САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П.КОРОЛЕВА

МАТРИЧНЫЕ ВЫЧИСЛЕНИЯ НА ПЕРСОНАЛЬНОЙ ЭВМ

САМАРА 1994

Государственный комитет Российской Федерации
по высшему образованию

Самарский государственный аэрокосмический
университет имени академика С.П.Королева

МАТРИЧНЫЕ ВЫЧИСЛЕНИЯ НА ПЕРСОНАЛЬНОЙ ЭВМ

Методические указания

Самара 1994

Составитель С.В. Суханов

УДК 681.3

Матричные вычисления на персональной ЭВМ :
Метод.указания/ Самар.гос.аэрокосм.ун-т;Сост.С.В.Суханов.
Самара,1994. 28с.

Указания содержат порядок использования пакета программ матричных вычислений MATLAB для персональных ЭВМ, совместимых с IBM PC. Описаны функции пакета, реализующие операции матричной алгебры,на примерах рассмотрены приемы работы с индексами, массивами, векторами и матрицами, изложены возможности экспорта/импорта данных, связи с другими программными средствами, вывода графической информации на печатающее устройство.

Указания предназначены для студентов факультета "Информатика" специальности 01.02 "Прикладная математика", выполняющих лабораторные работы и курсовые проекты по курсам: "Теория сигналов", "Цифровая обработка сигналов", "Методы прикладной математики", "Планирование эксперимента и статистический анализ" и ряда других. Указания будут также полезны для широкого круга пользователей персональных ЭВМ. Подготовлены на кафедре технической кибернетики.

Печатаются по решению редакционно-издательского совета
Самарского государственного аэрокосмического университета
имени академика С.П.Королева

Рецензент К.В. Овчинников

ВВЕДЕНИЕ

В настоящих методических указаниях, являющихся продолжением работы [1], в которой даны элементарные сведения, необходимые начинающему пользователю, даются более подробные сведения о пакете MATLAB, рассматривается на примерах специфика матричных операций и функций, описываются способы связи с внешними программами и вопросы вывода текстовой и графической информации на печать.Предполагается, что читатель знаком с пакетом MATLAB в объеме работы [1].

При написании настоящих методических указаний был использован рукописный текст [2], распространяемый в виде компьютерного файла на некоммерческой основе.

Приводимые в тексте примеры представляют собой вывод на экран монитора информации, вводимой пользователем, и выводимой компьютером, поэтому знаки пунктуации, обеспечивающие синтаксическую правильность текста, даны с отступом.

1. ОПЕРАЦИИ НАД МАТРИЦАМИ

Матричные операции являются основными в MATLAB; возможны те же операции, что и в книге, и на бумаге, ограничение только в возможностях работы компьютера с символами. Полный список функций MATLAB приведен в Приложении.

1.1. Транспонирование

Штрих (апостроф) ' означает транспонирование матрицы с действительными элементами. Примеры:

операции `A = [1 2 3; 4 5 6; 7 8 0], B = A'` *дают*

```
A =                B =
     1  2  3            1  4  7
     4  5  6            2  5  8
     7  8  0            3  6  0
```

`x = [-1 0 2]'` *представляет собой*

```
x =
    -1
     0
     2
```

Штрих ' транспонирует матрицу в формальном смысле. Если Z комплексная матрица, то Z' будет комплексно-сопряженной транспонированной. Это может привести к неожиданным результатам, если обращаться с комплексными данными небрежно. Чтобы получить несопряженную транспонированную матрицу, следует использовать Z.' или `conj(Z')`.

1.2. Сложение и вычитание

Сложение и вычитание матриц обозначаются через + и -. Эти операции определены тогда, когда матрицы имеют одинаковые размеры. Например, с вышеупомянутыми матрицами выражение A + x некорректно, так как A - квадратная матрица порядка 3, а x имеет размеры 3x1. Однако выражение C = A + B вполне допустимо и дает

```
C =
     2   6  10
     6  10  14
    10  14   0
```

Сложение и вычитание также определены для случая, когда один из операндов является скаляром, т.е. матрицей размера 1x1. В этом случае скалярная величина прибавляется к каждому элементу матрицы или вычитается из каждого элемента. Например, y = x - 1 дает

```
y =
    -2
    -1
     1
```

1.3. Умножение матриц

Умножение матриц обозначается звездочкой *. Умножение допускается, когда число столбцов первого операнда равно числу строк второго. Например, вышеуказанные векторы x и y оба размером 3x1, поэтому выражение x*y некорректно, и выдается сообщение об ошибке. Однако для векторов определены и широко используются другие типы произведения. Наиболее общеупотребимым является внутреннее произведение, которое называется также точечным или скалярным.

В этом случае x'*y дает в результате ans = 4. Разумеется, y'*x даст тот же результат. Для этих же векторов есть два внешних (матричных) произведения, которые являются транспонированием друг друга, а именно:

```
x*y'                y*x'
ans =               ans =
     2   1  -1           2   0  -4
     0   0   0           1   0  -2
    -4  -2   2          -1   0   2
```

Поэлементное произведение описано в следующем разделе. В MATLAB нет обозначений для других векторных произведений, но можно легко самим написать программу для их расчета.

Произведение матрицы на вектор является частным случаем произведения матрицы на матрицу. Например, для наших A и x b = A*x вполне допустимо и дает в результате

```
b =
     5
     8
    -7
```

Естественным образом определяется умножение скалярной величины на любую матрицу и наоборот:

```
pi*x
ans =
   -3.1416
    0.0000
    6.2832
```

1.4. Деление матриц

В MATLAB есть два знака, обозначающих деление матриц: \ и /; если A является невырожденной квадратной матрицей, то A\B и B/A формально являются соответственно результатом левого или правого умножения B на обратную к A матрицу, т.е. inv(A)*B и B*inv(A), но на самом деле результат получается без вычисления обратных матриц. Обычно:

X = A\B - *решение уравнения* A*X = B;

X = B/A - *решение уравнения* X*A = B;

Левое деление A/B определено, когда B имеет ровно столько же строк, что и A. Если A - квадратная матрица, то она разлагается на множители с помощью метода Гаусса. Эти множители используются для решения уравнений A*X(:,j) = B(:,j), где B(:,j) - j-й столбец матрицы B. В результате получается матрица X с теми же размерами, что и B. Если это почти вырожденная матрица (согласно условной оценке LINPACK, RCOND), то печатается предупреждающее сообщение.

