac{\varphi_2(x_0) D\varphi_1(x_0) - \varphi_1(x_0) D\varphi_2(x_0)}{\varphi_2{}^2(x_0)}. \quad (6)$$

2. Если $\varphi_j(x)$, $j \in I$,— квазидифференцируемые в точке x_0 функции, $I = 1,2,...,N$, $D\varphi_j(x_0)$ — их квазидифференциалы в этой точке, то и функции

$$f_1(x) = \max_{j \in I} \varphi_j(x), \; f_1(x) = \min_{j \in I} \varphi_j(x)$$

тоже являются квазидифференцируемыми в точке x_0, при этом

$$Df_1(x_0) = \left[\underline{\partial} f_1(x_0), \overline{\partial} f_1(x_0) \right], \; Df_2(x_0) = \left[\underline{\partial} f_2(x_0), \overline{\partial} f_2(x_0) \right],$$

где

$$\underline{\partial} f_1(x_0) = co \left\{ \underline{\partial} \varphi_j(x_0) - \sum_{\substack{k \in R(x_0) \\ k \neq j}} \overline{\partial} \varphi_k(x_0) \;\middle|\; j \in R(x_0) \right\},$$

$$\overline{\partial} f_1(x_0) = \sum_{k \in R(x_0)} \overline{\partial} \varphi_k(x_0), \; \underline{\partial} f_2(x_0) = \sum_{k \in Q(x_0)} \underline{\partial} \varphi_k(x_0), \quad (7)$$

$$\overline{\partial} f_2(x_0) = co \left\{ \overline{\partial} \varphi_j(x_0) - \sum_{\substack{k \in Q(x_0) \\ k \neq j}} \underline{\partial} \varphi_k(x_0) \;\middle|\; j \in Q(x_0) \right\}.$$

Здесь $R(x_0) = \left\{ k \in I \middle| \varphi_k(x_0) = f_1(x_0) \right\}$, $Q(x_0) = \left\{ k \in I \middle| \varphi_k(x_0) = f_2(x_0) \right\}$.

Заметим, что множество квазидифференцируемых функций представляет собой линейное пространство, замкнутое относительно всех «алгебраических» операций, операций взятия поточечного максимума, минимума, последовательного максимина.

Примеры вычисления квазидифференциалов

Пример 1. Пусть функция $f(x)$ непрерывно дифференцируема на $S \subset E^n$ и $x_0 \in S$. Поскольку

$$\frac{\partial f(x_0)}{\partial g} = (f'(x_0), g),$$

то ясно, что $f(x)$ является квазидифференцируемой функцией на S, причем в качестве КВД функции $f(x)$ в точке x_0 можно взять пару множеств

$$Df(x_0) = \left[f'(x_0), 0_n \right],$$

т. е. $\underline{\partial} f(x_0) = \left\{ f'(x_0) \right\}$, $\overline{\partial} f(x_0) = \left\{ 0_n \right\}$. Очевидно, что пара множеств $\left[0_n, f'(x_0) \right]$ тоже является КВД функции $f(x)$ в точке x_0. Таким образом,

функция $f(x)$, дифференцируемая в точке x_0, является и субдифференцируемой, и супердифференцируемой в этой точке.

Пример 2. Пусть $f(x)$ - конечная выпуклая на выпуклом открытом множестве $S \subset E^n$ функция, $x_0 \in S$. Из (3) ясно, что функция $f(x)$ является квазидифференцируемой в точке x_0, причем в качестве КВД можно взять

$$Df(x_0) = \left[\partial f(x_0), 0_n\right],$$

т. е. $\underline{\partial} f(x_0) = \{\partial f(x_0)\}$, $\overline{\partial} f(x_0) = \{0_n\}$. Поскольку $\overline{\partial} f(x_0) = \{0_n\}$, то выпуклая функция субдифференцируема в точке x_0.

Пример 3. Пусть $f(x)$ - конечная вогнутая на выпуклом открытом множестве $S \subset E^n$ функция и $x_0 \in S$. В качестве КВД можно взять

$$Df(x_0) = \left[0_n, \overline{\partial} f(x_0)\right],$$

Поскольку $\underline{\partial} f(x_0) = \{0_n\}$, то вогнутая функция супердифференцируема в точке x_0.

Пример 4. Пусть $x = (x_1, x_2) \in E^2$, $x_0 = (0,0) = 0_2$.

$$f(x) = f(x_1, x_2) = 3|x_1| - 4|x_2|.$$

Имеем $f(x) = 3f_1(x) + (-4)f_2(x) = 3\max\{f_{11}(x), f_{12}(x)\} - 4\max\{f_{21}(x), f_{22}(x)\}$, где $f_{11}(x) = x_1$, $f_{12}(x) = -x_1$, $f_{21}(x) = x_2$, $f_{22}(x) = -x_2$.

Функции $f_{11}(x)$, $f_{12}(x)$, $f_{21}(x)$ и $f_{22}(x)$ непрерывно дифференцируемы, и потому

$$Df_{11}(x_0) = \left[\underline{\partial} f_{11}(x_0), \overline{\partial} f_{11}(x_0)\right] = \left[\left\{\begin{pmatrix}1\\0\end{pmatrix}\right\}, \left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right],$$

$$Df_{12}(x_0) = \left[\underline{\partial} f_{12}(x_0), \overline{\partial} f_{12}(x_0)\right] = \left[\left\{\begin{pmatrix}-1\\0\end{pmatrix}\right\}, \left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right],$$

$$Df_{21}(x_0) = \left[\underline{\partial} f_{21}(x_0), \overline{\partial} f_{21}(x_0)\right] = \left[\left\{\begin{pmatrix}0\\1\end{pmatrix}\right\}, \left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right],$$

$$Df_{22}(x_0) = \left[\underline{\partial} f_{22}(x_0), \overline{\partial} f_{22}(x_0)\right] = \left[\left\{\begin{pmatrix}0\\-1\end{pmatrix}\right\}, \left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right].$$

Применяя (6) и (7) имеем

$$Df_1(x_0) = \left[\underline{\partial} f_1(x_0), \overline{\partial} f_1(x_0)\right] = \left[co\left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}\right\}, \left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right],$$

$$Df_2(x_0) = \left[\underline{\partial} f_2(x_0), \overline{\partial} f_2(x_0)\right] = \left[co\left\{\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\-1\end{pmatrix}\right\}, \left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right].$$

$$Df(x_0)=\left[\underline{\partial} f(x_0),\overline{\partial} f(x_0)\right]=3Df_1(x_0)-4Df_2(x_0)=$$

$$=3\left[co\left\{\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}-1\\0\end{pmatrix}\right\},\left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right]-4\left[co\left\{\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}0\\-1\end{pmatrix}\right\},\left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right]=$$

$$=\left[co\left\{\begin{pmatrix}3\\0\end{pmatrix},\begin{pmatrix}-3\\0\end{pmatrix}\right\},\left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right]-\left[co\left\{\begin{pmatrix}0\\4\end{pmatrix},\begin{pmatrix}0\\-4\end{pmatrix}\right\},\left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right]=$$

$$=\left[co\left\{\begin{pmatrix}3\\0\end{pmatrix},\begin{pmatrix}-3\\0\end{pmatrix}\right\},\left\{\begin{pmatrix}0\\0\end{pmatrix}\right\}\right]+\left[\left\{\begin{pmatrix}0\\0\end{pmatrix}\right\},co\left\{\begin{pmatrix}0\\4\end{pmatrix},\begin{pmatrix}0\\-4\end{pmatrix}\right\}\right]=$$

$$=\left[co\left\{\begin{pmatrix}3\\0\end{pmatrix},\begin{pmatrix}-3\\0\end{pmatrix}\right\},co\left\{\begin{pmatrix}0\\4\end{pmatrix},\begin{pmatrix}0\\-4\end{pmatrix}\right\}\right].$$

Элементы программного обеспечения

Для наглядности нахождения КВД предлагается воспользоваться приложением с графическим интерфейсом пользователя (см. рис. 1).

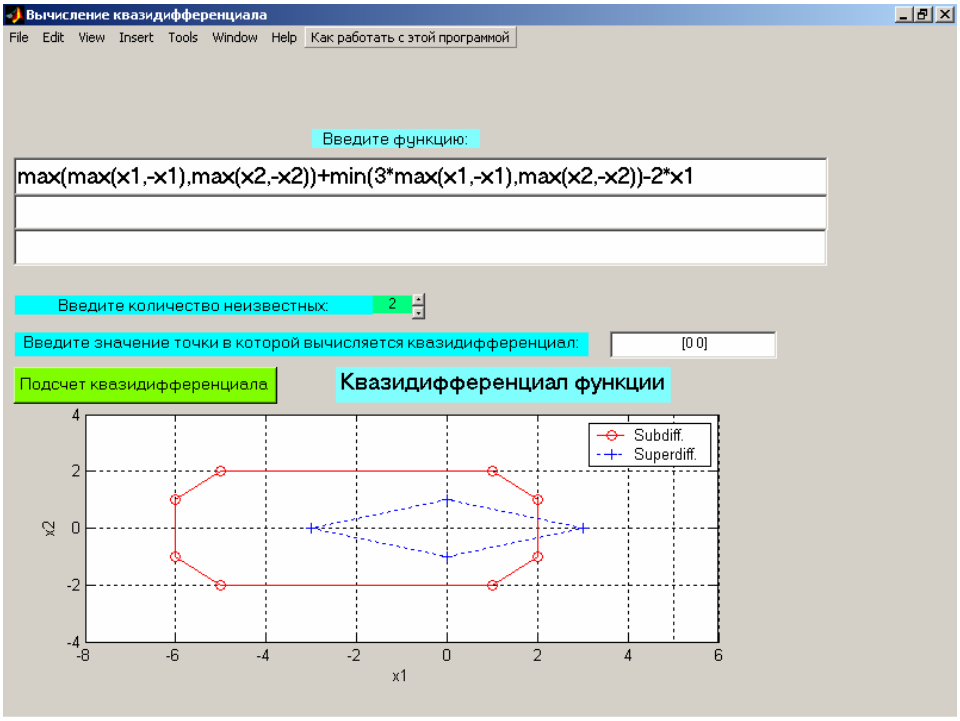


Рис. 1. Интерфейс программы для вычисления КВД.

Для вычисления КВД пользователю предлагается ввести функцию в текстовые поля на (рис. 1) их три. При помощи полосы скроллинга указывается количество неизвестных, в данном примере их две $x1$, $x2$. Автоматически, при указании количества неизвестных при помощи полосы скроллинга, создается нулевой вектор соответствующей размерности, который

отображается в текстовом поле. Этот нулевой вектор, который конечно можно изменить, и есть, то значение точки, в которой будет вычислен квазидифференциал. После всех этих действий нажимая на кнопку «подсчет квазидифференциала», запускается алгоритм аналитического нахождения квазидифференциала. Если количество переменных не более трех, то квазидифференциал отображается на графике, где субдифференциал это красный многоугольник с вершинами из кружочков, а супердифференциал это синий многоугольник с (пунктирной линией) вершинами, которого являются знаки «плюс».

Данный алгоритм, позволяет вычислять квазидифференциал от функций n - переменных, содержащей любые комбинации (суперпозиции) функций $\max$, $\min$, возведения в целую степень и всех «алгебраических» операций. Например

$$\sum_k (\min_i f_{ik}(x) + \min_j g_{jk}(x))^{m_k} + \min_i \max_j \ldots \max_k f_{ij\ldots k}(x),$$

здесь $x = (x_1, x_2, \ldots, x_n) \in E^n$, m_k - целое число.

Для реализации данной программы потребовались процедуры для работы со строками, матрицами, символьными вычислениями основанные на встроенной в MATLAB библиотеке пакета Maple, визуальной средой GUIDE предназначенной для написания приложений с графическим интерфейсом, а так же ToolBox Optimization.

Алгоритм решения поставленной задачи состоит из двух больших этапов. На первом этапе, анализируя введенную пользователем функцию, конструируется граф (дерево), а на втором этапе из полученного графа строится искомый квазидифференциал (КВД). Рассмотрим каждый из этапов более детально.

Первый этап. Программа анализирует выражение для построения графа следующей структуры. Это всевозможные суперпозиции функций $\max$, $\min$, гладких функций от любого количества переменных, а также четырех арифметических операций $(+,-,*,/)$ и операции возведения в степень $(\cdot)^n$. Сам алгоритм получения графа большой и имеет ряд тонкостей, поэтому рассмотрим данный этап на примере. Пусть

$$f(x) = \max\{\max\{x1, -x1\}, \max\{x2, -x2\}\} - 2 \cdot x1 + \min\{3 \cdot x2, -x1^2\}, \quad (8)$$

здесь $x = (x_1, x_2) = (x1, x2) \in E^2$.

Наша цель разложить выражение на *простейшие элементы* используя указанные выше операции $(\max, \min, +, -, *, /, (\cdot)^n)$, здесь под *простейшими элементами* подразумевается дифференцируемые функции.

Данное выражение $f(x)$ разбивается на слагаемые операцией «+», таким образом, получаем первые три ветви (см. рис. 2.)

$$f_{11}(x) = \max\left\{\max\left\{x1, -x1\right\}, \max\left\{x2, -x2\right\}\right\},$$
$$f_{12}(x) = -2 \cdot x1,$$
$$f_{13}(x) = \min\left\{3 \cdot x2, -x1^2\right\}.$$

Теперь каждую ветвь анализируем по отдельности. Из первой ветви, применяя, операцию «max« образуется еще две

$$f_{21}(x) = \max\left\{x1, -x1\right\}, \qquad f_{22}(x) = \max\left\{x2, -x2\right\}.$$

Вторая ветвь $f_{12}(x)$ заканчивается, так как она представляет собой дифференцируемую функцию. Из третьей ветви исходят еще две

$$f_{23}(x) = 3 \cdot x2, \qquad f_{24}(x) = -x1^2.$$

операцией «min«, и заметим, что они на этом заканчиваются, так как являются дифференцируемыми функциями. Каждая из ветвей $f_{21}(x)$ и $f_{22}(x)$ операцией «max« образует еще по две,

$$f_{31}(x) = x1, \; f_{32}(x) = -x1,$$
$$f_{33}(x) = x2, \; f_{34}(x) = -x2,$$

соответственно. На этом процесс построения графа заканчивается, ибо наше выражение было разложено до дифференцируемых функций.

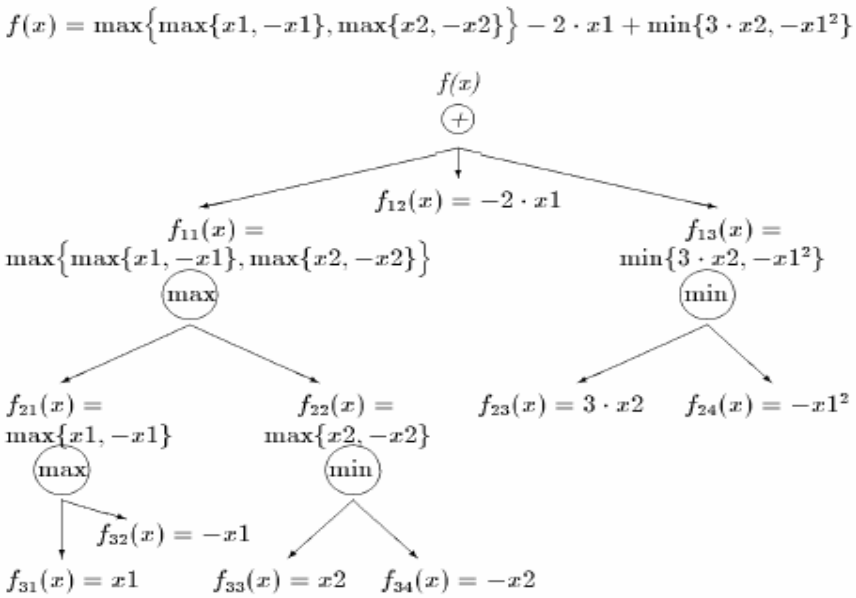


Рис. 2. Дерево (граф) функции.

Второй этап. Применяя построенный в первом этапе граф, найдем квазидифференциал (КВД) нашей функцией. Во-первых, найдем КВД наших *простейших элементов*, например, в точке $x_0 = (0,0) = 0_2$, в зависимости от того в какие композиции (суперпозиции) функций «max« или «min« они входили:

$$\begin{aligned} &Df_{12}(x_0)=\left[f'_{12}(x_0),0_2\right]=\left[(-2,0),0_2\right], \quad Df_{23}(x_0)=\left[0_2,f'_{23}(x_0)\right]=\left[0_2,(0,3)\right], \\ &Df_{24}(x_0)=\left[0_2,f'_{24}(x_0)\right]=\left[0_2,0_2\right], \qquad Df_{31}(x_0)=\left[f'_{31}(x_0),0_2\right]=\left[(1,0),0_2\right], \\ &Df_{32}(x_0)=\left[f'_{32}(x_0),0_2\right]=\left[(-1,0),0_2\right], \quad Df_{33}(x_0)=\left[f'_{33}(x_0),0_2\right]=\left[(0,1),0_2\right], \\ &Df_{34}(x_0)=\left[f'_{34}(x_0),0_2\right]=\left[(0,-1),0_2\right]. \end{aligned} \tag{10}$$

Теперь используя найденные КВД (10), граф и данные полученные при его построение, а, также применяя аппарат квазидифференциального исчисления мы найдем КВД нашей функции $f(x)$ в точке $x=x_0$. Зная квазидифференциалы функций $f_{31}(x)$ и $f_{32}(x)$ найдем КВД функций $f_{21}(x)$: $Df_{21}(x_0)=\left[co\left\{(1,0),(-1,0)\right\},0_2\right]$.

Аналогично, последовательно находим

$$Df_{22}(x_0)=\left[co\left\{(0,1),(0,-1)\right\},0_2\right],$$

$$Df_{11}(x_0)=\left[co\left\{(1,0),(-1,0),(0,1),(0,-1)\right\},0_2\right],$$

$$Df_{13}(x_0)=\left[0_2,co\left\{(0,3),(0,0)\right\}\right],$$

$$Df(x_0)=Df_{11}(x_0)+Df_{12}(x_0)+Df_{13}(x_0)=$$

$$=\left[co\left\{(-1,0),(-3,0),(-2,1),(-2,-1)\right\},co\left\{(0,0),(0,3)\right\}\right].$$

Для построения графа (дерева) на первом этапе использовалась функция **graf_tree**:

[g,oper,term,b,oper0,term0] = graf_tree(func)

где входной аргумент func — это строка, содержащая выражение функции $f(x)$. Выходной аргумент g — это массив, содержащий граф (дерево) функции $f(x)$, oper — матрица, в которой указаны какие операции были выполнены для образования графа (дерева), term — матрица, где элемент term(i, j) указывает количество ветвей исходящих из g(i, j), а аргументы b, oper0, term0 — вспомогательные массивы. При помощи функции **co_diff_smooth.m**:

[subd1, superd1] = co_diff_smooth(func1, dim, x0)

находится квазидифференциал гладкой функции func1, т. е. *простейшего элемента*. Здесь входной аргумент dim — означает количество неизвестных, x0 — значение точки x_0. Выходные аргументы subd1 и superd1— это массивы, субдифференциал и супердифференциал, соответственно.

При помощи функций **oper_sum.m**, **oper_div.m**, **oper_mult.m**, **oper_power.m**, **oper_min.m** и **oper_max .m** реализуется указанное во введение квазидифференциальное исчисление. Функция **sum_convex.m**: s = sum_convex(a,b) — складывает выпуклые многогранники a и b, где a, b и s матрицы, столбцы которых являются вершинами многогранника.

Заключение

Используя КВД можно проверить условия оптимальности, если они не выполнены, найти направления наискорейшего спуска или подъема, а также КВД применяется в численных методах для поиска оптимума. После несложных модификаций выше указанный алгоритм позволяет находить *непрерывный кодифференциал*. Непрерывные кодифференциалы более пригодны для численных методов, чем КВД. Наглядность и скорость построения КВД в системе MATLAB, позволяет исследовать широкий класс негладких функционалов, а возможность представления в визуальной форме допускает использования в учебных целях.

Работа осуществлена при поддержке Российского фонда фундаментальных исследований (проект РФФИ № 03-01-00668).

Литература

1. *Демьянов В. Ф.* Обобщение понятия производной в негладком анализе // Соросовский образовательный журнал.— 1996.— №5.— С.121–127.
2. *Демьянов В. Ф.* Негладкий анализ на плоскости. Часть I // Соросовский образовательный журнал.— 1997.— №8.— С.122–127.
3. *Демьянов В.Ф.* Негладкий анализ на плоскости. Часть II // Соросовский образовательный журнал.— 1997.— №9.— С.123–127.
4. *Демьянов В. Ф., Малоземов В. Н.* Введение в минимакс.— М.: Наука, 1972.— 368 с.
5. *Демьянов В. Ф., Васильев Л. В.* Недифференцируемая оптимизация.— М.: Наука, 1981.— 384 с.
6. *Демьянов В. Ф., Рубинов А. М.* Основы негладкого анализа и квазидифференциального исчисления.— М.: Наука, 1990.— 432 с.
7. *Ануфриев И. Е.* Самоучитель MATLAB 5.3/6.x.— СПб.: БХВ-Петербург, 2003.— 736 с.
8. *Матросов А. В.* Maple 6. Решение задач высшей математики и механики.— СПб.: БХВ-Петербург, 2001.— 528 с.
9. *Дьяконов В. П., Абраменкова И. В., Круглов В. В.* MATLAB 5.3.1. с пакетами расширений / Под ред. В. П. Дьяконова.— М.: Нолидж, 2001.— 880 с.

УДК 519.68:[5/6+3]

МОДЕЛИРОВАНИЕ МАГНИТНЫХ СВОЙСТВ ДВОЙНЫХ ПЕРОВСКИТОВ

Тарасевич Ю. Ю., Панченко Т. В., Манжосова Е. Н.
Астраханский государственный университет, Астрахань,
e-mail: tarasevich@astranet.ru

Двойными перовскитами называются соединения со структурой $A(B1_{1/2}B2_{1/2})O_3$. Атомы сортов B1 и B2 в упорядоченных соединениях чередуются в шахматном порядке (рис. 1). В неупорядоченных соединениях строгий порядок чередования атомов сортов B1 и B2 нарушается. Нарушение чередующегося порядка атомов B1 и B2 в этих соединениях связывают с изменением физических свойств.

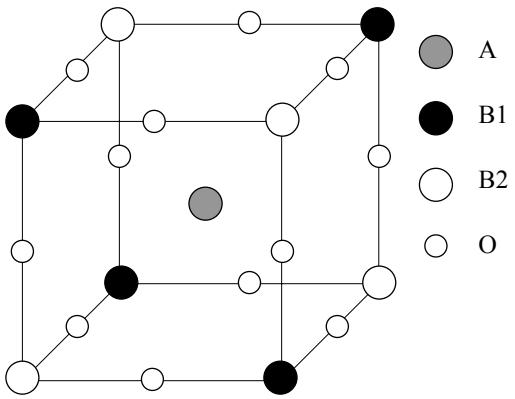


Рис. 1. Структура двойного перовскита.

Большое число соединений имеют структуру двойного перовскита. Среди них соединения с формулой $A_2(FeM)O_6$ (A = Ba, Sr, Ca и M = Mo, Re) представляют особенный интерес, поскольку в образцах Sr_2FeMoO_6 был обнаружен эффект магнетосопротивления при малых полях и комнатных температурах. Ионы Fe и Mo в этом соединении чередуются в позициях атома B перовскита со структурой ABO_3. Локальный спин S = 5/2 порождается ионами Fe^{3+} (электронная конфигурация $3d^5$), в то время как полоса проводимости частично занята $4d^1$ электронами ионов Mo^{5+}. В этом случае антиферромагнитный суперобмен между спинами $3d^5$ S = 5/2 и $4d^1$ S = 1/2 приводит к большой намагниченности ниже температуры Кюри T_C: Ba_2FeMoO_6 демонстрирует магнетосопротивление при комнатных температурах и антиферримагнитный фазовый переход при температуре $T_C \approx$ 320–340 K.

В случае частично неупорядоченного соединения ионы Fe^{3+} занимают узлы Mo^{5+} и наоборот, что приводит к суперобменному взаимодейст-

вию типа Fe–O–Fe и Fe–O–Mo. Если ион Fe^{3+} (S = 5/2), находящийся в позиции Mo^{5+}, окружен ионами Fe^{3+}, его магнитный момент стремиться стать антипараллельным спинам ионов Fe^{3+}, что приводит к снижению намагниченности.

В пакете MATLAB нами создана программа, предназначена для вычисления намагниченности и свободной энергии двойных перовскитов в зависимости от степени беспорядка в расположении атомов сортов B1 и B2 и температуры. В программе реализовано моделирование методом Монте-Карло с использованием одного из вариантов алгоритма Метрополиса [1].

При моделировании учитываются только атомы сортов B1 и B2, используется трехмерная кубическая решетка атомов с периодическими граничными условиями. Каждый атом характеризуется спином S_i (числовым значением и направлением). Магнитное взаимодействие между атомами задается обменными интегралами J_{ij}, где *i* и *j* могут принимать значения B1 и B2.

Энергия системы определяется по формуле.

$$E = \sum_{i \neq j} J_{ij} \cdot S_i \cdot S_j \,. \tag{1}$$

Намагниченность системы определяется по формуле

$$M = \frac{1}{N} \cdot \sum_i S_i \,, \tag{2}$$

где N — полное число атомов.

В программе реализован алгоритм Метрополиса следующим образом: на первом шаге (инициализация) моделируется кристаллическая решетка размера *Msize*×*Msize*×*Msize* со случайной ориентацией спинов и заданной долей беспорядка в расположении атомов. Если доля беспорядка равна 0, то атомы B1 и B2 располагаются в шахматном порядке (упорядоченная кристаллическая решетка).

На втором шаге случайным образом выбирается атом. Затем вычисляется энергия решетки для случаев положительной (E_{up}) и отрицательной (E_{down}) ориентации спина выбранного атома по формуле (1). Спин выбранного атома устанавливается положительным с вероятностью

$$p(up) = \frac{\exp\left(-E_{\text{up}} / kT\right)}{\exp\left(-E_{\text{up}} / kT\right) + \exp\left(-E_{\text{down}} / kT\right)}$$

или отрицательным с вероятностью

$$p(down) = 1 - p(up),$$

где T — температура системы, k — постоянная Больцмана. Второй шаг повторяется заданное количество раз (*steps*) (один шаг Монте-Карло).

На третьем шаге энергия (1) и намагниченность решетки (2) заносятся в промежуточный массив результатов.

Второй и третий этапы повторяются многократно. Если флуктуации энергии становятся менее заданной величины (в наших расчетах — 1%) или число итераций превысит максимально допустимое значение (в наших расчетах от 250 до 500 итераций), то процесс моделирования прекращается.

Входными данными в программе являются:

- *Msize* — размер кубической решетки;
- *Tmax, Tmin, Tstep* — начальная, конечная температура системы и шаг изменения температуры;
- *MaxIter, MinIter* — максимальное и минимальное число итераций;
- *eps* — точность вычислений;
- *steps* — количество шагов Метрополиса;
- J_{B1B2} — константа обменного взаимодействия между атомами B1 и B2;
- J_{B1B1} — константа обменного взаимодействия между атомами B1 и B1;
- J_{B2B2} — константа обменного взаимодействия между атомами B2 и B2;
- S_{B1} — значение модуля спина атома B1;
- S_{B2} — значение модуля спина атома B2;
- *kB* — постоянная Больцмана. В программе предполагается измерение температуры и энергии системы в Кельвинах, поэтому постоянная Больцмана $k = 1$;
- *fn* — имя файла, в котором сохраняются результаты работы программы.

Выходными данными являются:

- *energy* — энергия на атом;
- *err* — погрешность в определении *energy*;
- *mr1* — намагниченность системы;
- *dmr1* — погрешность в определении намагниченности
- *mr2* — разность намагниченностей в подрешетках;
- *dmr2* — погрешность в определении *mr2*.

Выходные данные сохраняются в файле *fn*. Кроме того, в отдельном файле сохраняется последняя конфигурация спинов системы, что позволяет запустить программу с использованием ранее полученного распределения спинов.

На рис. 2 и рис. 3 приводятся результаты расчетов энергии для одной из концентраций дефектов для соединения $Sr_2(FeMo)O_6$, полученных с использованием значений интегралов обменного взаимодействия из работы [3].

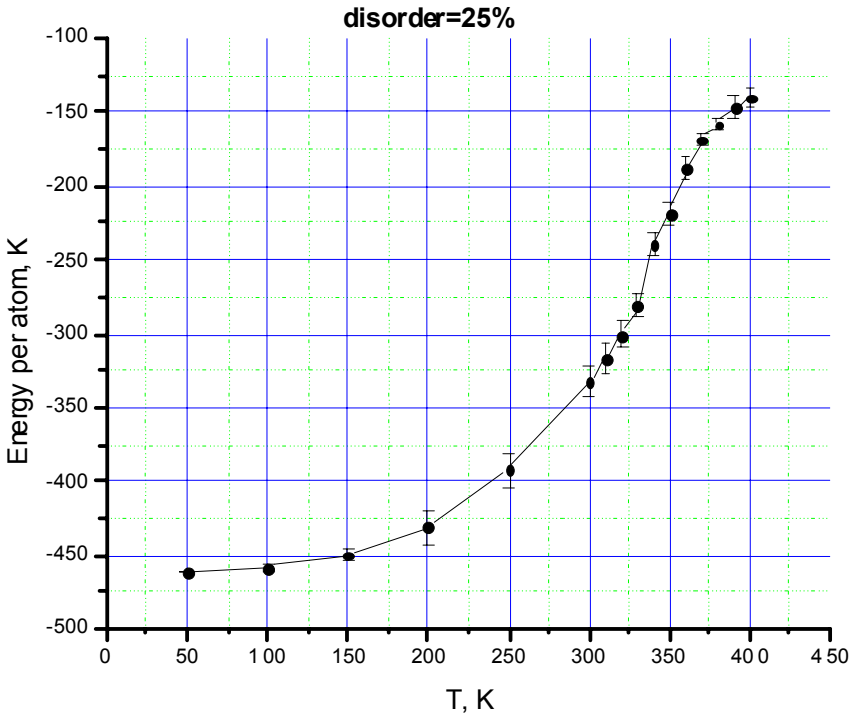


Рис. 2.

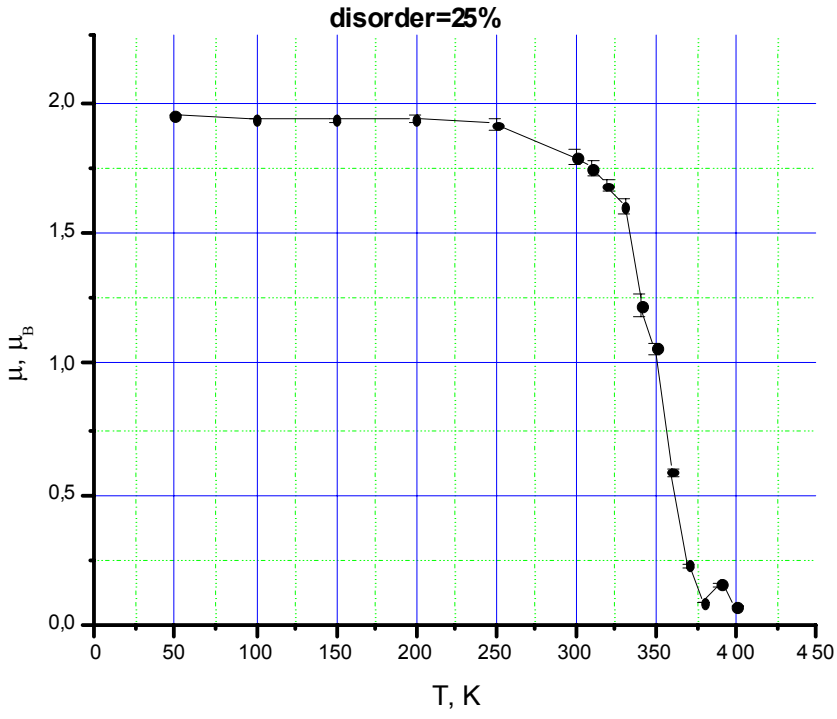


Рис. 3.

При температуре 0K алгоритм Метрополиса неприменим. Для этого случая нами разработана программа, предназначенная для моделирования

коррелированной перколяции[1] на кубической решетке с узлами двух сортов.

В основу программы положен алгоритм Хошена–Копельмана [5] или алгоритмом многократной маркировки кластеров[2], как первоначально назвали его сами авторы. Главным достоинством алгоритма является то, что он позволяет определить порог перколяции и распределение кластеров по размерам за один проход по решетке. Кроме того, алгоритм позволяет экономить машинную память, так как для его работы достаточно хранить в памяти только один ряд (слой) рассматриваемой решетки.

В основе алгоритма лежит соображение о том, что принадлежность узла к тому или иному кластеру является глобальным свойством и может быть определена только после просмотра всей решетки. Идея алгоритма заключается в том, что всем занятым узлам решетки приписываются различные кластерные метки.

Рассмотрим работу алгоритма на примере задачи узлов на квадратной решетке. Моделировать решетку будет массив a размером $L \times L$. Пусть узлы решетки заполнены с вероятностью p. Будем генерировать случайные числа, равномерно распределенные на интервале [0,1]. Если сгенерированное число меньше или равно p, очередному элементу массива a присвоим значение 1, иначе — 0.

Для того, чтобы работа со всеми элементами массива была единообразной, удобно добавить к нему одну строку с номером 0 и один столбец с номером 0, заполненные нулям. Граничные условия такого типа называется свободными. Кроме того, широко используются периодические граничные условия. Для подсчета распределения кластеров по размерам потребуется вспомогательный массив N.

Начнем просматривать последовательно все элементы массива, начиная с элемента $a[1,1]$. Нулевой столбец будем пропускать. Возможны следующие ситуации.

1. Если текущий узел не имеет соседних занятых узлов, то предположительно он относится к новому кластеру и ему приписывается очередной номер кластерной метки k.
2. Заносим в массив N размер нового кластера $N(k)$: = 1. Поскольку генерация решетки и присвоение кластерной метки происходит одновременно, то при анализе соседних узлов, естественно, рассматривается только та часть решетки, которая уже сгенерирована к текущему моменту.

---

[1] Перколяция (percolation — англ.) означает протекание. В русской литературе можно встретить и теория перколяции, и теория протекания и даже теория просачивания. Название возникло в связи с тем, что ряд первых работ в этом направлении был посвящен процессам просачивания (протекания) жидкостей или газов через пористую среду. До сих пор это направление занимает существенную часть в работах по теории перколяции. Подробнее см. в книге [2].

[2] Цепочка связанных объектов, называется в теории перколяции кластер (cluster — англ. — гроздь).

3. Если текущий узел имеет только один соседний занятый узел, то эти два узла принадлежат к одному и тому же кластеру. Текущему узлу присваиваем номер правильной кластерной метки соседнего узла $k_{\text{proper}}^{\text{neighbor}}$, увеличиваем размер кластера на единицу $N(k_{\text{proper}}^{\text{neighbor}}) := N(k_{\text{proper}}^{\text{neighbor}}) + 1$.

4. Если среди соседей текущего узла имеется несколько занятых узлов, то текущий узел объединяет соседние кластеры в один. Главной находкой авторов алгоритма является идея о том, что не следует проводить переприсваивание кластерных меток уже просмотренным узлам. Определяем минимальную правильную кластерную метку соседних узлов и присваиваем ее текущему узлу. Подсчитываем полное число узлов в новом объединенном кластере $N(k) := 1 + \sum_i N(k_i)$.

5. Суммирование проводится по всем соседним занятым узлам с положительными кластерными метками. Теперь различные узлы, принадлежащие одному и тому же кластеру, помечены различными кластерными метками. Чтобы отметить их принадлежность к одному кластеру необходимо изменить их кластерные метки. Для этого в элементы массива *N*, соответствующие неправильно помеченным кластерам, заносится —*k* — номер правильной кластерной метки со знаком минус.