Если A неквадратна, то она разлагается на множители с помощью ортогонализации Хаусхолдера с поворотом столбцов. Эти множители используются для определенных и переопределенных уравнений в смысле наименьших квадратов. В результате получается матрица X размером mxn, где m - число столбцов матрицы A, а n - число строк матрицы B. Каждый столбец X имеет k ненулевых элементов, где k - эффективный ранг A.

Правое деление легче всего определить через левое деление:

```
B/A = (A'\B')'
```

Например, т.к. вектор b был рассчитан как A*x, то

```
z = A\b     даст
z =
    -1
     0
     2
```

Иногда использование / и \ для вычислений решений наименьших квадратов недо- и переопределенных систем уравнений может вызвать удивление. Так, можно "разделить" один вектор на другой, (x и y были определены ранее): `s = x\y` дает `s = 0.8000`, т.е. s является скалярной величиной - решением переопределенного уравнения `xs = y` в смысле наименьших квадратов. Предлагаем читателю самому объяснить, почему `S = y/x` дает

```
S =
    0.0000    0.0000   -1.0000
    0.0000    0.0000   -0.5000
    0.0000    0.0000    0.5000
```

1.5. Возведение матриц в степень

Выражение `A^p` означает "A в степени p" и определено, если A - квадратная матрица, а p - скаляр. Если p - целое число, большее 1, то степень вычисляется повторным умножением. Для остальных значений p расчет включает собственные значения и собственные векторы: если

`[V,D] = eig(A),` *то* `A^p = V*D.^p/V` .

Если P - матрица, а - скаляр, то `a^P` есть a, возведенное в степень P, с применением собственных значений и собственных векторов. Однако запись `X^P`, где X и P - матрицы, будет ошибочной.

1.6. Трансцендентные функции матриц

В MATLAB выражения типа `exp(A)` и `sqrt(A)` рассматриваются как операции над отдельными элементами массива, определенного матрицей A. В MATLAB, кроме того, есть возможность вычислять и трансцендентные функции матриц, такие, как экспонента и логарифм. Эти функции определены для квадратных матриц; они довольно трудоемки для вычислений, но иногда позволяют выявить тонкие математические особенности. Трансцендентная математическая функция будет интерпретироваться как матричная в том случае, если в конце имени функции

добавляется "m", например, expm(A) или sqrtm(A). В MATLAB определены всего три подобные функции:

expm - матричная экспонента;

logm - матричный логарифм;

sqrtm - матричный квадратный корень.

Однако перечень при необходимости легко можно расширить введением дополнительных M-файлов или использованием funm. Для получения дополнительной информации воспользуйтесь help.

1.7. Функции, полезные для построения матриц

Несколько функций построения матриц создают матрицы, удобные для использования. Эти функции включают eye(A), которая выдает единичную матрицу того же размера, что A. Надо использовать броское имя, т.к. I и i используются как индексы или sqrt(-1). Эта группа включает также zeros и ones, которые образуют постоянные матрицы разных размеров, и rand, которая генерирует матрицу равномерно или нормально распределенных случайных элементов. Пример создания матрицы случайных величин:

```
A = rand(4,3)
A =
    0.2113    0.8096    0.4832
    0.0824    0.8474    0.6135
    0.7599    0.4524    0.2749
    0.0087    0.8075    0.8807
```

Три функции diag, triu и tril обеспечивают доступ к диагональной, верхней треугольной и нижней треугольной частям матрицы. Например,

```
tril(A)
ans =
    0.2113    0         0
    0.0824    0.8474    0
    0.7599    0.4524    0.2749
    0.0087    0.8075    0.8807
```

Кроме того, очень полезны size и length. Функция size выдает двухэлементный вектор, содержащий число строк и столбцов матрицы. Если известно, что переменная является вектором, то length выдает длину вектора; тот же результат дает max(size(V)).

Большие матрицы можно получить из меньших путем заключения в квадратные скобки меньших матриц. Например,

```
C = [A eye(4); ones(A) A^2]
```

создает одну большую матрицу C, предполагая, что A имеет четыре строки. Меньшие матрицы в этом типе конструкций должны иметь согласованные размеры, иначе будет сообщение об ошибке.

1.8. Особенности вычислений с неопределенными значениями

Специальное значение NaN означает Not-a-Number (не число) в MATLAB. Обычно оно создается неопределенным выражением типа 0/0 вместо сообщения об ошибке, благодаря условным обозначениям, установленным стандартами для арифметического устройства с плавающей точкой IEEE. В статистике величины типа NaN могут использоваться для представления отсутствующих значений.

Правильное обращение с отсутствующими значениями вызывает затруднение и часто меняется в разных ситуациях. Однако, MATLAB в своей трактовке NaN является постоянным и точным; они распространяются естественно до конечного результата в любых расчетах. Таким образом, если NaN используется в любом промежуточном расчете, то окончательный результат будет NaN.

Практически это означает, что все NaN должны удаляться из данных до выполнения статистических рассчетов. Все NaN из вектора x можно удалить несколькими способами, используя специальную функцию isnan:

```
x = x(find(~isnan(x)));   или
x = x(~isnan(x));   или
x(isnan(x)) = [];
```

Если из матрицы данных надо удалить все строки, содержащие NaN, то можно использовать такой оператор:

```
X(any(isnan(X)'),:) = [];
```

2. ОПЕРАЦИИ НАД МАССИВАМИ

Мы уже использовали термин "операции над массивами" для названия поэлементных математических операций. В отличие от обычных алгебраических матричных операций поэлементные операции над массивами обозначаются точкой "." перед символом операции.