Таким образом, когда вся решетка будет сформирована, пустые узлы будут помечены нулями, а занятые узлы — положительными целыми числами. Кроме того, будет сформирован массив кластерных меток, состоящий из положительных и отрицательных целых чисел. Узлам, сразу помеченным правильными кластерными метками, соответствуют положительные целые числа — размер кластера, а узлам, которые в процессе анализа изменили свои первоначальные метки, — отрицательные.

В программе реализовано очевидное обобщение алгоритма на случай кубической решетки. Использование пакета MATLAB 6 позволило записать алгоритм чрезвычайно компактно и сделать анализ различных ситуаций единообразным. Ниже мы приводим ту часть программы, в которой реализован собственно алгоритм Хошена–Копельмана.

```
if a(i,j,q) = = 1   % текущий узел занят
   nbs=[];          % массив с правильными кластерными метками
                    % соседних узлов

% Последовательно проверяем все соседние узлы и, если они заполнены, % добавляем в массив номера правильных кластерных меток
   if (a(i-1,j,q) ~= 0)
      nbs = [nbs, proper(a,LL,i-1,j,q)];
   end;
   if (a(i,j-1,q) ~= 0)
      nbs = [nbs, a(i,j-1,q)];
   end;
   if (a(i,j,q-1) ~= 0)
```

```
      nbs = [nbs, proper(a,LL,i,j,q-1)];
   end;

% следующие две проверки обеспечивают
% задание периодических граничных условий
   if (i = = msize + 1) & (a(2,j,q) ~= 0)
      nbs = [nbs, proper(a,LL,2,j,q)];
   end;
   if (j = = msize + 1) & (a(i,2,q) ~= 0)
      nbs = [nbs, proper(a,LL,i,2,q)];
   end

   if isempty(nbs) % текущий узел не имеет заполненных соседей
      a(i,j,q) = k;    % присваиваем узлу кластерную метку
      LL(k) = 1;     % размер кластера — 1 узел
      k = k+1;       % увеличиваем счетчик кластерных меток
   else              % у текущего узла есть заполненные соседи
       minlab = min(nbs); % определяем минимальную правильную метку
       a(i,j,q) = minlab;     % и присваиваем ее текущему узлу
       LL(minlab) = sum(LL(unique(nbs))) + 1; % определяем новый
    % размер кластера
   for w=1:length(nbs)  % меняем метки у всех кластеров,
         if nbs(w) ~= minlab % соединившихся в данном узле
            LL(nbs(w)) = - minlab;
         end % if
     end % for
  end
end   % if
```

Программа интерактивно запрашивает входные данные, а результаты моделирования записывает в файл.

Входные данные:

- *msize* — линейный размер матрицы, на которой проводится моделирование;
- *N* — число испытаний;
- *pc* — параметр, определяющий степень беспорядка. При pc = 0 атомы двух сортов расположены в узлах кристаллической решетке строго в чередующемся порядке, при pc = 0.5 расположения атомов хаотичное.
- *FileName* — имя файла, в который будут записаны результаты моделирования. Данные записываются в формате MATLAB в файл с расширением .mat.

Выходные данные:

- *Msize*, *N* и *pc* —значения, заданные во входных данных;
- *y* — вероятность появления стягивающего кластера;
- *av* — средний размер кластера, усредненный по всем испытаниям; при вычислении среднего размера кластера стягивающий кластер не учитывается;

- *Pinf* — мощность бесконечного кластера — вероятность того, что наугад выбранный узел принадлежит стягивающему кластеру[3].
- *SS* — массив размера msize[3]/2, содержащий усредненное по всем испытаниям распределение кластеров по размерам, то есть SS(i) равен среднему числу кластеров размера i.

Магнитные свойства зачастую связывают со средним числом соседей у узлов, принадлежащих перколяционному кластеру. На рис. 4 изображен график зависимости среднего числа соседей в зависимости от степени беспорядка. Наиболее вероятно, что характер зависимости вблизи порога перколяции определяется эффектами, связанными с конечнвм размером решетки.

Как показали наши предыдущие исследования [4], при концентрации дефектов приблизительно более 15% в системе образуется перколояционный кластер. Это значит, что при больших концентрациях дефектов в системе $A_2(FeMo)O_6$ происходит фазовое расслоение на макроскопические области: состоящий из цепочек Fe–O–Fe, и имеющие антиферромагнитные свойства, состоящие из цепочек Fe–O–Mo и имеющие ферромагнитные свойства и парамагнитные области, состоящие из цепочек Mo–O–Mo.

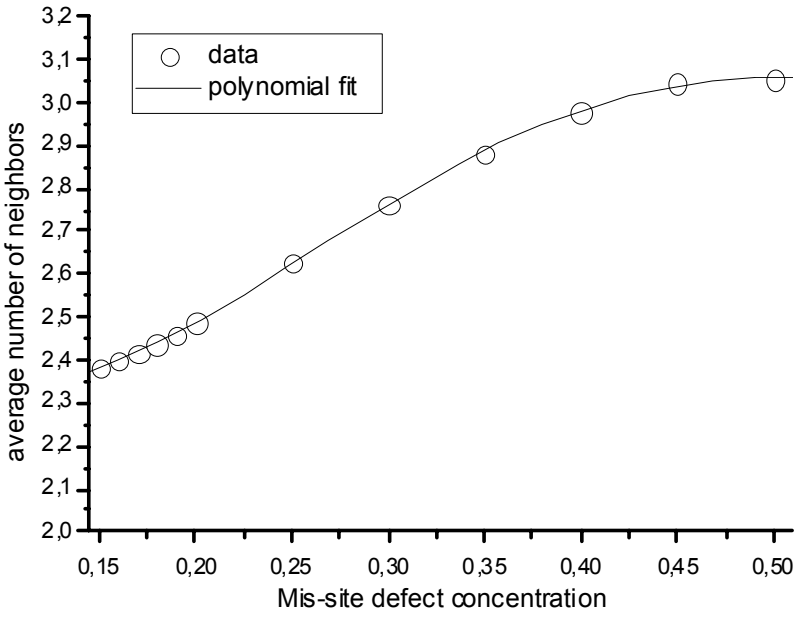


Рис. 4.

[3] Кластер, соединяющий две противоположные стороны системы, называется *перколяционным* (*p*ercolating), *бесконечным* (infinity), *стягивающим* (s*p*anning) или *соединяющим* (connecting). Изучение свойств соединяющего кластера — одна из задач теории перколяции. Ниже порога перколяции имеются только кластеры конечного размера.

Литература

1. *Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Keller E.* Equation of state calculations for fast computing machines // J. Chem. Phys.— 54.— 1953.— P.1114.
2. *Тарасевич Ю. Ю.* Перколяция: теория, приложения, алгоритмы.— М.: Едиториал УРСС, 2002.
3. *Ogale A. S., Ogale S. B., Ramesh R., Venkatesan T.* Octahedral cation site disorder effects on magnetization in double-perovskite Sr_2FeMoO_6: Monte Carlo simulation study // Applied Physics Letters.— V.75.— N.4.— 1999.— P.537–539.
4. *Tarasevich Yu. Yu., Manzhosova E. N.* On site percolation on the correlated simple cubic lattice // International Journal of Modern Physics C.— V.14.— N.10.— 2003.— P.1–8.
5. *Hoshen J., Kopelman R.* Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. // Phys. Rev. B.— **14**(8).— 1976.— P.3438–3445.

УДК 621.588.38:628.8

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ПЕРЕНОСА ЗАГРЯЗНЯЮЩИХ ЧАСТИЦ ПРИ ИЗГОТОВЛЕНИИ ИНТЕГРАЛЬНЫХ СХЕМ

Терещенко А. М., Ревякин А. М.
Московский институт электронной техники
(государственный технический университет), Москва,
e-mail: hm2@miee.ru

Постоянно растущая необходимость хранить и обрабатывать большие массивы информации требует непрерывного совершенствования вычислительной техники. Основой элементной базы современной вычислительной техники являются сверхбольшие интегральные схемы (СБИС) с уровнем интеграции 4М-16М.

В связи с этим основная тенденция развития микроэлектроники — повышение уровня интеграции микросхем — реализуется путем увеличения размера кристалла и уменьшения минимальных размеров элементов. Уменьшение топологических норм проектирования СБИС, сопровождается соответствующим снижением критических размеров загрязняющих частиц (ЗЧ), которые становятся соизмеримыми с размерами вируса $\approx 0{,}07$ мкм. Осаждение ЗЧ на поверхности кристаллов является одной из основных причин дефектов изделий микроэлектроники, а также оказывает вредное воздействие на обслуживающий персонал и окружающую природу. Поэтому решение проблемы определения концентрации и распространения ЗЧ в воздушных потоках является важной практической задачей.

Для ее решения предложена математическая модель переноса ЗЧ в производственных помещениях при наличии воздушных потоков и различных источников ЗЧ. Применение пакета MATLAB позволило эффективно проанализировать процесс переноса ЗЧ и разработать алгоритм выбора оптимальных направлений воздушных потоков для увеличения выхода годных интегральных схем. В основе алгоритма лежат методы дискретной оптимизации, в частности методы решения задач целочисленного линейного программирования. Особое внимание в докладе уделено вопросам оптимального размещения типового оборудования для производства интегральных схем по критерию минимальной приносимой дефектности.

Описаны критические траектории ЗЧ, соответствующие физической модели процесса. Предложены схемы оптимальной организации движения воздушных потоков, при которых в рабочие зоны попадает минимальное количество загрязняющих частиц. Уравнения модели решены методом сеток. Вычисления и визуализация результатов проведены с использованием пакета MATLAB.

Литература

1. *Лисовец Ю. П., Ревякин А. М., Рычагов М. Н., Терещенко С. А.* Пакет MATLAB и его применение в лабораторном компьютерном практикуме.— М.: МИЭТ, 1998.
2. *Дьяконов В. П.* Справочник по применению системы PC MATLAB.— М.: Наука, 1993.
3. *Потемкин В. Г.* MATLAB: Справочное пособие.— М.: Диалог-МИФИ, 1997.
4. *Колесников Д. Н., Дашутина Е. В., Пахомова В. И.* Введение в MATLAB и примеры решения задач оптимизации и моделирования.— СПб.: Изд-во СПбГТУ, 1995.

УДК 004

МОДЕЛИРОВАНИЕ РАСПРОСТРАНЕНИЯ ЭЛЕКТРОМАГНИТНОЙ ВОЛНЫ В СЛУЧАЙНОЙ ТРЕХМЕРНОЙ ДИСКРЕТНОЙ СРЕДЕ

Федотов И. В.,
Томский политехнический университет, Кафедра вычислительной техники, Томск, e-mail: tenshi3@mail.ru
Спицын В. Г.
Томский политехнический университет, Кафедра вычислительной техники, Томск, e-mail: spitsyn@ce.cctpu.edu.ru

Аннотация

Целью данной работы является создание численной модели многократного взаимодействия электромагнитной волны с дискретными неоднородностями, хаотически расположенными в случайной трехмерной дискретной среде. Задача решается методом Монте-Карло. В качестве средства для решения поставленной задачи выбран пакет MATLAB.

В данной работе проведено моделирование процесса распространения волны в трехмерно — неоднородной случайной дискретной среде. Решена задача прохождения волны через случайную слоистую среду, включающую полупрозрачный объект в форме параллелепипеда. Исследованы зависимости энергий рассеянного и поглощенного сигнала от факторов, описывающих неоднородную структуру случайной дискретной среды. Проведены численные расчеты углового спектра многократно рассеянной волны.

Метод

В решении задач распространения электромагнитных волн в случайных неоднородных дискретных средах используются в основном аналитические методы. В ряде работ (см. [3–5]) рассмотрены методы исследования систем уравнений, составленных для решения задачи определения коэффициентов отражения и поглощения для произвольного числа слоев. В результате получены общие схемы, в виде которых можно представить решения. Однако эти решения имеют чрезвычайно громоздкий вид и, если число слоев превышает 3, почти бесполезны для анализа влияния того или иного слоя на коэффициент поглощения или коэффициент преломления. В представленной работе задача решается методом Монте-Карло [1, 2]. В данной работе создается численная модель многократного взаимодейст-

вия электромагнитной волны с дискретными неоднородностями, хаотически расположенными в случайной трехмерной дискретной среде.

Полагается, что источник электромагнитной волны с изотропной диаграммой направленности находится на поверхности слоистой среды. Длина волны меньше размеров слоев и рассеяние происходит некогерентным образом на статистически независимых дискретных неоднородностях [6]. Излучатель электромагнитной волны представляется источником фотонов с соответствующей диаграммой излучения. Начальные координаты фотонов задаются в точке расположения излучателя. Тип взаимодействия волны с дискретными неоднородностями определяется в соответствии с задаваемыми сечениями рассеяния и поглощения.

Созданная в данной работе модель многократного взаимодействия электромагнитной волны с дискретными неоднородностями, хаотически расположенными в случайной трехмерной дискретной среде учитывает текущее значение параметров, описывающих состояние среды. В результате вычислений мы получаем распределение коэффициента поглощения среды, энергий рассеянного и поглощенного сигналов в координатном пространстве, а так же углы выхода фотонов из среды.

Результаты численного моделирования

Рассмотрен источник электромагнитной волны с изотропной индикатрисой излучения, расположенный в центре декартовой системы координат на поверхности среды с координатами X = 0, Y = 50, Z = 50. Среда характеризуется присутствием 22 слоев и объекта в форме параллелепипеда. Предполагается, что взаимодействие потока фотонов со случайными дискретными неоднородностями происходит в соответствие с изотропной индикатрисой рассеяния. Все результаты представлены в относительных единицах. Данные показаны в поперечном сечении в плоскости XY. Величина энергии нормирована на максимум энергии в каждом кадре.

На рис. 1 и рис. 2 представлены исходные данные о средней длине свободного распространения волны в среде и коэффициенте поглощения волны в среде соответственно. На рис.3, рис. 4, рис. 5 представлены результаты расчетов энергии рассеянного сигнала и энергии поглощенного сигнала в среде соответственно, а так же углы выхода фотонов из среды.

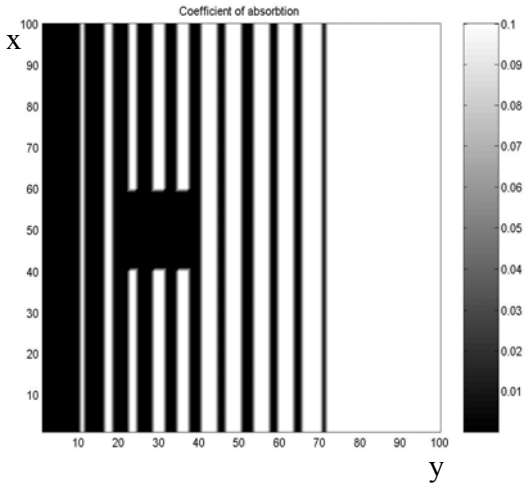


Рис. 1. Коэффициент поглощения в среде.

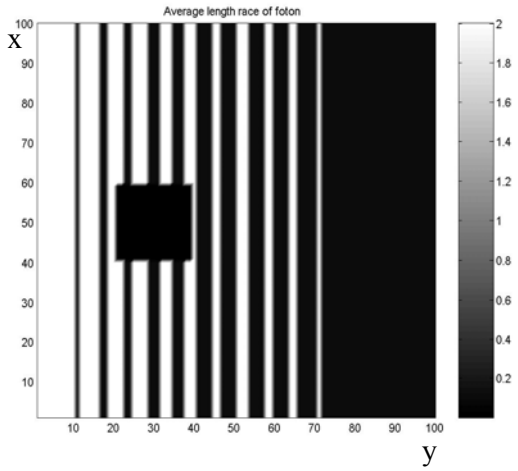


Рис. 2. Средняя длина свободного распространения фотона в среде.

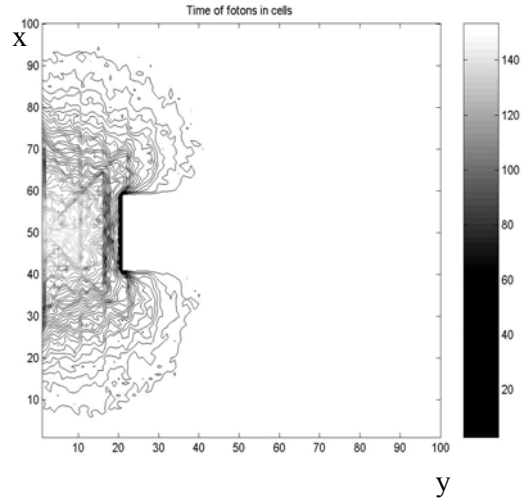


Рис. 3. Распределение энергии рассеянного сигнала в среде.

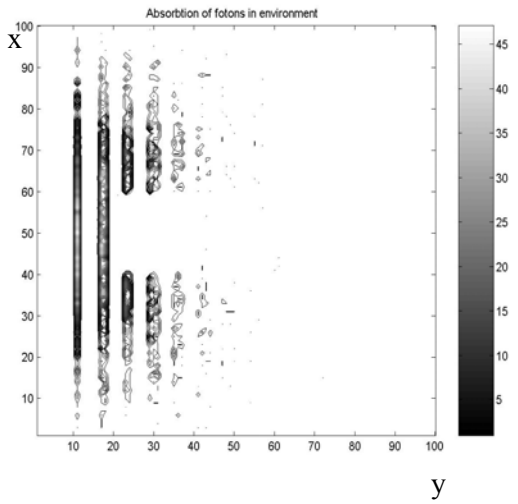


Рис. 4. Распределение энергии поглощенного сигнала в среде.

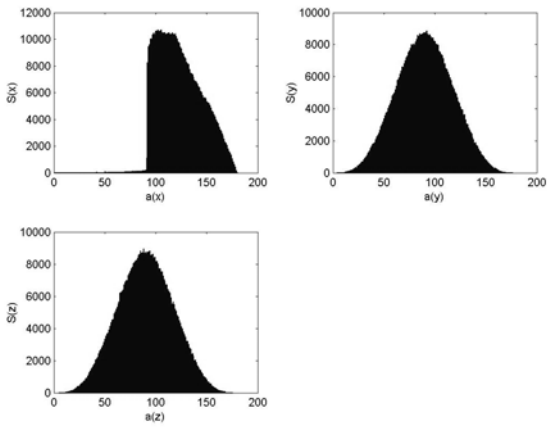


Рис. 5. Углы выхода фотонов из среды.