2.1. Сложение и вычитание массивов

Матричное и поэлементное сложение и вычитание ничем не отличаются друг от друга, поэтому "+" и "-" могут рассматриваться и как матричные операции, и как операции над массивами.

2.2. Умножение и деление массивов

Умножение массивов (поэлементное) обозначается ".*". Если A и B имеют одинаковые размеры, то A.*B - массив, элементы которого являются произведениями элементов A и B. Например, если

```
x = [1 2 3]; y = [4 5 6];      ,то
z = x.*y   даем
z =
    4   10   18
```

Выражения A./B и A.\B дают отношения отдельных элементов. Поэтому

```
z = x.\y   даем
z =
    4.0000    2.5000    2.0000
```

2.3. Степени массивов

Поэлементное возведение в степень обозначается ".^". Далее приведено несколько примеров, использующих векторы x и y (показатель и основание могут быть константами):

```
z=x.^y              z=x.^2          z=2 .^[x y]
z=                  z=              z=
   1   32  729         1   4   9       2   4   8   16   32   64 .
```

Внимание! Последний пример иллюстрирует одну из синтаксических тонкостей MATLAB, а именно: пробел между цифрой 2 и точкой, относящейся к знаку операции (возведения в степень) над массивом. Если бы этого пробела не было, то эта точка интерпретировалась бы как десятичная точка, связанная с 2; операция возведения в степень, соответственно, рассматривалась бы как матричная, что привело бы к ошибке, т.к. матрица [x y] не является квадратной. Альтернативой пробелу являются скобки, устанавливающие правильную последовательность операций.

2.4. Операции отношения

Существует шесть (реляционных) операций отношения, которые могут использоваться при сравнении двух матриц одного размера:

< - *меньше;*
<= - *меньше или равно;*
> - *больше;*
>= - *больше или равно;*
== - *равно;*
~= - *не равно.*

Сравнение осуществляется между парами соответствующих элементов, результатом является матрица из нулей и единиц, где 1 соответствует значению TRUE (истина), а 0 - значению FALSE (ложь). Например, 2+2 ~= 4 - это просто Ø.

Операторы отношения могут давать наглядную картину элементов матрицы, которые удовлетворяют различным условиям. Вот, например, магический квадрат шестого порядка

```
A = magic(6)
A =
    35    1    6   26   19   24
     3   32    7   21   23   25
    31    9    2   22   27   20
     8   28   33   17   10   15
    30    5   34   12   14   16
     4   36   29   13   18   11
```

Магический квадрат порядка n представляет собой квадратную матрицу порядка n, состоящую из целых чисел от 1 до n.^2, суммы элементов которой по рядам и столбцам равны между собой. Если посмотреть на наш магический квадрат внимательно, можно заметить, что элементы, кратные 3, находятся на каждой третьей диагонали. Для отражения этой любопытной особенности выполним

```
P = (rem(A,3)==Ø)
```

Здесь знак "==" - оператор проверки на равенство, rem(A,3) - матрица остатков, каждый из элементов которой сравнивается с Ø, поэтому P будет матрицей из Ø и 1:

```
P =
    Ø   Ø   1   Ø   Ø   1
    1   Ø   Ø   1   Ø   Ø
    Ø   1   Ø   Ø   1   Ø
    Ø   Ø   1   Ø   Ø   1
    1   Ø   Ø   1   Ø   Ø
    Ø   1   Ø   Ø   1   Ø
```

Чтобы сделать картину еще более наглядной, используем команду format+ , которая печатает "+" вместо положительных элементов, "-" вместо отрицательных и пробел вместо нулей:

```
format+
P
      +  +
    +  +
     +  +
      +  +
    +  +
     +  +
```

Функция find полезна с реляционными операторами, находящими ненулевые элементы в Ø–1 матрицах, а значит, и элементы данных, которые удовлетворяют какому-либо условию. Например, если Y - вектор, то find(Y < 3.Ø) выдает вектор, содержащий индексы всех тех элементов Y, которые меньше трех.

Отношение X==NaN выдает NaN везде, т.к. в соответствии с арифметическими спецификациями IEEE любая операция с NaN дает в результате NaN. Но иногда возникает необходимость проверить NaN. Для этого используется функция isnan(X), которая выдает 1 в тех местах, где элементы X равны NaN, и Ø во всех остальных. Полезной является обратная функция finite(x), которая возвращает 1, если элемент X не равен NaN.

2.5. Логические операции

Существуют три логические операции, которые работают поэлементно и в основном используются на Ø–1 матрицах:

& - И; | - ИЛИ; ~ - НЕ.

Операторы & и | сравнивают два скаляра или две матрицы одного размера. Для матриц они работают поэлементно; если A и B - две матрицы, состоящие из нулей и единиц, то A&B является другой Ø–1 матрицей, представляющей логическое И соответствующих элементов A и B. Логические операторы рассматривают любые ненулевые значения как истинные и в качестве элемента с индексами i,j возвращают 1, если значение A(i,j)&B(i,j) – ИСТИНА и ноль, если значение - ЛОЖЬ.

Логическое НЕ (~) является унарным оператором. Выражение ~A возвращает Ø, где A не нуль, и 1, где A нуль. Таким образом, выражения P|(~P) и P&(~P) возвращают соответственно все 1 или все Ø.

Функции any и all эффективны в сочетании с логическими операторами. Для векторов, элементами которых являются Ø и 1, any(x) дает 1, если в x есть хотя бы один ненулевой элемент, и Ø, если x состоит из одних нулей. Функция all(x) возвращает 1, если все элементы x ненулевые, и возвращает Ø, если есть хотя бы один

нулевой элемент. Особенно полезны эти функции бывают в условном операторе, т.к. if проверяет простое условие, а не вектор всевозможных условий:

```
if all(A<5)
    тело цикла
end
```

С матрицами any и all работают по столбцам и возвращают вектор-строку с результатом для каждого столбца. Применение этих функций дважды, вроде any(any(A)), всегда дает скаляр.

2.6. Элементарные математические функции

Элементарные математические функции работают с матрицами поэлементно. Например,

```
A = [1 2 3;4 5 6]; B = fix(pi*A);   C = cos(pi*B);
A =                 B =             C =
     1   2   3           3   6   9       -1   1  -1
     4   5   6          12  15  18        1  -1   1
```

Список элементарных математических функций:

abs	- абсолютное значение или модуль комплексного числа;
sqrt	- квадратный корень;
real	- действительная часть комплексного числа;
imag	- мнимая часть;
conj	- комплексное сопряжение;
round	- округление до ближайшего целого;
fix	- округление в направлении нуля;
floor	- округление в отрицательном направлении;
ceil	- округление в положительном направлении;
sign	- функция знака (signum);
rem	- остаток;
sin	- синус;
cos	- косинус ;
tan	- тангенс ;
asin	- арксинус ;
acos	- арккосинус;
atan	- арктангенс;
atan2	- арктангенс отношения (y,x), определен на [-pi,pi];
sinh	- синус гиперболический;

cosh - косинус гиперболический;
tanh - тангенс гиперболический;
exp - экспонента по основанию e;
log - натуральный логарифм;
log10 - логарифм по основанию 10;
bessel - функция Бесселя;
gamma - гамма-функция;
rat - действительная аппроксимация.

3. ВЕКТОРЫ И ИНДЕКСЫ

Способность MATLAB к индексированию позволяет легко обращаться к строкам, столбцам, отдельным элементам матриц и подматрицам. Векторы и индексы используются довольно часто, они дают возможность эффективной обработки разнообразных данных.

3.1. Генерация векторов

Двоеточие является очень важным знаком в MATLAB. Сгенерируем вектор-строку чисел от 1 до 5 с приращением 1:

```
x = 1:5
x =
    1   2   3   4   5
```

Могут быть и другие приращения:

```
y = 0:pi/4:pi
y =
    0.0000   0.7854   1.5708   2.3562   3.1416
```

Возможно отрицательное приращение:

```
z = 6:-1:1
z =
    6   5   4   3   2   1
```

Использование двоеточия позволяет легко создавать таблицы: для получения столбца значений аргумента транспонируется вектор - строка, полученная с помощью двоеточия; затем рассчитывается столбец значений функции и формируется матрица из этих двух столбцов, например:

```
x = (0.0:0.2:1.0)';
y = exp(-x).*sin(x);
```

```
[x y]
ans =
        0.0000    0.0000
        0.2000    0.1627
        0.4000    0.2610
        0.6000    0.3099
        0.8000    0.3223
        1.0000    0.3096
```

3.2. Индексирование

Отдельные элементы матрицы можно выделить, заключив в скобки их индексы. Выражение, используемое как индекс, округляется до ближайшего целого числа. Например, если дана матрица:

```
A =
     1   2   3
     4   5   6
     7   8   9
```

то оператор `A(3,3) = A(1,3) + A(3,1)` *дает результат*

```
A =
     1   2   3
     4   5   6
     7   8  10
```

Индекс может быть вектором. Если X и V - векторы, то `X(V)` - это `[X(V(1)),X(V(2)),...,X(V(n))]`. Для матриц индексы-векторы дают доступ к смежным и несмежным подматрицам. Пусть A - матрица порядка 10. Тогда `A(1:5,3)` определяет подматрицу размером 5*1 или вектор-столбец, который состоит из первых пяти элементов третьего столбца матрицы A. Аналогично, `A(1:5,7:10)` определяет подматрицу из элементов первых пяти строк последних четырех столбцов матрицы A.

Использование двоеточия вместо индекса обозначает соответствующие строку или столбец целиком: `A(:,3)` обозначает третий столбец, а `A(1:5,:)` обозначает первые пять строк. Довольно сложный эффект возникает при использовании двоеточия с обеих сторон оператора присваивания:

```
A(:,[3 5 10]) = B(:,1:3)
```

заменяет третий, пятый и десятый столбцы матрицы A на три первых столбца матрицы B. В общем случае, если v и w - векторы с целыми элементами, то `A(v,w)` - матрица, составленная из элементов A с индексом строки из v и индексом столбца из w.

3.3. Индексирование 0–1 векторами

0–1 векторы, обычно создаваемые в результате операций отношения, могут использоваться для описания подматриц. Пусть A - матрица размера m*n, L - вектор длины m, составленный из 0 и 1. Тогда A(L,:) определяет те строки A, где элементы L - не нули.

Ниже показано, как с помощью вектора могут быть удалены те элементы, которые больше трех стандартных отклонений:

```
x = x(x<=3*std(x));
```

В другом примере матрица X заменяется теми своими строками, третий элемент в которых больше 100:

```
L = X(:,3) > 100;  X = X(L,:);
```

3.4. Пустые матрицы

Оператор `x = []` обязывает x "быть матрицей размера 0*0". Последующее использование этой матрицы не приведет к состоянию ошибки. Это отличается от оператора `clear(x)`, который удаляет x из списка текущих переменных. Пустые матрицы существуют в рабочей области и имеют нулевой размер. Функция `exist` может использоваться для проверки наличия матрицы (или файла), а функция `size` покажет, является ли матрица пустой.

Можно генерировать и пустые векторы. Если n меньше 1, то `1:n` не содержит элементов и является, таким образом, пустым вектором.

Более важным является другой способ удаления строк и столбцов матрицы, который заключается в присваивании им значения пустой матрицы. Так,

```
A(:,[2 4]) = []
```

удаляет второй и четвертый столбцы матрицы A.

При задании пустых матриц в качестве аргументов некоторые функции возвращают математически правдоподобные значения. К этим функциям относятся, например, `prod`, `det` и `sum`, которые возвращают 1,1 и 0 соответственно.