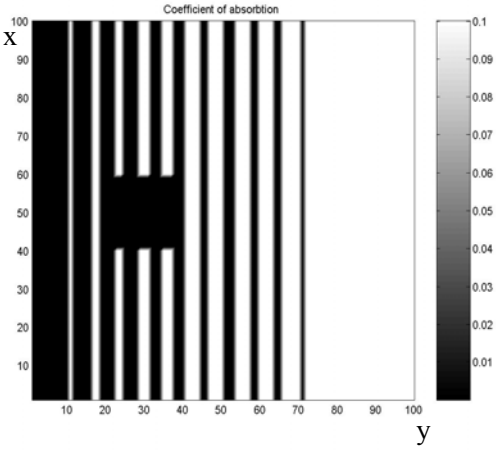


Рис. 6. Коэффициент поглощения в среде.

На рис. 6, рис. 7 представлены исходные данные для случая, когда средняя длинна пробега в объекте увеличена по сравнению с предыдущем экспериментом. Рис. 8, рис. 9, рис. 10 представляют результаты моделирования. Видно, что в свете рассеянного сигнала объект практически не виден. Это связано с тем, что фотоны проникают внутрь объекта. На изображении поглощенной энергии напротив контрастность объект лучше, чем в предыдущем эксперименте. Такой эффект объясняется проникновением фотонов через объект. Углы выходов фотонов практически не изменились в связи с тем, что объект в данном случае незначительно влияет на общую картину выхода фотонов из среды.

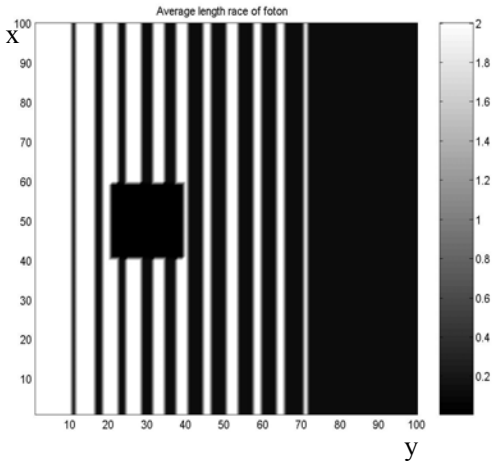


Рис. 7. Средняя длина свободного распространения фотона в среде.

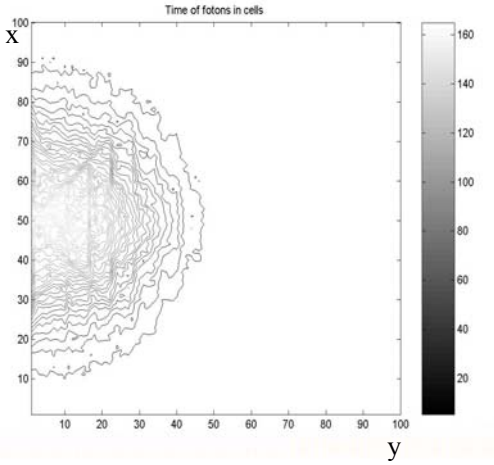


Рис. 8. Распределение энергии рассеянного сигнала в среде.

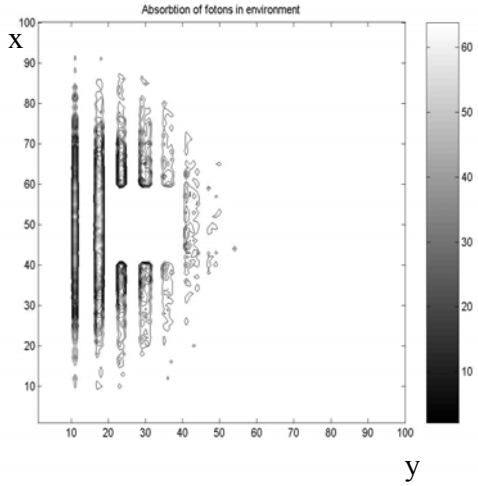


Рис. 9. Распределение энергии поглощенного сигнала в среде

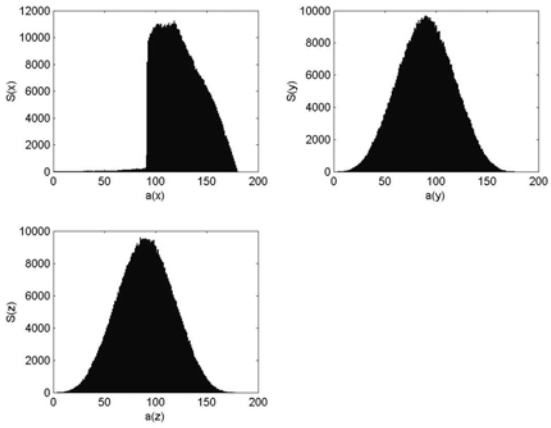


Рис. 10. Углы выхода фотонов из среды.

Исходные данные для третьего эксперимента представлены на рис. 11 и рис. 12. Объект изменен и расположен таким образом, что бы оказать

значительное влияние на выход фотонов из среды. Результаты моделирования представлены на рис. 13, рис. 14, рис. 15.

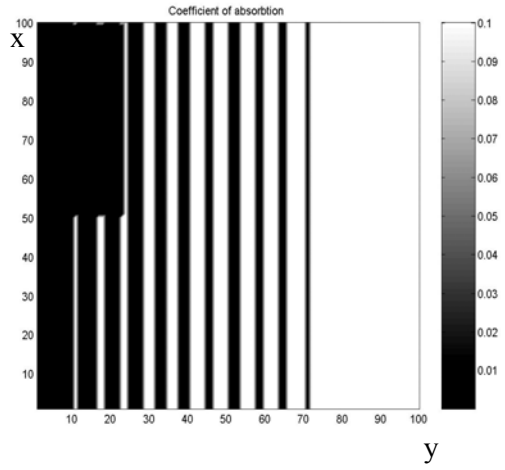


Рис. 11. Коэффициент поглощения в среде.

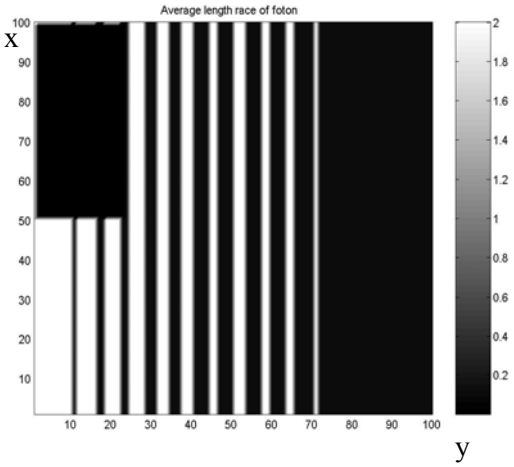


Рис. 12. Средняя длина свободного распространения фотона в среде.

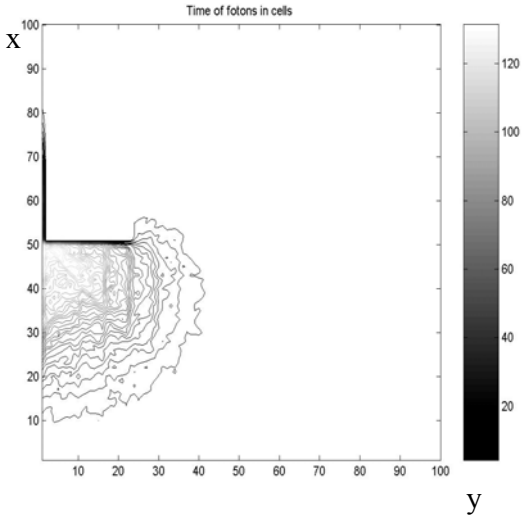


Рис. 13. Распределение энергии рассеянного сигнала в среде.

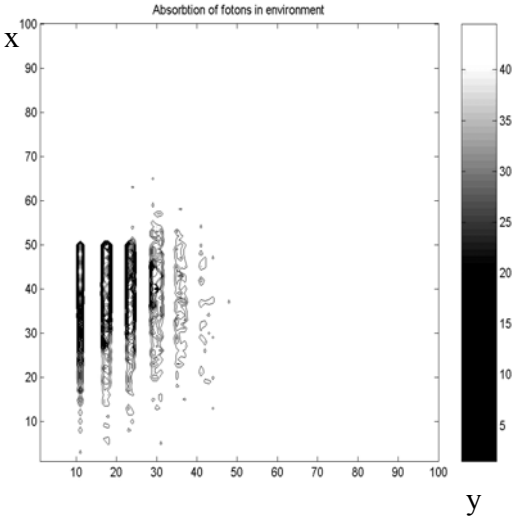


Рис. 14. Распределение энергии поглощенного сигнала в среде

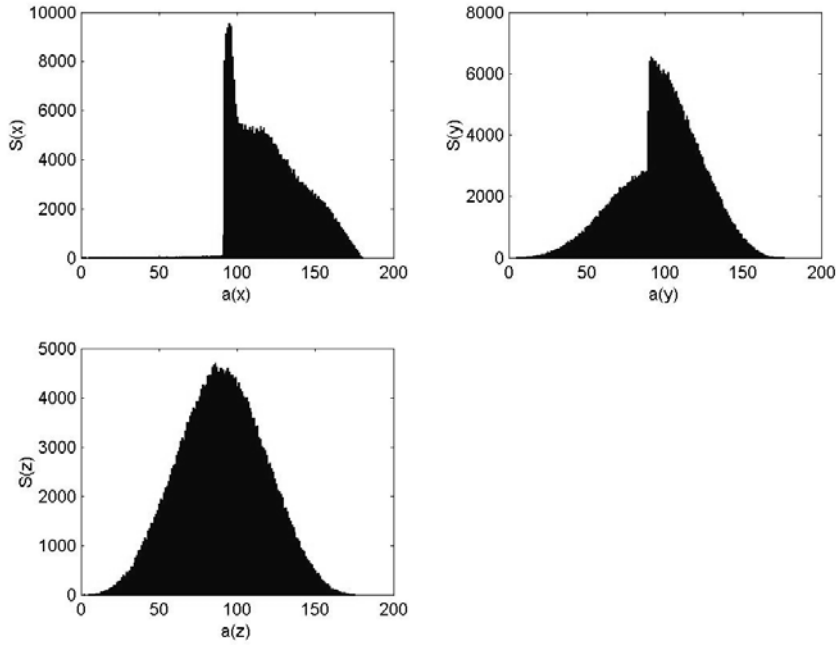


Рис. 15. Углы выхода фотонов из среды.

Влияние объекта видно особенно выразительно в плоскости у рис. 15. При угле рассеяния $\alpha_y = 90^o$ виден перепад в количестве вышедших фотонов. Такой результат можно объяснить тем, что фотоны попав в объект, имеют малую вероятность пройти через него. На рис. 13 отчетливо наблюдается попадание фотонов в область тени объекта, что соответствует физическому явлению дифракции. Такой результат объясняется тем, что фотоны в среде движутся хаотически и соответственно могут равновероятно переизлучиться в любом направлении, в том числе и в область тени.

Заключение

В работе создана численная модель многократного взаимодействия электромагнитной волны с дискретными неоднородностями, хаотически расположенными в случайной трехмерной дискретной среде.

Проведено моделирование процесса распространения волны в трехмерно — неоднородной случайной дискретной среде. Решена задача прохождения волны через случайную слоистую среду, содержащую объект в форме параллелепипеда. Исследованы зависимости энергий рассеянного и поглощенного сигнала от факторов, описывающих неоднородную структуру случайной дискретной среды. Проведены численные расчеты углового спектра многократно рассеянной волны в зависимости от параметров.

Анализ результатов численных экспериментов показал существенное отличие контрастности изображения для случая поглощенной энергии сигнала и для энергии рассеянного сигнала в среде. На основе анализа полу-

ченных результатов можно сделать вывод о более контрастных изображениях объектов для случая поглощенной энергии сигнала, так как энергия рассеянного сигнала распределена в большем пространстве.

Литература

1. *Ермаков С. М., Михайлов Г. А.* Курс статистического моделирование.— М.: Наука, 1976.— 319 с.
2. *Соболь И. М.* Численные методы Монте–Карло.— М.: Наука, 1973.— 311 с.
3. *Апресян Л. А. Кравцов Ю. А.* Теория переноса излучения.— М: Наука, 1983.— 216 с.
4. *Бреховских Л. М.* Волны в слоистых средах.— М.: Наука, 1973.— 505 с.
5. *Crook A. W.* The reflection and transmission of light by any system of paralel isotropic films // J. Opt. Soc. Am.
6. *Спицын В. Г.* Моделирование рассеяния радиоволн на возмущениях ионосферной плазмы, создаваемых космическим аппаратом.— Томск: «STT», 2002.— 174 с.

УДК 629.7.054(082)

ИСПОЛЬЗОВАНИЕ ПАКЕТА MATLAB В СПЕЦИАЛИЗИРОВАННОЙ ИНТЕГРИРОВАННОЙ СРЕДЕ

Шатский М. А.
Московский государственный технический университет им. Н. Э. Баумана, Москва, e-mail: mshatski@mtu-net.ru

Современный летательный аппарат (ЛА) представляет собой сложный нестационарный, нелинейный динамический объект управления, описываемый математической моделью высокого порядка. Его характеристики могут существенно изменяться в зависимости от режима полета: высоты, скорости и других параметров. Значительный разброс в области полета статических и динамических характеристик не позволяет создать систему стабилизации (ССт) ЛА с постоянными значениями параметров автопилота. При разработке системы стабилизации с переменными настройками автопилота процесс определения законов изменения настроек в зависимости от режима полета ЛА является достаточно трудоемким и длительным. Разработчикам ССт ЛА приходится работать с большими массивами данных, выполнять большое количество рутинных операций, что увеличивает время и стоимость проектирования системы стабилизации. На начальных стадиях расчета вновь проектируемых систем стабилизации ЛА, а также при их модернизации и доработке, с целью сокращения стоимостных и временных затрат процесс определения законов изменения настроек автопилота в зависимости от режима полета ЛА целесообразно автоматизировать с помощью специализированных программных продуктов.

Постановка задачи и исходные данные

Рассмотрена задача синтеза системы стабилизации ЛА применительно к продольному каналу осесимметричного маневренного ЛА нормальной аэродинамической схемы в линейной постановке [2]. Синтез системы стабилизации ЛА производится из условия удовлетворения заданным техническим требованиям (ТТ) во всей области полета (ОП) ЛА.

Процедура расчета

Основные этапы решения поставленной задачи, показанные на рис. 1, реализованы в пакете «Автоматизация расчета систем стабилизации летательных аппаратов» (АРСС). *Пакет АРСС позволяет*:

- получать из массива аэродинамических характеристик ЛА динамические характеристики, передаточные функции и частотные характеристики ЛА для любого режима полета;
- находить в области полета ЛА зоны постоянных настроек автопилота, при которых удовлетворяются предъявляемые к ССт ТТ;
- использовать различные алгоритмы параметрической оптимизации как для отдельных контуров, так и для всей системы;
- получать законы изменения параметров ССт в зависимости от условий полета ЛА в классе кусочно-линейных регуляторов;
- исследовать поведение ССт в различных условиях полета ЛА.

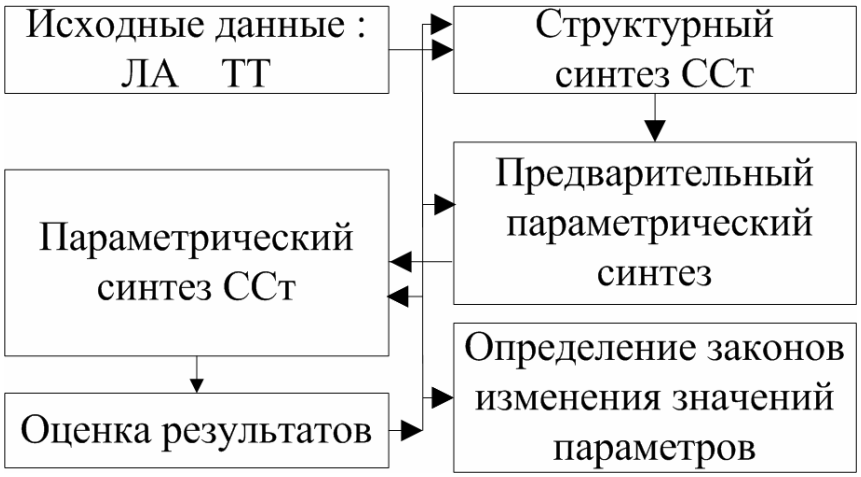


Рис. 1. Основные этапы решения задачи.

При работе с пакетом АРСС пользователь после *ввода исходных данных* проводит *структурный синтез* ССт, заключающийся в выборе структурной схемы из набора типовых, наиболее часто используемых схем. Выбор структурной схемы определяется динамикой ЛА, методом наведения, предъявляемыми к системе ТТ, техническими возможностями и поддерживается справочной информацией, предоставляемой программой.

На следующем этапе работы проводится *предварительный параметрический синтез* системы стабилизации, основная цель которого — выбор динамических характеристик рулевого привода. Для этого используется метод стандартных коэффициентов с учетом ряда ограничений, характерных для конкретных типов рулевых приводов.

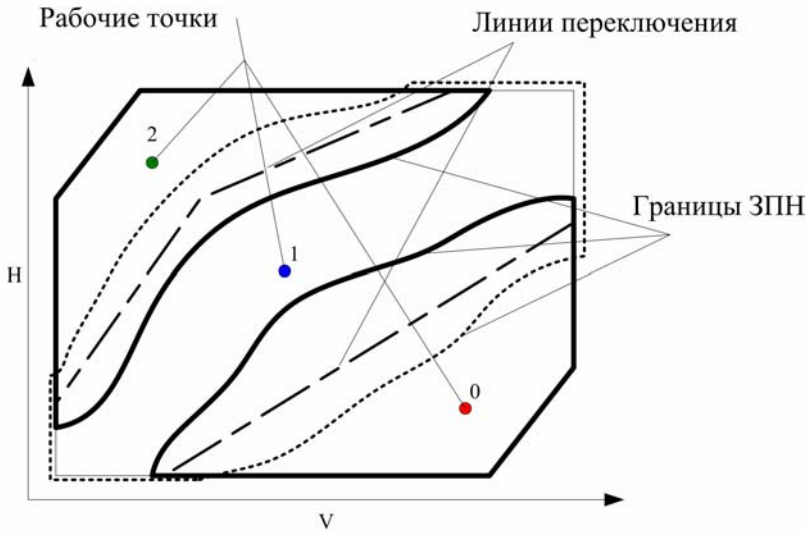


Рис. 2. Область полета.