4. МАТРИЧНЫЕ ФУНКЦИИ

Матричные функции MATLAB в большой степени определяют производительность. Одни из них встроены в интерпретатор MATLAB, другие - в библиотеку внешних М-файлов. В этом разделе будет дан обзор имеющихся функций.

4.1. Разложение на треугольные множители

Основным способом разложения квадратной матрицы является представление ее в виде произведения двух матриц, одна из которых получена перестановкой из нижнетреугольной матрицы, а другая является верхнетреугольной матрицей. Это разложение часто называют LU-разложением или реже LR-разложением. Большинство алгоритмов, используемых при вычислении множителей, являются вариантами метода Гаусса.

Множители получаем как результат функции lu. Разложение это используется для получения инверсии в функции inv и для вычисления определителя в функции det. На этом разложении основано также решение линейных уравнений и деление матриц.

Например, пусть

```
A =
     1   2   3
     4   5   6
     7   8   9
```

Для получения LU-разложения выполним

```
[L,U] = lu(A)
L =                                   U =
    0.1429   1.0000   0                 7.0000   8.0000   0.0000
    0.5714   0.5000   1.0000            0        0.8571   3.0000
    1.0000   0        0                 0        0        4.5000
```

Заметьте, что L получена перестановкой из нижнетреугольной матрицы, имеющей единицы на главной диагонали, а U - верхнетреугольная матрица. Для проверки правильности разложения вычислим произведение L*U , которое должно совпасть с первоначальной A. В самом деле

```
ans =
     1   2   3
     4   5   6
     7   8   9
```

Матрицу, обратную матрице A, можно получить следующим образом:

```
X = inv(A)
```

Фактически обратная матрица рассчитывается как произведение обратных к треугольным множителям

```
X = inv(U)*inv(L)
```

Получим определитель матрицы A

```
d = det(A)
d =
    27
```

Это вычислено по определителям треугольных множителей

```
d = det(L)*det(U)
d =
    27.0000
```

Почему два значения d выдаются в таком различном формате? Когда мы запрашиваем у MATLAB расчет `det(A)` , учитывается, что все элементы A являются целыми числами, поэтому и определитель - целое число. Элементы U не являются целыми числами, поэтому во втором случае d интерпретируется как вещественное.

Чтобы решить систему линейных уравнений, надо найти решение матричного уравнения $Ax = b$, что выполняется с помощью оператора матричного деления MATLAB (вектор - столбец `b = [1 3 5]'`)

```
x = A\b
x =
    0.3333
    0.3333
    0.0000
```

Фактически решение находилось из решения двух треугольных систем

```
y = L\b,   x = U\y
```

Промежуточное решение

```
y =
    5.0000
    0.2857
    0.0000
```

Разложение на треугольные множители используется также специальной функцией `rcond` , оценивающей обратное условное число квадратной матрицы. И еще две функции могут быть включены в эту группу из-за тесной связи их алгоритмов с LU-разложением. Функция `chol` для симметричной положительно определенной матрицы находит разложение Колеского. Ступенчатая форма приведенного ряда прямоугольной матрицы `rref` представляет интерес в теоретической алгебре и включена в MATLAB в учебных целях.

4.2. Ортогональное разложение на множители

QR-разложение полезно как для квадратных, так и для треугольных матриц. Оно выражает матрицу через произведение ортогональной матрицы и верхнетреугольной матрицы. Например, возьмем матрицу с неполным рангом: средний столбец представляет собой среднее двух других столбцов

```
A =
     1    2    3
     4    5    6
     7    8    9
    10   11   12
```

Неполнота ранга обнаруживается при разложении

```
[Q,R] = qr(A)
Q =
    -0.0776    -0.8331     0.5444     0.0605
    -0.3105    -0.4512    -0.7709     0.3251
    -0.5433    -0.0694    -0.0913    -0.8317
    -0.7762     0.3124     0.3178     0.4461
R =
   -12.8841   -14.5916   -16.2992
          0    -1.0413    -2.0826
          0          0     0.0000
          0          0          0
```

Мы, конечно, могли бы проверить, что произведение Q*R даст нам ту же матрицу A, но не станем зря тратить время. Нули ниже диагонали в R обусловлены ее треугольной структурой; нуль же на диагонали в R(3,3) означает, что R, а значит и A, не имеют полного ранга.

QR-разложение используют при решении линейных систем с числом уравнений большим, чем число переменных. Например, линейная система Ax = b представляет собой 4 уравнения с тремя неизвестными. Вектор-столбец b = [1 3 5 7]' . Наилучшее решение системы в смысле наименьших квадратов рассчитывается как

```
x = A\b
Warning:  Rank deficient,  rank=2  tol=1.4594E-014
x =
    0.5000
    0.0000
    0.1667
```

Получено предупреждение о неполноте ранга. Величина `tol` - это допуск, используемый при оценке возможности пренебречь диагональным элементом R. Решение x рассчитано с использованием разложения в два шага

```
y = Q'*b;
x = R\y;
```

Если бы нам надо было проверить вычисленные решения путем формирования `A*x` , мы бы увидели, что оно равно `b` в пределах погрешности округления. Это говорит о том, что даже если система уравнений `Ax = b` является переопределенной и с неполным рангом, то эти уравнения совместимы. Есть бесконечно много решений вектора x, QR-разложением нашли одно из них.

Это разложение является также основанием для функций `null` и `orth`, которые генерируют ортонормальной базис для нулевого пространства и область заданной прямоугольной матрицы.

4.3. Декомпозиция сингулярного значения

Мы не будем пытаться здесь объяснить декомпозицию сингулярного значения; ограничимся тем, что объявим ее эффективным средством анализа задач, включая матрицы. Функция `svd` вычисляет три коэффициента в декомпозиции сингулярного значения

```
[U,S,V] = svd(A);
```

причем

```
A = U*S*V';
```

Матрицы U и V являются ортогональными, матрица S - диагональной. Функция `svd(A)` выдает диагональные коэффициенты S, которые являются сингулярными значениями матрицы A.

Декомпозиция сингулярного значения используется несколькими другими функциями, включая псевдоинверсию `pinv(A)` , ранг матрицы `rank(A)` , евклидову матричную норму `norm(A,2)` , условное число `cond(A)` .

4.4. Собственные значения

Если `A` - квадратная матрица размера `n` , то вектор `d` , удовлетворяющий матричному уравнению `Ax = dx` , содержит собственные значения матрицы A. Они находятся при помощи функции `eig(A)` , которая выдает собственные значения в вектор - столбец. Если A - действительная, симметричная, то собственные значения будут действительными. Но если A - не симметричная, то собственные значения могут быть комплексными. Например,

```
A = [0 1; -1 0];
eig(A)
ans =
        0.0000 + 1.0000i
        0.0000 - 1.0000i
```

Вычислим собственные значения и собственные векторы

```
[X,D] = eig(A);
```

в этом случае диагональные элементы матрицы D являются собственными значениями, а столбцы матрицы X являются соответственно собственными векторами, так что A*X = X*D .

Двумя промежуточными результатами, используемыми при расчете собственных значений, являются форма Гессенберга, hess(A) и форма Шура, schur(A). Форма Шура используется при расчете трансцендентных математических функций матриц, таких как sqrtm(A) и logm(A).

Если A и B квадратные матрицы, то функция eig(A,B) выдает вектор d , содержащий обобщенные собственные значения, удовлетворяющие матричному уравнению Ax = dBx . В этом случае

```
[X,D] = eig(A,B);
```

возвращает диагональную матрицу D обобщенных собственных значений и матрицу X, столбцы которой являются соответственно собственными векторами, так что A*X = B*X*D . Промежуточные результаты, используемые в решении задачи получения обобщенных собственных значений, вычисляет функция qz(A,B).

4.5. Ранг

В MATLAB имеется несколько разных функций, где ранг матрицы рассчитывается неявно: в rref(A) , в A\B для неквадратной A, в orth(A) , в null(A) и в псевдоинверсии pinv(A) . Используются три разных алгоритма с тремя разными критериями точности приближения и поэтому возможно, что три разные значения ранга могут быть получены для одной и той же матрицы.

Для rref(A) ранг матрицы есть число ненулевых строк. Алгоритм исключения, используемый для rref(A) , является самым быстрым из трех алгоритмов определения ранга, но он численно наименее сложный и наименее надежный. Алгоритм pinv(A) требует больше всего времени, но является самым надежным и, следовательно, используется также для расчета явного ранга в rank(A).

5. СВЯЗЬ С MS-DOS И ДРУГИМИ ПРОГРАММАМИ

5.1. Прогон внешних программ

Символ восклицательного знака ! - переход во внешнюю оболочку используется для указания, что остаток входной строки должен быть передан как команда в операционную систему. Это полезно для утилит или прогона других программ без выхода из MATLAB. Например,

```
!f77 simplprog
```

активизирует компилятор Фортрана F77,

```
!translate
```

запускает программу `translate` , пакета MATLAB (см. п.5.2),

```
!edit darwin.m
```

вызывает редактор `edit` и загружает в него файл `darwin.m`. После завершения программ управление возвращается в MATLAB.

Вызывать внешние программы можно из программных файлов или функций пользователя. Сделать подобное в MATLAB можно, записав М-файл,который:

а) записывает переменные на диск;

б) прогоняет внешние программы (считывающие файлы данных, обрабатывающие их и записывающие результат обратно на диск);

в) загружает обработанные файлы обратно в рабочую память.

Вот, к примеру, предполагаемая М-функция, которая находит решение уравнения Гарфилда, используя внешнюю программу с именем GAREQN:

```
function y = garfield(a,b,q,r)
save gardata a b q r
!gareqn
load gardata
```

Вы должны написать программу (на Фортране или другом языке) с именем GAREQN, которая читает файл с именем `gardata.mat` , обрабатывает их и помещает результат в тот же самый файл. Для чтения и записи в MAT-файлы можно использовать подпрограммы - утилиты MATLAB.

Это средство удобно для привязки ваших собственных программ к MATLAB. Во многих системах оно быстрее и удобнее физической загрузки нового объектного кода.

5.2. Экспорт и импорт данных

Данные из других программ и из внешней среды могут вводиться в MATLAB несколькими способами. Аналогично данные из MATLAB могут экспортироваться во внешнюю среду.

Выбор наилучшего метода зависит от того, сколько данных, находятся ли они в читабельной форме, какая это форма и т.д. Вот несколько вариантов, выберите подходящий.

1. Если у вас небольшое количество данных, скажем, не больше 10 - 15 элементов, то легко ввести эти данные явно, в виде списка элементов, используя квадратные скобки [ ]. Этот метод не удобен для больших объемов данных, т.к. нельзя отредактировать ввод, если ошибетесь.
2. Используйте текстовый редактор для набивки программного M - файла, который вводит ваши данные в виде явного списка элементов. Этот метод хорош, когда данные еще не находятся в читабельной для машины форме и вам нужно их каким-то образом набрать. Преимущество по сравнению с первым методом в том, что можно использовать редактор для изменения данных или для исправления ошибок. Затем вы можете многократно использовать M-файл для повторного ввода данных.
3. Если данные хранятся в кодах ASCII со строками фиксированной длины, заканчивающимися символом новой строки (возврат каретки), и с пробелами между числами, то такой файл называется плоским файлом. Плоские файлы ASCII - кодов могут редактироваться с помощью обычного текстового редактора и могут непосредственно считываться в рабочую область оперативной памяти с помощью функции MATLAB `load` . Результат ввода присваивается переменной, имя которой совпадает с именем файла.
4. Преобразуйте ваш файл данных в формат файла MATLAB и используйте команду `load`. Программа `translate` предоставляется вместе с MATLAB в библиотеке утилит. Программа `translate` преобразует плоские файлы ASCII, двоичные файлы, неформатированные фортрановские файлы и DIF - файлы (формат обменных данных электронных таблиц и т.д.) в специальные MAT-файлы, использующиеся для сохранения на диске содержимого рабочей области MATLAB. Этот метод является лучшим, если у вас большой объем данных, и они находятся уже в дисковом файле вашего компьютера.