Параметрический синтез системы стабилизации (рис.2) основан на разбиении всей ОП ЛА на зоны постоянных настроек (ЗПН), в которых ССт будет удовлетворять предъявляемым к ней ТТ, имея постоянные настройки. Этап параметрического синтеза представляет собой итеративный процесс, который начинается с выбора начального расчетного режима полета ЛА (начальной рабочей точки) и параметрической оптимизации системы с использованием различных методов. После параметрической оптимизации производится расчет границы ЗПН. Затем выбирается следующая рабочая точка в части ОП ЛА, не охваченной ЗПН, и процедура повторяется до тех пор, пока вся ОП ЛА не будет перекрыта ЗПН.

Задача *параметрической оптимизации* ССт, сводится к следующему. Технические требования к системе стабилизации имеют вид ограничений-неравенств:

- время переходного процесса $t_{пп\ min} \le t_{пп} \le t_{пп\ max}$,
- перерегулирование $\sigma \le \sigma_{доп}$,
- статическая ошибка по выходной координате $|\varepsilon_{стат}| \le |\varepsilon_{доп}|$.

При известной структуре ССт требуется уточнение значений параметров с целью выполнения предъявляемых к системе технических требований. Помимо этого, может ставиться задача минимизации одного из показателей функционирования системы.

Для параметрической оптимизации ССт может быть использована процедура условной минимизации функции нескольких переменных:

$$\min_{(k)}\{f(\mathbf{k}) : q_i(\mathbf{k}) \le 0,\ i = 1,2,...,I\},$$

где в качестве ограничений $q_i(\mathbf{k})$ используются показатели функционирования системы, являющиеся функциями вектора оптимизируемых параметров $\mathbf{k}$ ($q_i = p_i - B_i$; p_i — i-я характеристика системы, ограниченная сверху ТТ — предельно допустимыми значениями B_i).

В качестве скалярного критерия $f(\mathbf{k})$ может использоваться один из показателей функционирования системы или константа. В этом случае процедура оптимизации будет остановлена при удовлетворении условий $q_i(\mathbf{k})<0$, т. е. *при получении настроек автопилота при которых система стабилизации, удовлетворяет ТТ*. При этом целесообразно ввести запас $F_{iзад}$, позволяющий учесть неточности моделей ЛА и элементов ССт:

$$\min_k \{ f(k) : q_i(k) + F_{i\,зад} \le 0,\ \ i = 1,2,..., I \} \ .$$

Предлагаемая постановка задачи позволяет использовать стандартные процедуры численной оптимизации, реализованные в математических программных продуктах, например, в ППП MATLAB.

Полученные ЗПН позволяют построить линии переключения настроек автопилота, которые находятся пользователем на основании рассчитанных границ ЗПН и различных дополнительных соображений (например, ограничений на сложность алгоритмов управления). Линии переключения определяют *законы изменения значений параметров* автопилота.

Принципы построения интегрированной среды проектирования систем стабилизации

Современный уровень развития компьютерных технологий дает возможность создавать средства автоматизации проектирования, позволяющие повысить эффективность работы пользователя, освободить его от рутинных операций и, следовательно, сократить время и стоимость разработки, повысить ее надежность за счет рассмотрения большего числа возможных вариантов решения задачи. Поэтому современные средства разработки систем стабилизации должны включать в себя полный набор инструментов, которые могут потребоваться разработчику в процессе работы, т. е. являются интегральной средой (ИС). Среди элементов, которые должна содержать ИС можно выделить следующие:

- вычислительное ядро, включающее в себя как стандартные, так и специально разработанные для решения конкретных задач процедуры анализа, синтеза, оптимизации и моделирования динамических систем;
- инструменты, связанные с документированием результатов исследований. При этом желательна максимальная автоматизация всех действий, связанных с составлением отчетов и получением распечаток результатов работы;
- оперативная справочная система, содержащая общие сведения о принципах работы ИС, описание используемых алгоритмов, рекомендации по их использованию, описание интерфейса;
- инструменты, предназначенные для обучения работе с системой.

ИС должна обладать интуитивно понятным интерфейсом, построенным в соответствии современными стандартами в области разработки гра-

фических интерфейсов пользователя. Одним из важнейших свойств ИС является расширяемость, т. е. возможность легкого подключения дополнительных программных модулей, возможность настройки под конкретного пользователя и конкретную задачу.

Структура ИС АРСС

Рассмотренные положения использовались при реализации ИС АРСС. Структура ИС и использованные компьютерные технологии показаны на рис.3. При создании основной программы ИС АРСС была использована среда программирования Delphi 6. Среда Delphi представляет собой средство быстрой разработки приложений. Поэтому ее использование при разработке программных продуктов позволило максимально уменьшить затраты на проектирование пользовательского интерфейса. Для параметрической оптимизации использовались функции ППП MATLAB (Optimization Toolbox). Также использовались и другие возможности ППП MATLAB, связанные с синтезом, моделированием и анализом динамических систем.

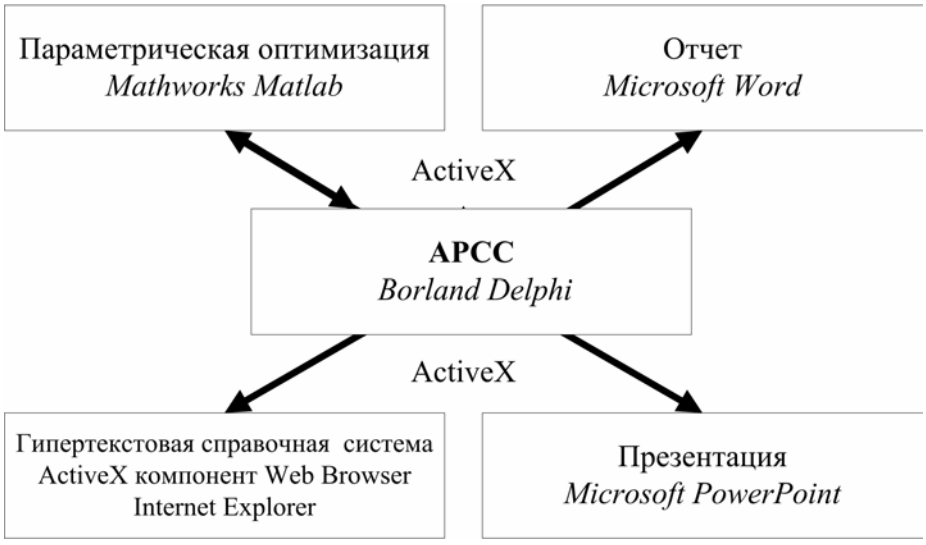


Рис. 3 Структура ИС АРСС.

Функции автоматизации документирования выполнены средствами текстового редактора Microsoft Word, который дает возможность получать отчет о выполненных расчетах, содержащий необходимые графики и числовые данные в виде текста и таблиц с заголовками и нумерацией.

В процессе работы пользователю предоставляются справочные данные и сведения теоретического характера с помощью гипертекстового справочного пособия. Для этого используются ActiveX компонент WebBrowser и программа Microsoft Internet Explorer. Для демонстрации

возможностей системы АРСС и обучения работе с ней с помощью программы Microsoft PowerPoint разработана презентация.

Все компоненты ИС объединены с помощью интерфейса ActiveX.

Основная программа ИС АРСС имеет интерфейс многостраничного типа, каждая страница которого соответствует определенному этапу расчета системы стабилизации ЛА. При этом пользователь имеет возможность работать, не привязываясь к определенному сценарию расчета.

На основе ИС АРСС был разработан комплект методических материалов, предназначенный для проведения лабораторной работы по теме «Расчет систем стабилизации летательных аппаратов», входящей в состав соответствующего курса. В учебной версии ИС АРСС используются упрощенные алгоритмы расчета системы стабилизации ЛА и имеется расширенная гипертекстовая справочная система.

Использование системы АРСС при чтении лекционных курсов в высших и средних технических учебных заведениях позволит повысить эффективность подготовки специалистов, умеющих успешно разрабатывать и эксплуатировать современную технику.

Литература

1. *Архангельский А. Я.* Разработка прикладных программ для Windows в Delphi 5.— М.: Бином, 1999.
2. *Топчеев Ю. И., Потемкин В. Г., Иваненко В. Г.* Системы стабилизации.— М.: Машиностроение, 1974.
3. MATLAB. The Language of Technical Computing. Application Program Interface Guide. Version 5.— The MathWorks, Inc., 1999.
4. Thomas Coleman, Mary Ann Branch, Andrew Grace Optimization Toolbox For Use with MATLAB. User's Guide. Version 2.— The MathWorks, Inc., 1999.

УДК 004

ИСПОЛЬЗОВАНИЕ MATLAB ПРИ МОДЕЛИРОВАНИИ ПРОЦЕССА КОПАНИЯ ОДНОКОВШОВЫМ ЭКСКАВАТОРОМ С ОБРАТНОЙ ЛОПАТОЙ

Щербаков И. С.
Сибирская государственная автомобильно-дорожная академия, Омск,
e-mail:ischerbakov@sibadi.omsk.ru

При строительстве каналов мелиорации или траншей нефтепроводов точность копания, согласно СНиП, не должна превышать 5 сантиметров. Недобор грунта экскаватором оборачивается затратами на ручной труд и потерю времени. Для повышения точности копания экскаватора, особенно на последнем этапе — планировке, алгоритм работы системы должен обеспечивать угол резания ковша в пределах 50–100 градусов (оптимальные углы резания грунта) к плоскости резания. А так же управление углами поворотов стрелы, рукояти (и ковша) при перемещении ковша от дальней точки резания к ближней на заключительном этапе — планировании грунта.

При построении модели использовались системы однородных координат. Матрицы сдвигов и поворотов для нахождения искомых координат оборудования.

В статье рассмотрена система из Стрелы, Рукояти и Ковша перемещающаяся в плоскости *XOY* и поворачивающаяся вокруг оси 0Z. Центр Координат расположен в шарнире «Рама экскаватора — Стрела». Координата второго шарнира стрелы «Стрела - Рукоять» вычисляется сдвигом матрицы из Центра Координат *XYZO* вдоль оси *OX* на величину длины стрелы и умножив на угол поворота вокруг оси *OZ* заданный в виде матрицы поворота. В качестве матрицы координат ковша [8×3] используется матрица координат куба с размерами граней равными длине, высоте и ширине ковша, расположенному в центре координат *O*. Матрица сдвига [8×3] будет иметь первые восемь значений (координаты *X*) равными длине стрелы. Матрица поворотов ковша, стрелы, рукояти [3×3] вокруг оси *OZ* представляет специальную матрицу:

Z = [cos(zz) -sin(zz) 0; sin(zz) cos(zz) 0; 0 0 1]

Что бы получить координаты куба (условно представляющие ковш) сдвинутым на величину 2.73 (длина стрелы в метрах)по оси *X* необходимо: сложить матрицу куба «vert» с матрицей сдвига «axes» и сохранить результат в матрице «vert» для последующей передачи результата в другую систему координат.

vert[8x3] + axes[8x3] = vert[8x3]

axes =

```
 2.7300      0      0
 2.7300      0      0
 2.7300      0      0
 2.7300      0      0
 2.7300      0      0
 2.7300      0      0
 2.7300      0      0
 2.7300      0      0

vert =
      0         0       0
      0    1.2500       0
 1.2500    1.2500       0
 1.2500         0       0
      0         0  1.2500
      0    1.2500  1.2500
 1.2500    1.2500  1.2500
 1.2500         0  1.2500
```

Что бы получить координаты стрелы поднятой на угол —pi/8 необходимо умножить получившуюся матрицу «vert» на матрицу поворота Z

```
Z = [cos(zz) -sin(zz) 0; sin(zz) cos(zz) 0; 0 0 1];
Z =

 0.9239   0.3827        0
-0.3827   0.9239        0
 0         0       1.0000

Z * vert =

 2.5222   1.0447        0
 2.0438   2.1996        0
 3.1987   2.6779        0
 3.6770   1.5231        0
 2.5222   1.0447   1.2500
 2.0438   2.1996   1.2500
 3.1987   2.6779   1.2500
 3.6770   1.5231   1.2500
```

Полученый результат сохраняется в матрице «vert» и «strela_smesh».

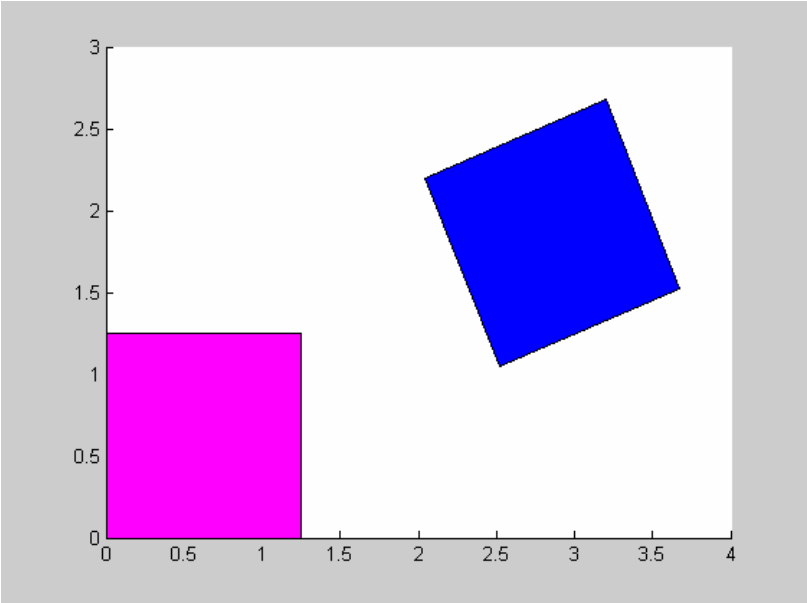


Рис. 1. Символический рисунок поднятой Стрелы.

Что бы получить координаты Рукояти с Ковшом, необходимо матрицу координат Ковша сложить с матрицей сдвига «axes» имеющей значения длины Рукояти, повернуть получившуюся матрицу на угол наклона Рукояти «dr_r», сложить с углом наклона Стрелы «dr_s» и совместить начало Рукояти с концом Стрелы. То есть сложить значения матриц «vert» и «strela_smesh»

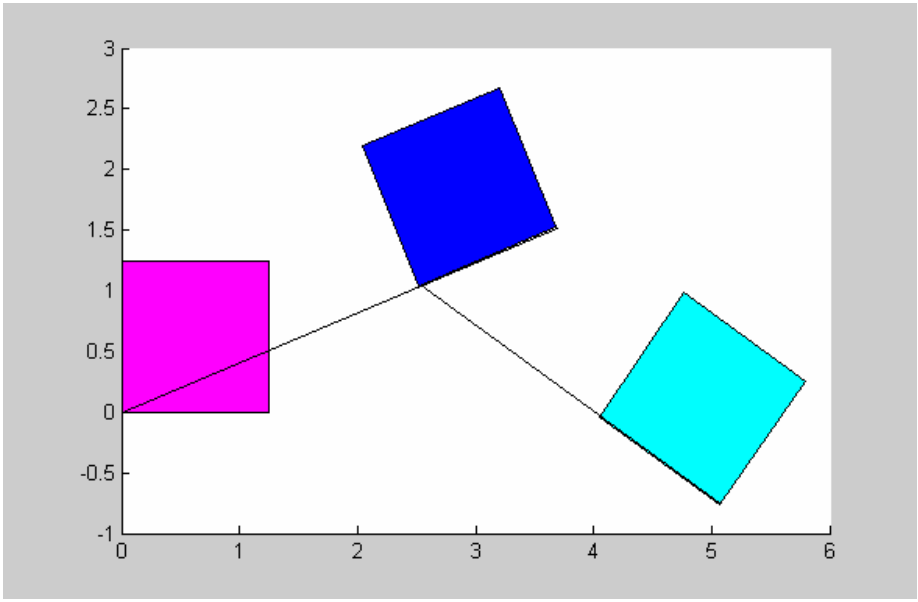


Рис. 2. Символический рисунок поднятой Стрелы и опущенной Рукояти. Квадрат закрашенный цветом «циан» обозначает Ковш. Нижняя грань с координатами X=5.0664 Y=-0.7612 Z=0(1.25) — режущая кромка Ковша

```
vert =

    4.0471   -0.0377         0
    4.7706    0.9816         0
    5.7899    0.2581         0
    5.0664   -0.7612         0
    4.0471   -0.0377    1.2500
    4.7706    0.9816    1.2500
    5.7899    0.2581    1.2500
    5.0664   -0.7612    1.2500
```

Координаты ковша

Далее производится анализ, коснулся ли зуб Ковша (координата vert[16]) линии планирования (стал меньше переменной glubina_kopanija = —1), если нет, то производится дальнейшее опускание Рукояти (изменение угла наклона Рукояти dr_r с заданной точностью опускания a_dr_r). Если координата vert[16] стала меньше переменной glubina_kopanija = —1, значит в этом цикле произошло пересечение линии планирования и необходимо передать управление на Стрелу — поднять ее на некоторый угол dr_s и продолжить цикл опускания Рукояти до достижения уровня планирования грунта.

При заданной точности в переменной a_dr_r = 0.01, вычисляемая точность планирования —0.62 сантиметра.

Литература

1. Динамика управления роботами / Под ред. Е. И. Юревича.— М.: Наука, 1984.
2. *Мэтьюз Д. Г., Финк К. Д.* Численные методы. Использование MATLAB. — 3-е изд.— М., СПБ., К: Изд. дом «Вильямс», 2001.

Приложение. Фрагмент программы

```
clear all;  n=1;    count = 0;  %view(0,0);
x=0; y=0; z=0;                      % Начало координат
h=1.25; l=1.25; w=1.25;       % Высота Длина Ширина Ковша
dlina_strely = 2.73;              % Длина стрелы
dlina_rukojaty = 1.87;          % Длина рукояти
glubina_kopanija = -1;            % Глубина копания
fac=[1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8];
% Углы поворота Стрелы Вверх ->  3 4 6 8 32 32 8 6 4 3 <- Вниз
dr_s = 8;       % От 8 до 2 поднятие стрелы
% Первоначально поднимаем Рукоять вверх и начинаем опускать на
%  заданый  уровень.  В  последствии  пользуемся  сохраненным
%  значением угла
% поворота рукояти в переменной  dr_r
% От 1000 до 1 опускание рукояти до уровня планирования
```

```
dr_r  = 1;    a = 1; rukojat_90_grad = 1;
start = 0;  vert(1) = 1;  vert(4) = 4;
while dr_s >= 2 & rukojat_90_grad > 0
   %-------------  Рисуем Ковш и задаем начальные данные  ----------
   zz=0.0;            % Начальный угол поворота для ковша
   % Поворот вокруг оси OZ
   Z     =     [cos(zz)     -sin(zz)     0;     sin(zz)     cos(zz)     0;     0     0     1];
   %  положение ковша в пространстве
   x=0; y=0; z=0;      % Начало координат
   % матрица координат Ковша
   vert = [x y z; x y z; x y z; x y z; x y z; x y z; x y z; x y z];
   % Создание матрицы линейных перемещений
   axes = zeros(8, 3); % Для совместимости с матрицей vert [8x3]
   % матрица размеров ковша
   leng = [0 0 0; 0 w 0; l w 0; l 0 0; 0 0 h; 0 w h; l w h; l 0 h];
   vert = (vert + leng)*Z;     % Координаты вершин кубика в центре
   % Запомнили на будущее координаты  ковша в центре
   vert_center = vert;
   patch('faces',fac,'vertices',vert,'FaceColor','m');
   % --------------------Создаем Стрелу. -----------------
   axes(1:8) = dlina_strely;       % Задаем длину Стрелы
   vert = vert + axes;             % Выдвигаем на 4 единицы по X.
   % ------------------ Подьем Стрелы ------------------------------
   y_s = -pi/dr_s;           % Угол подьема Стрелы
   zz = y_s;
   % Поворот вокруг оси 0Z
   Z = [cos(zz) -sin(zz) 0; sin(zz) cos(zz) 0; 0 0 1];
   % Повернули Стрелу (подняли на несколько градусов)
   vert = vert*Z;
   patch('faces',fac,'vertices',vert,'FaceColor','b');
   % Матрица хранения смещения конца Стрелы
   strela_smesh  = zeros(8, 3);
   % Запоминаем положение конца Стрелы
   % после поднятия
   strela_smesh(1:8)  = vert(1);
   strela_smesh(9:16) = vert(9);
   strela_smesh(17:24)= vert(17);

   % ------------------ Поворот Рукояти ------------------------------
   % Задаем длину Рукояти и угол наклона в Центре Координат
   axes(1:8) = dlina_rukojaty;     % Длина Рукояти 6 единиц по X
   a_dr_r = 0.01;                  % Точность опускания Рукояти
   % Опускаем Рукоять с Ковшом пока не достигли дна траншеи
   % или Рукоять не достигла наклона 90 град
   while vert(16) > glubina_kopanija  & rukojat_90_grad > 0
      dr_r = dr_r + a_dr_r; % Опускание + Точность поворота Рукояти
      % Угол складывается из поворота Рукояти + Стрелы
      zz = dr_r + y_s;
      % Поворот вокруг Z
      Z = [cos(zz) -sin(zz) 0; sin(zz) cos(zz) 0; 0 0 1];
```

```
    vert = vert_center + axes;  % Выдвигаем на .. единицы по X.
    % Поворачиваем Рукоять длиной .. ед. вокруг Z
    vert = vert*Z;

    % помещаем Рукоять на место по запомненому  смещению
    vert = vert + strela_smesh;
    pause(0.3);     % даем прорисоваться рисунку одну секунду
    % проверка положения Рукояти (90 град)
    rukojat_90_grad = (vert(1) - strela_smesh(1));
end
% Цикл закончен - Зуб Ковша коснулся дна траншеи
patch('faces',fac,'vertices',vert,'FaceColor','r');
pause(0.3);         % даем прорисоваться рисунку 0.3 секунды
% Поднимаем стрелу на угол dr_s с точностью (0.1)
dr_s = dr_s - 0.1;
% и   повторяем цикл с новым значением угла подьема стрелы
end
```

УДК 537.81

РЕШЕНИЕ НЕЛИНЕЙНОЙ ЗАДАЧИ МАГНИТОСТАТИКИ В СИСТЕМЕ FEMLAB

Шмелев В. Е.
Владимирский государственный университет, Владимир,
e-mail:shmelioff@vpti.vladimir.ru

Введение

Система конечноэлементных расчетов FEMLAB дает исследователю богатые возможности моделирования разнообразных физических полей. Большой интерес представляют в этой системе прикладные режимы расчета магнитостатических полей в электромагнитных системах с нелинейными магнитными материалами. Все эти прикладные режимы основаны на векторном уравнении магнитостатики относительно векторного магнитного потенциала. Задание нелинейных магнитных свойств в вычислительных моделях вызывает некоторые затруднения у пользователей системы FEMLAB. Затруднения вызваны тем, что ее графический интерфейс всегда показывает пользователю параметры магнитных свойств в коэффициентной форме (даже в тех случаях, когда выбрана генеральная форма решения краевой задачи).

В статье нелинейная задача магнитостатики будет рассматриваться на примере упрощенной электромагнитной системы трехфазного трансформатора, работающего в статическом режиме. Для некоторого подтверждения достоверности моделирования расчеты интегральных параметров магнитного поля будем выполнять цепными и полевыми методами, а затем сравним результаты.

Электротехническая постановка задачи и цепной расчет

На рис. 1 показано схематичное изображение электромагнитной системы трехфазного трансформатора. Геометрические параметры этой системы, а также токи и числа витков обмоток имеют следующие значения: l = 0,28 м; h = 0,234 м; a = 0,05 м; b = 0,05 м; толщина магнитопровода $h_{\text{М}}$ = 0,064 м; $w_1 = w_2 = w_3 = 500$. Токи в первичных обмотках i_1, i_2, i_3 пусть определяются выражением MATLAB 0.3*real(exp(1i*([0 -2*pi/3 2*pi/3] +pi/4))) А, т. е. i_1 = 0.21213 А; i_2 = 0.077646 А; i_3 = -0.28978 А.

По условию задачи требуется определить распределение векторного магнитного потенциала и магнитной индукции в электромагнитной системе трансформатора. Динамическими эффектами можно пренебречь. Тре-

буется определить также магнитное потокосцепление во всех обмотках, если во вторичных обмотках намотано по 80 витков.

Основная кривая намагничивания материала магнитопровода задана таблично (табл. 1).

Таблица 1.

H, А/м	10	20	50	70	100	200	500	1000	2000
B, Тл	0.03	0.1	0.38	0.58	0.66	0.9	1.18	1.29	1.326

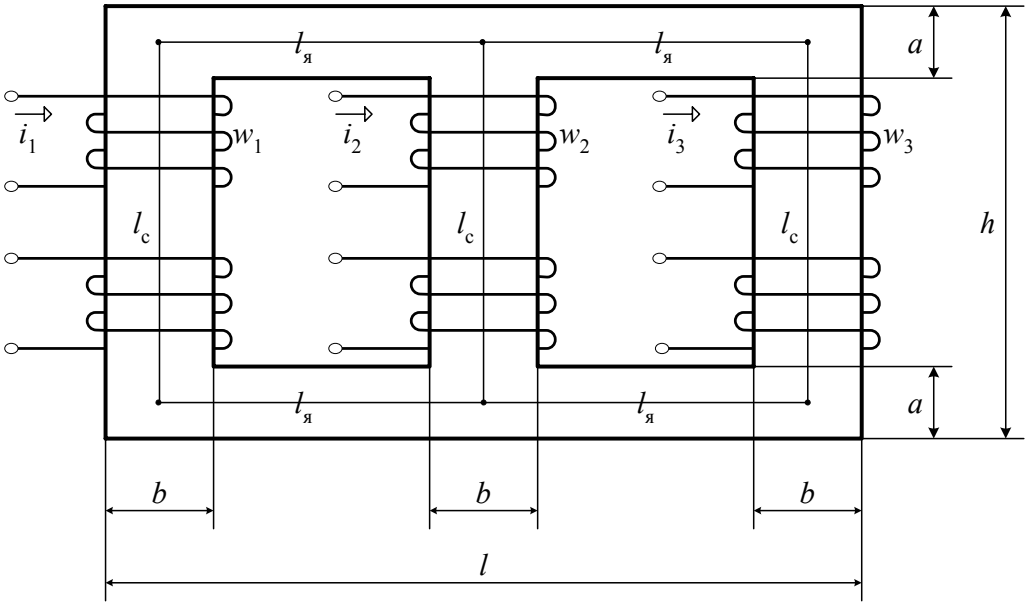


Рис. 1. Схематичное изображение электромагнитной системы трансформатора.

Цепной расчет начинается с построения схемы замещения магнитной цепи. Она изображена на рис. 2. Прежде чем рассчитывать вебер-амперные характеристики ветвей, нужно определить их эффективные геометрические параметры.

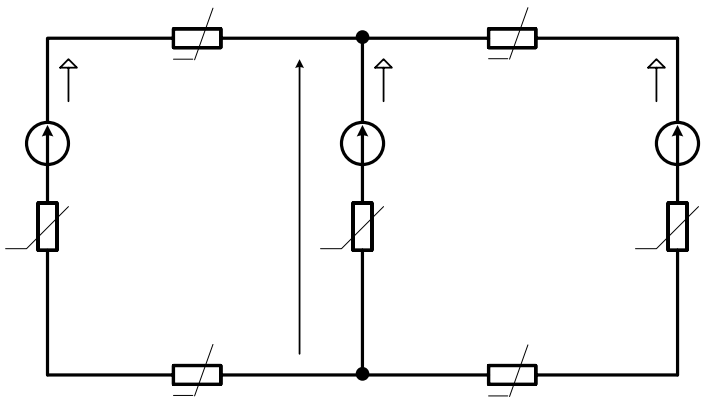


Рис. 2. Схема замещения магнитной цепи.

Эффективная длина стержня $l_с = h–a = 0.184$ м. Эффективная длина ярма $l_я = (h–a)/2 = 0.115$ м. Площади поперечного сечения участков магнитной цепи: $S_с = b \cdot h_М = 3{,}2 \cdot 10^{-3}$ м^2; $S_я = a \cdot h_М = 3{,}2 \cdot 10^{-3}$ м^2.

Вебер-амперная характеристика участка цепи получается из основной кривой намагничивания путем замены значений напряженности магнитного поля H на магнитное напряжение $H \cdot l$-F, где l — длина участка (ветви), $F = i \cdot w$ — намагничивающая сила обмотки, а также замены значений магнитной индукции B на магнитный поток $B \cdot S$, где S — площадь поперечного сечения ветви.

Интерполяция основной кривой намагничивания и расчет вебер-амперных характеристик ветвей могут быть осуществлены путем выполнения следующей последовательности операторов MATLAB:

```
Hp=[10 20 50 70 100 200 500 1000 2000];
Bp=[0.03 0.1 0.38 0.58 0.66 0.9 1.18 1.29 1.326];
H=[-Hp(end:-1:1) 0 Hp];
B=[-Bp(end:-1:1) 0 Bp];
Hi=-2000:5:2000;
figure(1)
Bi=interp1(H,B,Hi,'cubic');
plot(Hi,Bi)
grid on
I=0.3*real(exp(1i*([0 -2*pi/3 2*pi/3] +pi/4))); % Токи первичных обмоток, А
w=500; % Числа витков обмоток
ls=0.184; lz=0.115; Sv=3.2E-3; % Длина стержня и ярма, площади ветвей
U=(ls+2*[lz;0;lz])*Hi-repmat(w*I.',1,size(Hi,2)); % Напряжения ветвей, А
Fi=Bi*Sv; % Магнитные потоки ветвей, Вб
figure(2)
plot(U(1,:),Fi,'k-',U(2,:),Fi,'b-',U(3,:),Fi,'g-')
hold on
% Для графических расчетов надо вебер-амперные характеристики
% привести к одним и тем же значениям магнитных напряжений
Fii=zeros(size(U));
Fii(1,:)=interp1(U(1,:),Fi,U(2,:),'cubic');
Fii(2,:)=Fi;
Fii(3,:)=interp1(U(3,:),Fi,U(2,:),'cubic');
plot(U(2,:),sum(Fii),'r-') % Суммарный магнитный поток приравниваем к 0
grid on
hh=findobj(allchild(0),'type','line');
set(hh,'linewidth',2);
```

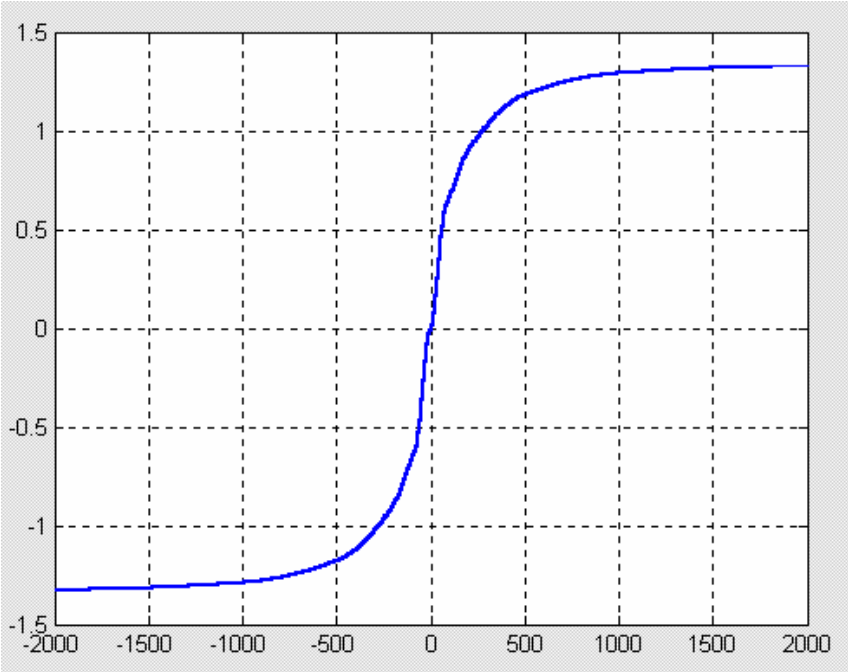


Рис. 3. Основная кривая намагничивания электротехнической стали.

В результате выполнения вычислительного сценария будет создано две фигуры MATLAB. В первой фигуре будет график основной кривой намагничивания (рис.3). Во второй фигуре (рис.4) — графики вебер-амперных характеристик ветвей (черный — ветвь 1, синий — ветвь 2, зеленый — ветвь 3). Красным цветом показана вебер-амперная характеристика суммарного магнитного потока ветвей, который в соответствии с первым законом Кирхгофа должен быть равен нулю. Пересечение красной кривой с осью нулевого значения потока даст значение магнитного напряжения ветвей: U_2 = -31.5527 А. Теперь построим вертикальную прямую для этого значения магнитного напряжения. Пересечение этой прямой с вебер-амперными характеристиками ветвей даст значения магнитных потоков ветвей. Вертикальную прямую можно построить следующим оператором MATLAB:

plot(-31.5527*[1 1],1.2E-2*[-1 1],'r-')

Итак, рис. 4 дает значения магнитных потоков ветвей: Φ_1 = 2.7667E-3 Вб; Φ_2 = 8.864E-4 Вб; Φ_3 = -3.6531E-3 Вб. Если эти числа поделить на площади ветвей, получим средние значения магнитной индукции ветвей: B_1 = 0.86459 Тл; B_2 = 0.277 Тл; B_3 = -1.1416 Тл. Оператором MATLAB

(-31.5527+I*w)./(ls+2*[lz;0;lz]).'

вычислим средние значения напряженности магнитного поля ветвей: H_1 = 179.98 А/м; H_2 = 39.512 А/м; H_3 = -426.19 А/м.