Опишем два метода возвращения (экспорта) данных MATLAB во внешнюю среду.

1. Для небольших матриц используйте команду `diary` для создания на диске дневникового файла. Затем вы можете использовать текстовый редактор для

манипулирования дневниковым файлом. Этот файл содержит все функции MATLAB, использованные во время сеанса, что полезно для включения в документы, доклады и отчеты.

2. Сохраните ваши переменные, используя команду `save`, выйдите из MATLAB и используйте `translate` для преобразования MAT - файла в один из возможных форматов. Вы можете манипулировать данными непосредственно в MAT-файлах, используя функции MATLAB `load` и `save`. Формат MAT - файлов задокументирован в функции `load`.

6. ВЫВОД НА ПЕЧАТЬ ТЕКСТОВОЙ И ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Для вывода на печать текстовой информации удобнее всего использовать функцию `diary`, которая направляет в заданный текстовый файл всю выводимую пакетом на экран и вводимую пользователем с клавиатуры информацию (текст, числа, таблицы), например:

```
diary report.doc
```

Затем, выйдя из MATLAB, пользователь может с помощью привычного текстового редактора отредактировать этот файл `report.doc` , убрать лишнее, добавить необходимое и отпечатать результат на принтере.

Для вывода на печать графиков, построенных в графическом экране и выведенных на монитор с помощью графических функций MATLAB [1, п.2.2.4], необходимо сначала создать графический MET - файл, т.е. файл с расширением `.met`. Для этого используется функция `meta` *имя_файла* , которая открывает на диске специальный файл с именем *имя_файла*`.met` и выводит в него текущее содержимое графического экрана. В этот же файл можно затем направить и другие графики, выведенные на экран, для чего надо выполнить функцию `meta` без имени файла.

Графический постпроцессор GPP работает с метафайлами, независимыми от устройства, и создает файлы графиков, которые могут передаваться в конкретное устройство получения твердых копий. Программа GPP активизируется на уровне MS-DOS командой следующего вида:

GPP *имя_файла*`.met /Ddevice [/F /P /OP /OL /CS /CC]`

для разделения аргументов используются пробелы, квадратные скобки указывают на необязательность аргументов, а `device` может быть одним из следующих ключевых слов (полный список совместимых печатающих устройств выводится на экран при выполнении в среде MS-DOS команды GPP без аргументов):

`epsd` - Epson принтер, черновой вариант;

`epsf` - Epson принтер, чистовой вариант;

hpgl - плоттеры, совместимые с плоттерами Hewlett Packard;

ega - EGA графический дисплей.

GPP создает новый файл, содержащий управляющие коды для конкретного устройства. Расширение имени нового файла содержит первые три буквы ключевого слова устройства. Например, функции MATLAB

```
t = -5Ø:.3:5Ø;
y = sin(t)./t;
plot(t,y)
meta sincplot
plot(t,sin(t).*t)
meta
```

создают метафайл, названный SINCPLOT.MET, содержащий два графика. В MS-DOS после выхода из MATLAB команды

```
GPP  SINCPLOT  /DEPSD
COPY  SINCPLOT.EPS  PRN /B
```

преобразуют метафайл в коды принтера Epson, помещают их в новый файл SINCPLOT.EPS и посылают в принтер Epson для печати.

Опишем необязательные аргументы:

/F Используется для переадресации выходного файла GPP. Если указать имя устройства MS-DOS, то выход GPP направляется прямо на это устройство без создания часто очень большого файла Например, если мы имеем принтер Epson, соединенный с параллельным портом, то

```
GPP  SINCPLOT  /DEPSD  /FPRN
```

посылает коды Epson непосредственно в порт принтера PRN без создания промежуточного файла SINCPLOT.EPS. Другие допустимые спецификаторы выхода представляют собой обычные названия устройств DOS:

PRN - первый параллельный принтер;

LPT1 - то же, что и PRN: ;

LPT2 - второй параллельный принтер;

COM1 - первый последовательный порт;

COM2 - второй последовательный порт.

/P Задает паузу между графиками, если их несколько в метафайле.

/OP Печать твердой копии в вертикальном направлении.

/OL Печать твердой копии в горизонтальном направлении.

/CS Задает качество символов текста. Использует симплекс или набор символов в одну строку.

/CC Задает качество символов текста. Использует комплекс или многострочный набор символов.

Для упрощения применения GPP предоставлена команда уровня MS-DOS, названная GRAPH, которая прогоняет GPP и начальная конфигурация которой выдает коды принтера Epson. Например, из MS-DOS команда

```
GRAPH  SINCPLOT
```

обрабатывает метафайл SINCPLOT.MET и направляет выход в принтер Epson, соединенный с параллельным портом.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Автоматизация физического эксперимента на базе пакета матричных вычислений / Сост.С.В.Суханов; Самар. гос. аэрокосм. ун-т. Самара,1994. 20 с.
2. Документация к пакету MATLAB / Пер. с англ.