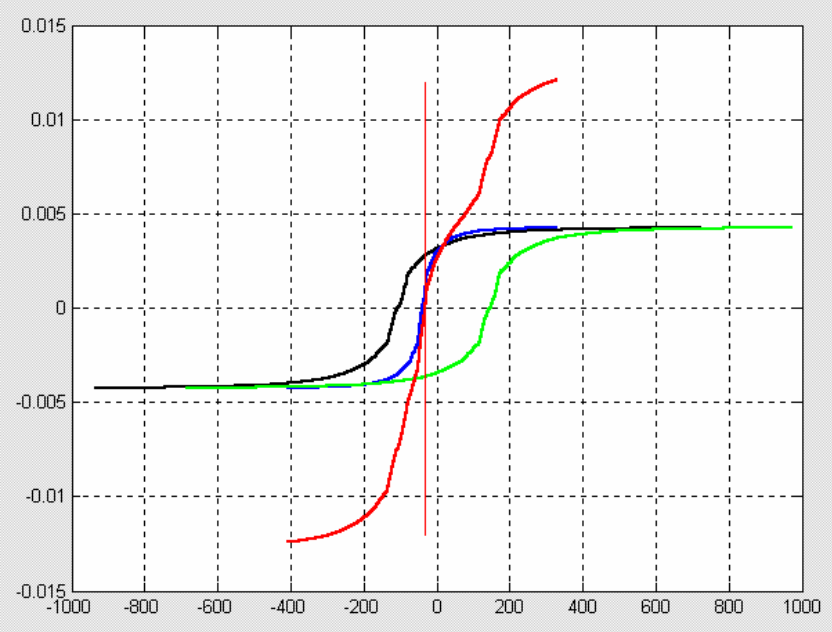


Рис. 4. Графический расчет состояния магнитной цепи

Трехмерная векторная краевая задача магнитостатики и ее реализация в одном из прикладных режимов FEMLAB

Для начала рассмотрим коэффициентную форму трехмерной краевой задачи магнитостатики. Закон полного тока в дифференциальной форме имеет вид

$$\mathrm{rot}\,\boldsymbol{H} = \delta, \qquad (1)$$

где $\boldsymbol{H}$ — вектор напряженности магнитного поля; δ — плотность тока. Уравнение (1) дополняется уравнением материальной связи между векторами напряженности магнитного поля и магнитной индукции $\boldsymbol{B}$. Это уравнение в линеаризованном виде записывается так:

$$\boldsymbol{H} = \nu_a \cdot \left(\boldsymbol{B} - \boldsymbol{B}_r\right) = \nu_a \boldsymbol{B} - \nu \boldsymbol{M}_r, \qquad (2)$$

где ν_a — величина, обратная абсолютной магнитной проницаемости среды (иначе ее называют удельным магнитным сопротивлением), $\boldsymbol{B}_r$ — вектор остаточной магнитной индукции, н — величина, обратная относительной магнитной проницаемости, $\boldsymbol{M}_r$ — вектор остаточной намагниченности вещества. Поле вектора магнитной индукции не имеет стоков и истоков ($\mathrm{div}\,\boldsymbol{B} = 0$), поэтому его можно выразить через векторный магнитный потенциал $\boldsymbol{A}$:

$$\boldsymbol{B} = \mathrm{rot}\,\boldsymbol{A}. \qquad (3)$$

Подставив (3) в (2), а затем (2) в (1), получим векторное уравнение магнитостатики в коэффициентной форме:

$$\mathrm{rot}\left(\nu_a\ \mathrm{rot}\,\mathbf{A} - \nu\ \ \mathbf{M}_r\right) = \delta. \qquad (4)$$

Считается, что для обеспечения единственности решения уравнения (4) необходимо, кроме граничных условий, введение условия калибровки,

определяющего дивергенцию векторного потенциала [1]. Однако в системе FEMLAB условие калибровки совсем не применяется в магнитостатических прикладных режимах. Это означает, что сама конечноэлементная технология несет в себе неявно заданное условие калибровки.

В декартовых координатах векторный потенциал имеет три компоненты:

$$\mathbf{A} = \mathbf{1}_x A_x + \mathbf{1}_y A_y + \mathbf{1}_z A_z;$$

$$\operatorname{rot}\mathbf{A} = \left(\mathbf{1}_x \frac{\partial}{\partial x} + \mathbf{1}_y \frac{\partial}{\partial y} + \mathbf{1}_z \frac{\partial}{\partial z}\right) \times \left(\mathbf{1}_x A_x + \mathbf{1}_y A_y + \mathbf{1}_z A_z\right) =$$

$$= \mathbf{1}_x \left(\frac{\partial}{\partial y} A_z - \frac{\partial}{\partial z} A_y\right) + \mathbf{1}_y \left(\frac{\partial}{\partial z} A_x - \frac{\partial}{\partial x} A_z\right) + \mathbf{1}_z \left(\frac{\partial}{\partial x} A_y - \frac{\partial}{\partial y} A_x\right)$$

$$\operatorname{rot}\left(\nu_a \operatorname{rot}\mathbf{A}\right) = \mathbf{1}_x \left(\frac{\partial}{\partial y}\left(\nu_a \frac{\partial}{\partial x} A_y - \nu_a \frac{\partial}{\partial y} A_x\right) - \frac{\partial}{\partial z}\left(\nu_a \frac{\partial}{\partial z} A_x - \nu_a \frac{\partial}{\partial x} A_z\right)\right) +$$

$$+\mathbf{1}_y \left(\frac{\partial}{\partial z}\left(\nu_a \frac{\partial}{\partial y} A_z - \nu_a \frac{\partial}{\partial z} A_y\right) - \frac{\partial}{\partial x}\left(\nu_a \frac{\partial}{\partial x} A_y - \nu_a \frac{\partial}{\partial y} A_x\right)\right) +$$

$$+\mathbf{1}_z \left(\frac{\partial}{\partial x}\left(\nu_a \frac{\partial}{\partial z} A_x - \nu_a \frac{\partial}{\partial x} A_z\right) - \frac{\partial}{\partial y}\left(\nu_a \frac{\partial}{\partial y} A_z - \nu_a \frac{\partial}{\partial z} A_y\right)\right).$$

Если в уравнении (4) заменить векторное поле ***A*** на три скалярных поля A_x, A_y, A_z, то получится скалярно-матричная форма уравнения магнитостатики:

$$-\operatorname{div}\left(\mathbf{c} \cdot \operatorname{grad} u - \boldsymbol{\gamma}\right) = f, \tag{5}$$

где $u = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}$ — матрица-столбец трех рассчитываемых скалярных полей;

$$\mathbf{c} = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \nu_a & 0 \\ 0 & 0 & \nu_a \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ -\nu_a & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\nu_a & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & -\nu_a & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} \nu_a & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \nu_a \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -\nu_a & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & -\nu_a \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\nu_a \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} \nu_a & 0 & 0 \\ 0 & \nu_a & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix};$$

c — матрица тензоров магнитных сопротивлений (здесь для краткости тен-

зоры показаны в виде матриц декартовых компонентов); в случае анизотропных магнитных свойств вещества матрица $\boldsymbol{c}$ имеет более сложный вид;

$\boldsymbol{\gamma} = \begin{bmatrix} 0 - \mathbf{1}_y \nu M_{rz} + \mathbf{1}_z \nu M_{ry} \\ \mathbf{1}_x \nu M_{rz} + 0 - \mathbf{1}_z \nu M_{rx} \\ -\mathbf{1}_x \nu M_{ry} + \mathbf{1}_y \nu M_{rx} + 0 \end{bmatrix}$ — матрица-столбец векторных полей, эквивалентирующих векторное поле остаточной намагниченности вещества; в терминах обозначений FEMLAB она представляется в виде

$$\boldsymbol{\gamma} = \begin{bmatrix} 0 & -\nu M_{rz} & \nu M_{ry} \\ \nu M_{rz} & 0 & -\nu M_{rx} \\ -\nu M_{ry} & \nu M_{rx} & 0 \end{bmatrix};$$

$f = \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix}$ — матрица-столбец декартовых компонентов плотности тока.

Скалярно-матричное уравнение (5) «зашито» в магнитостатический прикладной режим FEMLAB.

Теперь уравнение математической физики относительно векторного магнитного потенциала рассмотрим в генеральной форме, чтобы обосновать способы задания нелинейных магнитных свойств вещества. Векторной генеральной формой уравнения магнитостатики является закон полного тока (1). Это уравнение можно привести к скалярно-матричной форме, выразив rot $\boldsymbol{H}$ через дивергенцию матрицы-столбца трех векторных полей, связанных с напряженностью магнитного поля:

$$\operatorname{rot} \boldsymbol{H} = \mathbf{1}_x \left(\frac{\partial}{\partial y} H_z - \frac{\partial}{\partial z} H_y \right) + \mathbf{1}_y \left(\frac{\partial}{\partial z} H_x - \frac{\partial}{\partial x} H_z \right) + \mathbf{1}_z \left(\frac{\partial}{\partial x} H_y - \frac{\partial}{\partial y} H_x \right) =$$

$$= \begin{bmatrix} \mathbf{1}_x & \mathbf{1}_y & \mathbf{1}_z \end{bmatrix} \cdot \operatorname{div} \begin{bmatrix} 0 + \mathbf{1}_y H_z - \mathbf{1}_z H_y \\ -\mathbf{1}_x H_z + 0 + \mathbf{1}_z H_x \\ \mathbf{1}_x H_y - \mathbf{1}_y H_x + 0 \end{bmatrix}$$

Если ферромагнетик обладает изотропными безгистерезисными нелинейными магнитными свойствами, то компоненты напряженности магнитного поля можно следующим образом выразить через соответствующие компоненты магнитной индукции:

$$H_x = \mathrm{f}(B) \cdot B_x / B; \quad H_y = \mathrm{f}(B) \cdot B_y / B; \quad H_z = \mathrm{f}(B) \cdot B_z / B, \qquad (6)$$

где $B_x = \dfrac{\partial A_z}{\partial y} - \dfrac{\partial A_y}{\partial z}$; $B_y = \dfrac{\partial A_x}{\partial z} - \dfrac{\partial A_z}{\partial x}$; $B_z = \dfrac{\partial A_y}{\partial x} - \dfrac{\partial A_x}{\partial y}$; $B = \sqrt{B_x^2 + B_y^2 + B_z^2}$;

f(B) — скалярная функция, описывающая кривую магнитных свойств материала и вычисляющая напряженность магнитного поля по заданной магнитной индукции. Если функция задана таблично, то f(B) — интерполяция

или аппроксимация табличной зависимости.

Моделирование в режиме PDE Modes

Сразу после запуска GUI-приложения femlab Навигатором моделей выберем прикладной режим 3D/ PDE Modes/ General/ Nonlinear stationary. Кнопкой More откроем дополнительные строки редактирования закладки New Навигатора моделей. В строку редактирования Dependent variables впишем имена зависимых переменных. Этими переменными должны быть декартовы компоненты векторного магнитного потенциала. Обозначим их Ax, Ay, Az. Нажатие кнопки OK приведет к переходу GUI-приложения femlab в трехмерный режим прорисовки геометрии.

Для прорисовки магнитопровода трансформатора сначала в командном окне MATLAB выполним следующую последовательность операторов:

```
Rt=rect2(-0.14,0.14,-0.117,0.117);
RO1=rect2(-0.09,-0.025,-0.067,0.067);
RO2=rect2(0.025,0.09,-0.067,0.067);
Tm=Rt-(RO1+RO2);
Tm=rotate(Tm,pi/2);
```

Эти операторы создали двумерный композиционный объект, который можно импортировать в рабочую плоскость модели FEMLAB. Командой меню Draw/ Add/Edit/Delete Work Plane создадим рабочую плоскость *z*-*x* (*y*=-0.025). Командой меню File/ Insert from Workspace/ Geometry Object(s) вставим объект Tm в созданную рабочую плоскость. Командой Draw/ Extrude выполним экструзию двумерного объекта в трехмерную область на расстояние 0,05 м. Получится трехмерный геометрический объект — магнитопровод (рис. 5). Остается только дорисовать первичные и вторичные обмотки, а также внешнюю часть расчетной области.

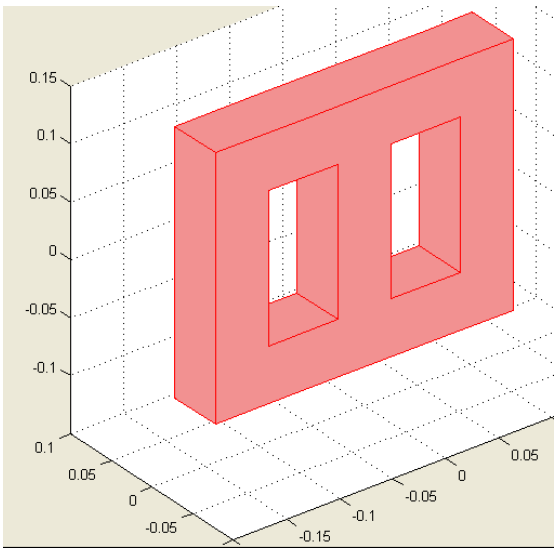


Рис. 5. Магнитопровод трансформатора.

Используя горизонтальные рабочие плоскости z=-0.01 и z=0.01, средства построения криволинейных объектов и их преобразования в двумерные солидные объекты, а также команды разделения объектов на подобласти и копирования, прорисуем первичные и вторичные обмотки трансформатора (рис. 6).

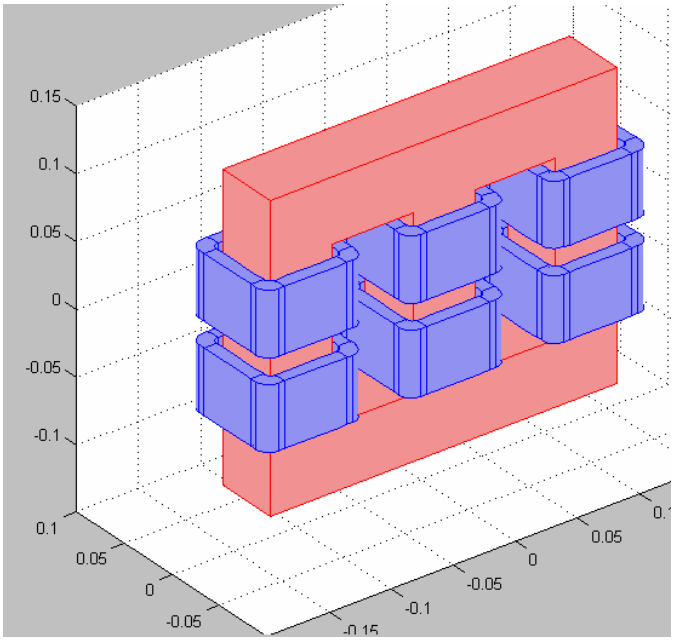


Рис. 6. Электромагнитная система трансформатора «в сборе»

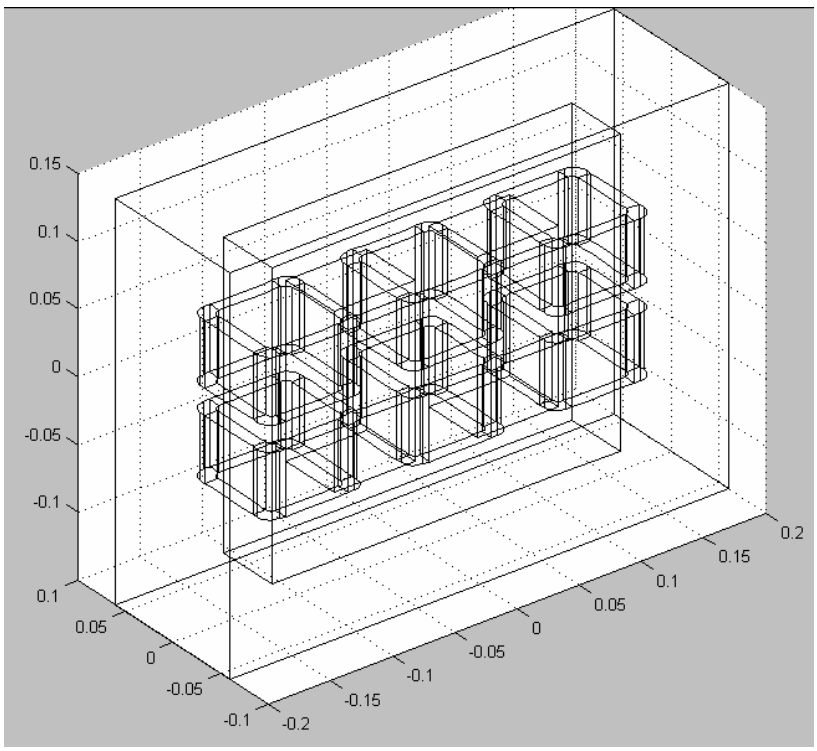


Рис. 7. Геометрия расчетной области.

Геометрическое моделирование завершается построением внешней части расчетной области. Здесь достаточно охватить электромагнитную систему трансформатора солидным прямоугольным параллелепипедом. Выполним оператор MATLAB

A1=block3(0.4,0.12,0.3,'center',[0 0 0]);

и командой меню File/ Insert from Workspace/ Geometry Object(s) вставим объект A1 в трехмерную расчетную область. Полностью готовая геометрическая модель изображена на рис. 7.

Поскольку внешняя граница расчетной области находится на некотором расстоянии от обмоток и магнитопровода, правомерно задать нулевое граничное условие Дирихле, а оно задается по умолчанию, поэтому заходить в режим Boundary Mode не обязательно. Перейдем сразу в режим Subdomain Mode.