ПРИЛОЖЕНИЕ

Список встроенных и библиотечных функций MATLAB (функция HELP)

```
>> help
MATLAB built-in functions:
Copyright (c) 1984, The MathWorks, Inc.
intro     <         chol      end       global    macro     qz        sprintf
help      >         clc       eps       grid      magic     rand      sqrt
demo      =         clear     error     hess      max       rat       startup
[         &         clg       eval      hold      memory    rcond     string
]         |         clock     exist     home      mesh      real      subplot
(         ~         conj      exit      ident     meta      relop     sum
)         abs       contour   exp       if        min       rem       svd
.         all       cos       expm      imag      nan       return    tan
,         ans       cumprod   eye       inf       nargin    round     text
;         any       cumsum    fft       input     norm      save      title
%         acos      delete    filter    inv       ones      schur     type
!         asin      det       find      isnan     pack      script    what
:         atan      diag      finite    keyboard  pause     semilogx  while
'         atan2     diary     fix       load      pi        semilogy  who
+         axis      dir       floor     log       plot      setstr    xlabel
-         balance   disp      flops     loglog    polar     shg       ylabel
```

*	break	echo	for	logop	prod	sign	zeros
\	casesen	eig	format	ltifr	prtsc	sin	
/	ceil	else	fprintf	ltitr	qr	size	
^	chdir	elseif	function	lu	quit	sort	

Strike any key to continue

...MATLAB search path is c:\matlab\mllib

Strike any key to continue

...Directory of M-files in c:\matlab\mllib

acosh	cjf2rjf	eqnerr2	funm	kaiser	neldstep	ratmovie	tanh
addtwopi	compan	etime	gallery	kron	null	rats	toeplitz
angle	computer	expm1	gamma	length	num2str	remez	trace
asinh	cond	expm2	gpp	log10	numf	remezdd	translat
atanh	conv	expm3	hadamard	logm	ode23	residue	triang
bar	conv2	feval	hamming	logspace	ode45	roots	tril
bartlett	corr	fft2	hankel	matdemo	odedemo	rot90	triu
bench	cosh	fftdemo	hanning	matlab	orth	rref	unmkpp
bessel	cov	fftshift	hilb	mean	pinv	rrefmovi	vdpol
bilinear	ctheorem	filtdemo	hist	median	plotdemo	rsf2csf	wow
blackman	deconv	fir1	histogra	membrane	poly	sinh	xcorr
blanks	demo	fir2	humps	menu	polyfit	specplot	xcorr2
boxcar	demolist	fitdemo	idft	menu1	polyval	spectrum	yulewalk
buttap	denf	fitfun	ifft	meshdemo	polyvalm	spline	zerodemo
butter	detrend	flipx	ifft2	meshdom	ppval	sqrtm	zeroin
census	dft	flipy	inpdef	mkpp	print	square	
chebap	diff	freqs	int2str	movies	quad	startup	
chebwin	edit	freqz	invhilb	nademo	quadstep	std	
cheby	eigmovie	fstab	isempty	nelder	rank	table1	

УКАЗАТЕЛЬ ФУНКЦИЙ MATLAB

abs 12
acos 12
all 11, 12
ans 4, 5, 7, 14, 16
any 8, 11, 12
asin 12
atan 12
atan2 12
bessel 13
ceil 12
chol 17
clear 15
cond 19
conj 3, 12
cos 12
cosh 13
det 15, 16, 17
diag 7
diary 22, 23
edit 21

eig 6, 19, 20
end 12, 26
exist 15
exp 6, 13
expm 7
eye 7
find 8, 11
finite 11
fix 12
floor 12
format 11
function 21
funm 7
gamma 13
GPP 23, 24, 25
help 7
hess 20
if 12
imag 12
inv 5, 16, 17
isnan 8, 11
length 7
load 21, 22, 23
log 13
log10 13
logm 7, 20
lu 16, 17
magic 10
max 7
meta 23, 24
NaN 8, 11
norm 19
null 19, 20
ones 7
orth 19, 20
pi 5, 12, 13
pinv 19, 20
plot 24
prod 15
QR 18, 19
qz 20
rand 7
Rank 18, 19, 20
rat 13
rcond 5, 17
real 12
rem 10, 12
round 12
rref 17, 20
save 21, 23
schur 20
sign 12
sin 12, 13, 24
sinh 12
size 7, 15
sqrt 6, 7, 12
sqrtm 7, 20
std 15
sum 15
svd 19
tan 12
tanh 13
tril 7
triu 7
zeros 7

Содержание

Введение 3

1. Операции над матрицами 3

1.1. Транспонирование 3
1.2. Сложение и вычитание 4
1.3. Умножение матриц 4
1.4. Деление матриц 5
1.5. Возведение матриц в степень 6
1.6. Трансцендентные функции матриц 6
1.7. Функции, полезные для построения матриц 7
1.8. Особенности вычислений с неопределенными значениями 8

2. Операции над массивами 8

2.1. Сложение и вычитание массивов 8
2.2. Умножение и деление массивов 9
2.3. Степени массивов 9
2.4. Операции отношения 9
2.5. Логические операции 11
2.6. Элементарные математические функции 12

3. Векторы и индексы 13

3.1. Генерация векторов 13
3.2. Индексирование 14
3.3. Индексирование 0–1 векторами 15
3.4. Пустые матрицы 15

4. Матричные функции 16

4.1. Разложение на треугольные множители 16
4.2. Ортогональное разложение на множители 18
4.3. Декомпозиция сингулярного значения 19
4.4. Собственные значения 19
4.5. Ранг 20

5. Связь с MS-DOS и другими программами 21

5.1. Прогон внешних программ 21
5.2. Экспорт и импорт данных. 22

6. Вывод на печать текстовой и графической информации 23

Библиографический список 25

Приложение 25

Указатель функций MATLAB 26

МАТРИЧНЫЕ ВЫЧИСЛЕНИЯ НА ПЕРСОНАЛЬНОЙ ЭВМ

Составитель Суханов Сергей Васильевич

Редактор Т.И. Кузнецова
Технический редактор Н.М. Калешюк

Подписано в печать 30.08.94. Формат 60x84 1/16.
Бумага офсетная. Печать офсетная.
Усл.печ.л. 1,6 Усл.кр.-отт. 1,7 Уч.-изд.л. 1,5.
Тираж 200 экз. Заказ 322 Арт. С-44 мр/94.

Самарский государственный аэрокосмический
университет имени академика С.П. Королева.
443086 Самара, Московское шоссе, 34.

ИПО Самарского государственного аэрокосмического
университета имени академика С.П. Королева.
443001 Самара, ул. Ульяновская, 18.