Чтобы уравнения материальных свойств, вписываемые в строки редактирования диалогового окна Subdomain Settings, не были слишком громоздкими, полезно создать expressions-переменные в соответствии с формулами (6). Выполним команду меню Options/ Add/Edit Expressions. Раскроется диалоговое окно Expression Variable Settings. С помощью этого диалогового окна создадим восемь переменных, причем пять из них определим на уровне геометрии, а три — на уровне зон. Для краткости представим определяющие выражения этих переменных в виде m-кода файла, сохраняющего вычислительную модель:

```
% Define expressions at geometry level
fem.expr={...
        'Bx',    'Azy-Ayz',...
        'By',    'Axz-Azx',...
        'Bz',    'Ayx-Axy',...
        'B',     'sqrt(Bx^2+By^2+Bz^2)',...
        'mu0',   '4E-7*pi'};

% Определяем expressions-переменные at subdomain level
% Определим поле направлений проводов в первичных и вторичных обмотках
% (только x- и y-компоненты, т. к. все витки в горизонтальной плоскости).
% Первичная (сверху), x-direction: dLpx.
dLpx={'0','0','-(y+0.025)/sqrt((x+0.14)^2+(y+0.025)^2)','0','0','0',... 6
      '-(y-0.025)/sqrt((x+0.14)^2+(y-0.025)^2)','0','1','0',...  10
      '0','-1','0','-(y+0.025)/sqrt((x+0.09)^2+(y+0.025)^2)','0',... 15
      '-(y-0.025)/sqrt((x+0.09)^2+(y-0.025)^2)','0','0','0','-
(y+0.025)/sqrt((x+0.025)^2+(y+0.025)^2)',... 20
      '0','0','0','-(y-0.025)/sqrt((x+0.025)^2+(y-0.025)^2)','0',... 25
      '1','0','-1','0','-(y+0.025)/sqrt((x-0.025)^2+(y+0.025)^2)',... 30
      '0','-(y-0.025)/sqrt((x-0.025)^2+(y-0.025)^2)','0','0','0',... 35
      '-(y+0.025)/sqrt((x-0.09)^2+(y+0.025)^2)','0','0','0','-(y-0.025)/sqrt((x-0.09)^2+(y-
0.025)^2)',... 40
      '0','1','0','-1','0',... 45
      '-(y+0.025)/sqrt((x-0.14)^2+(y+0.025)^2)','0','-(y-0.025)/sqrt((x-0.14)^2+(y-
0.025)^2)','0','0'}; %  50
```

```
% Первичная (сверху), y-direction: dLpy.
dLpy={'0','0','(x+0.14)/sqrt((x+0.14)^2+(y+0.025)^2)','0','-1','0',... 6
      '(x+0.14)/sqrt((x+0.14)^2+(y-0.025)^2)','0','0','0',...  10
      '0','0','0','(x+0.09)/sqrt((x+0.09)^2+(y+0.025)^2)','0',... 15
      '(x+0.09)/sqrt((x+0.09)^2+(y-
0.025)^2)','0','1','0','(x+0.025)/sqrt((x+0.025)^2+(y+0.025)^2)',... 20
      '0','-1','0','(x+0.025)/sqrt((x+0.025)^2+(y-0.025)^2)','0',... 25
      '0','0','0','0','(x-0.025)/sqrt((x-0.025)^2+(y+0.025)^2)',... 30
      '0','(x-0.025)/sqrt((x-0.025)^2+(y-0.025)^2)','0','1','0',... 35
      '(x-0.09)/sqrt((x-0.09)^2+(y+0.025)^2)','0','-1','0','(x-0.09)/sqrt((x-0.09)^2+(y-
0.025)^2)',... 40
      '0','0','0','0','0',... 45
      '(x-0.14)/sqrt((x-0.14)^2+(y+0.025)^2)','0','(x-0.14)/sqrt((x-0.14)^2+(y-
0.025)^2)','0','1'}; %  50
% Вторичная (снизу), x-direction: dLvx.
dLvx={'0','-(y+0.025)/sqrt((x+0.14)^2+(y+0.025)^2)','0','0','0',... 5
      '-(y-0.025)/sqrt((x+0.14)^2+(y-0.025)^2)','0','1','0','0',...  10
      '-1','0','-(y+0.025)/sqrt((x+0.09)^2+(y+0.025)^2)','0','-(y-0.025)/sqrt((x+0.09)^2+(y-
0.025)^2)',... 15
      '0','0','0','-(y+0.025)/sqrt((x+0.025)^2+(y+0.025)^2)','0',... 20
      '0','0','-(y-0.025)/sqrt((x+0.025)^2+(y-0.025)^2)','0','1',... 25
      '0','-1','0','-(y+0.025)/sqrt((x-0.025)^2+(y+0.025)^2)','0',... 30
      '-(y-0.025)/sqrt((x-0.025)^2+(y-0.025)^2)','0','0','0','-(y+0.025)/sqrt((x-
0.09)^2+(y+0.025)^2)',... 35
      '0','0','0','-(y-0.025)/sqrt((x-0.09)^2+(y-0.025)^2)','0',... 40
      '1','0','-1','0','-(y+0.025)/sqrt((x-0.14)^2+(y+0.025)^2)',... 45
      '0','-(y-0.025)/sqrt((x-0.14)^2+(y-0.025)^2)','0','0','0'}; %  50
% Вторичная (снизу), y-direction: dLvy.
dLvy={'0','(x+0.14)/sqrt((x+0.14)^2+(y+0.025)^2)','0','-1','0',... 5
      '(x+0.14)/sqrt((x+0.14)^2+(y-0.025)^2)','0','0','0','0',...  10
      '0','0','(x+0.09)/sqrt((x+0.09)^2+(y+0.025)^2)','0','(x+0.09)/sqrt((x+0.09)^2+(y-
0.025)^2)',... 15
      '0','1','0','(x+0.025)/sqrt((x+0.025)^2+(y+0.025)^2)','0',... 20
      '-1','0','(x+0.025)/sqrt((x+0.025)^2+(y-0.025)^2)','0','0',... 25
      '0','0','0','(x-0.025)/sqrt((x-0.025)^2+(y+0.025)^2)','0',... 30
      '(x-0.025)/sqrt((x-0.025)^2+(y-0.025)^2)','0','1','0','(x-0.09)/sqrt((x-
0.09)^2+(y+0.025)^2)',... 35
      '0','-1','0','(x-0.09)/sqrt((x-0.09)^2+(y-0.025)^2)','0',... 40
      '0','0','0','0','(x-0.14)/sqrt((x-0.14)^2+(y+0.025)^2)',... 45
      '0','(x-0.14)/sqrt((x-0.14)^2+(y-0.025)^2)','0','1','0'}; %  50
% После этого зададим модули плотности тока в зонах,
% учитывая, что толщина слоя обмоток 0,01 м, высота 0,05 м.
denc={'0','0','evalin(''base'',''I(1)*500'')/5E-4','0','evalin(''base'',''I(1)*500'')/5E-4', ... 5
      '0','evalin(''base'',''I(1)*500'')/5E-4','0','evalin(''base'',''I(1)*500'')/5E-4','0', ... 10
      '0','evalin(''base'',''I(1)*500'')/5E-4','0','evalin(''base'',''I(1)*500'')/5E-4','0', ... 15
      'evalin(''base'',''I(1)*500'')/5E-4','0','evalin(''base'',''I(1)*500'')/5E-
4','0','evalin(''base'',''I(2)*500'')/5E-4',... 20
      '0','evalin(''base'',''I(2)*500'')/5E-4','0','evalin(''base'',''I(2)*500'')/5E-4','0', ... 25
      'evalin(''base'',''I(2)*500'')/5E-4','0','evalin(''base'',''I(2)*500'')/5E-
4','0','evalin(''base'',''I(2)*500'')/5E-4',... 30
```

```
    '0','evalin(''base'',''I(2)*500'')/5E-4','0','evalin(''base'',''I(2)*500'')/5E-4','0', ... 35
    'evalin(''base'',''I(3)*500'')/5E-4','0','evalin(''base'',''I(3)*500'')/5E-
4','0','evalin(''base'',''I(3)*500'')/5E-4',... 40
    '0','evalin(''base'',''I(3)*500'')/5E-4','0','evalin(''base'',''I(3)*500'')/5E-4','0', ... 45
    'evalin(''base'',''I(3)*500'')/5E-4','0','evalin(''base'',''I(3)*500'')/5E-
4','0','evalin(''base'',''I(3)*500'')/5E-4'}; %  50

fem.equ.expr={
  'dLpx',dLpx,...
  'dLpy',dLpy,...
  'dLvx',dLvx,...
  'dLvy',dLvy,...
  'denc',denc};
```

В показанном фрагменте m-файла предполагается, что в момент создания expressions-переменных в рабочей области MATLAB находится переменная с именем I (токи первичных обмоток).

Определяющие выражения для коэффициентов PDE сведем в табл. 1.

Таблица 1.
Коэффициенты PDE для векторной задачи магнитостатики в генеральной форме.

Gamma для немагнитных материалов	Gamma для стали	F для всех зон
0 Bz/mu0 -By/mu0	0 namag(B)*Bz/B -namag(B)*By/B	denc*dLpx
-Bz/mu0 0 Bx/mu0	-namag(B)*Bz/B 0 namag(B)*Bx/B	denc*dLpy
By/mu0 -Bx/mu0 0	namag(B)*By/B -namag(B)*Bx/B 0	0

Как видно из табл. 1, для такой компактной записи коэффициентов PDE нужно написать внешний m-файл namag, вычисляющий модуль напряженности магнитного поля по заданному значению модуля магнитной индукции. Для нормальной работы нелинейного решателя системы FEMLAB нужна еще одна функция, вычисляющая производную функции namag. Текст этих m-функций приведен ниже.

```
function h=namag(b);
h=interp1(evalin('base','B'),evalin('base','H'),b,'cubic');
function h=namagder(b);
if abs(b)<=0.025, b=0.025; end
h=(namag(b*1.01)-namag(b))/b*100;
```

Эти m-функции нужно прописать в диалоговом окне Differentiation Rules, раскрываемом командой меню группы Options.

Теперь можно строить сетку. Поскольку зависимых переменных в модели три, расчет может выполняться только на самой грубой сетке и только для линейной задачи. При включении нелинейного решателя памяти компьютера (768 Мб) уже не хватает при любой настройке управления памятью в операционной системе. Параметры самой грубой сетки, состоя-

щей из Лагранжевых элементов первого порядка: число узлов 12425, число элементов 69573, число степеней свободы 37275. Отсюда вывод один: при практических расчетах трехмерных магнитостатических полей в устройствах с нелинейными магнитными материалами уравнения математической физики нужно составлять относительно скалярного магнитного потенциала. А такого прикладного режима в системе FEMLAB нет. Моделировать можно только в режиме PDE Modes.

Нелинейная скалярная краевая задача магнитостатики в генеральной форме

Понятие скалярного магнитного потенциала вводится на основе закона полного тока (1). Из этого закона следует, что полный ток не имеет стоков и истоков, т. е. дивергенция плотности тока равна нулю, и саму плотность тока можно представить в виде ротора некоторого векторного поля, называемого фиктивной намагниченностью:

$$\delta = \operatorname{rot} \mathbf{M}_{\text{ф}},$$

т. е. в соответствии с (1) получим

$$\operatorname{rot} \boldsymbol{H} = \operatorname{rot} \boldsymbol{M}_{\text{ф}} \quad \Rightarrow \quad \boldsymbol{H} = \boldsymbol{M}_{\text{ф}} - \operatorname{grad} \varphi_H,$$

где φ_H — скалярный магнитный потенциал, измеряемый в единицах тока.

Теперь на основе закона непрерывности линий магнитной индукции можно записать нелинейное скалярное уравнение магнитостатики в генеральной форме:

$$\operatorname{div} \boldsymbol{B} = \operatorname{div} \mathrm{f}(\boldsymbol{H}) = \operatorname{div}\left(\mathrm{f}(\boldsymbol{M}_{\text{ф}} - \operatorname{grad} \varphi_H)\right) = 0, \qquad (7)$$

где f — векторная функция, связывающая магнитную индукцию с напряженностью магнитного поля. В случае линейных изотропных магнитных свойств компоненты векторов $\boldsymbol{B}$ и $\boldsymbol{H}$ могут быть связаны между собой соотношениями вида

$$B_x = \mathrm{g}\left(\sqrt{\left(M_{\text{ф}x} - \frac{\partial \varphi_H}{\partial x}\right)^2 + \left(M_{\text{ф}y} - \frac{\partial \varphi_H}{\partial y}\right)^2 + \left(M_{\text{ф}z} - \frac{\partial \varphi_H}{\partial z}\right)^2}\right) \cdot$$

$$\cdot \left(M_{\text{ф}x} - \frac{\partial \varphi_H}{\partial x}\right) \Bigg/ \sqrt{\left(M_{\text{ф}x} - \frac{\partial \varphi_H}{\partial x}\right)^2 + \left(M_{\text{ф}y} - \frac{\partial \varphi_H}{\partial y}\right)^2 + \left(M_{\text{ф}z} - \frac{\partial \varphi_H}{\partial z}\right)^2}$$

и т. д. Здесь g — скалярная функция, связывающая модули магнитной индукции и напряженности магнитного поля. Остальные компоненты вектора $\boldsymbol{B}$ вычисляются по аналогичным формулам.

Ниже приведен фрагмент m-файла, сохраняющего FEMLAB-модель, в котором показано распределение выражений для коэффициентов PDE по зонам расчетной области. Здесь для простоты функция g, представленная в

табл. 1, аппроксимирована арктангенсом. Показаны также константы FEMLAB-модели.

```
appl{1}.equ.ga={{{'-ux*mu0';'-uy*mu0';'-uz*mu0'}},{{'-ux*mu0';'-uy*mu0'; ...
'(mfmax1-uz)*mu0'}}, ...
{{'-nam*atan(na*sqrt(ux^2+uy^2+uz^2))*ux/sqrt(ux^2+uy^2+uz^2)'; ...
'-nam*atan(na*sqrt(ux^2+uy^2+uz^2))*uy/sqrt(ux^2+uy^2+uz^2)'; ...
'-nam*atan(na*sqrt(ux^2+uy^2+uz^2))*uz/sqrt(ux^2+uy^2+uz^2)'}}, ...
{{'-nam*atan(na*sqrt(ux^2+uy^2+(mfmax1-uz)^2))*ux/sqrt(ux^2+uy^2+(mfmax1-
uz)^2)'; ...
'-nam*atan(na*sqrt(ux^2+uy^2+(mfmax1-uz)^2))*uy/sqrt(ux^2+uy^2+(mfmax1-
uz)^2)'; ...
'nam*atan(na*sqrt(ux^2+uy^2+(mfmax1-uz)^2))*(mfmax1-
uz)/sqrt(ux^2+uy^2+(mfmax1-uz)^2)'}}, ...
{{'-ux*mu0';'-uy*mu0';'(mfmax2-uz)*mu0'}}, ...
{{'-nam*atan(na*sqrt(ux^2+uy^2+(mfmax2-uz)^2))*ux/sqrt(ux^2+uy^2+(mfmax2-
uz)^2)'; ...
'-nam*atan(na*sqrt(ux^2+uy^2+(mfmax2-uz)^2))*uy/sqrt(ux^2+uy^2+(mfmax2-
uz)^2)'; ...
'nam*atan(na*sqrt(ux^2+uy^2+(mfmax2-uz)^2))*(mfmax2-
uz)/sqrt(ux^2+uy^2+(mfmax2-uz)^2)'}}, ...
{{'-ux*mu0';'-uy*mu0';'(mfmax3-uz)*mu0'}}, ...
{{'-nam*atan(na*sqrt(ux^2+uy^2+(mfmax3-uz)^2))*ux/sqrt(ux^2+uy^2+(mfmax3-
uz)^2)'; ...
'-nam*atan(na*sqrt(ux^2+uy^2+(mfmax3-uz)^2))*uy/sqrt(ux^2+uy^2+(mfmax3-
uz)^2)'; ...
'nam*atan(sqrt(ux^2+uy^2+(mfmax3-uz)^2))*(mfmax3-uz)/sqrt(ux^2+uy^2+(mfmax3-
uz)^2)'}}};
appl{1}.equ.ind=[1 1 1 1 2 3 3 3 4 3 1 1 1 5 3 3 6 1 1 1 7 3 3 8];
% Differentiation rules
fem.rules={'namagh','namaghder'};
% Define constants
fem.const={...
'mu0',    1.2566370614359173e-006,...
'mfmax1', 1860.8073189119671,...
'mfmax2', 681.10275026979207,...
'mfmax3', -2541.9100691817584,...
'nam',    0.85943669269623491,...
'na',     0.011938052083641213};
```

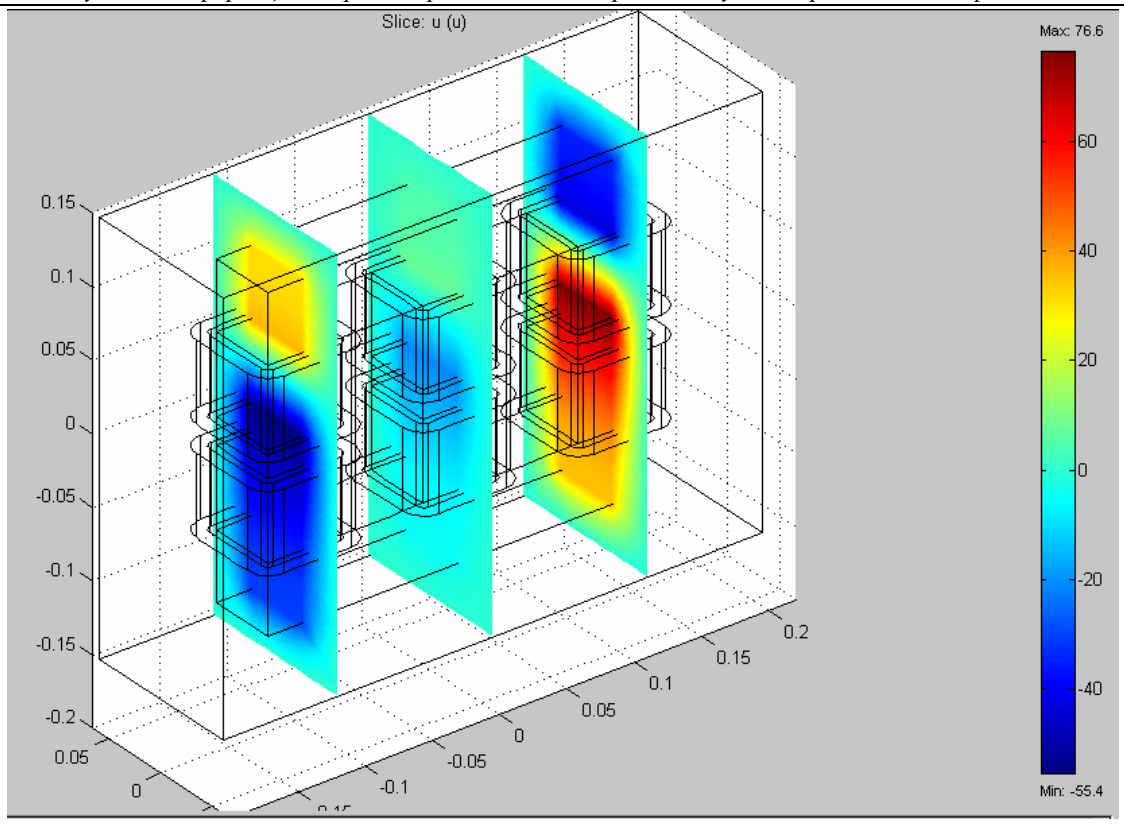


Рис. 8. Скалярный магнитный потенциал в сечениях расчетной области.

На рис. 8 показано распределение скалярного магнитного потенциала в трех сечениях магнитопровода трансформатора. От этого распределения по необходимости можно перейти к распределению напряженности поля и магнитной индукции. Можно вычислять также интегральные параметры поля в нужных областях. Такими параметрами могут быть магнитные напряжения, магнитные потокосцепления в обмотках и магнитные потоки, энергия магнитного поля, индуктивности и взаимные индуктивности обмоток.

Изложенная технология моделирования может применяться для решения практических задач электротехники и электроэнергетики.

Литература

1. *Бессонов Л. А.* Теоретические основы электротехники: Электромагнитное поле. Учебник для студентов вузов.— М.: Высш. школа, 2002.— 231 с.