SCILAB: Библиотеки

Глава 1. Библиотека линейной алгебры

Содержание главы:

Введение

Линейная алгебра, как никакая другая ветвь математики самым тесным образом переплелась с многочисленными приложениями и является важным инструментом в решении многих прикладных задач. Хотя в пакете Scilab выделена отдельная алгебраическая библиотека, команды, относящиеся к линейной алгебре разбросаны и по другим библиотекам и разделам. Поэтому в этом пособии будут представлены наиболее важные (по нашему мнению) команды, относящиеся к предмету матричных вычислений и связанных с ним задач. Здесь же будут для удобства приведены и некоторые элементарные команды, не входящие непосредственно в библиотеку линейной алгебры, но необходимые для работы с матрицами.

Доступ в алгебраическую и другие основные библиотеки является стандартным и, поэтому при вызове команд оттуда указывать имя библиотеки не требуется.

Краткий русско-английский словарь основных терминов линейной алгебры

Вырожденность	**singularity**
Детерминант (определитель)	**determinant**
Единичная матрица	**identity matrix**
Модуль матрицы	**absolute value**, **magnitude**
Норма матрицы	**matrix norm**
Обратная матрица	**inverse of matrix**
Псевдообратный	**pseudoinverse**
Пучок матриц	**pensil matrix**
Размерность	**dimension**
Ранг матрицы	**rank of a matrix**
Разреженная матрица	**sparse matrix**
Собственные значения матрицы	**eigenvalues of matrices**
Сингулярное разложение	**singular value decomposition (SVD)**
Ядро (нуль-пространство) матрицы	**kernel**, **nullspace**

Как найти обратную матрицу(inverse of matrix)?

Способ 1.

С помощью команды **inv**.
Синтаксис
inv(X)
Параметры
X : действительная или комплексная квадратная матрица, полиномиальная матрица или рациональная матрица.
Для полиномиальной матрицы команда даст тот же результат, что и **invr**.
В результате преобразования **inv(X)*X**, будет иметь вид единичной матрицы (то есть диагональные элементы =1, а остальные элементы =0). Обратная матрица находится методом LU-факторизации, используя библиотеку LAPACK.

Пример 1.

```
a=[1 2 3;2 2 3; 5 -1 2]
b=inv(a)
```

Результат:

```
b =
! - 1.          1.          -8.327E-17 !
! - 1.5714286   1.8571429   - .4285714 !
!   1.7142857 - 1.5714286     .2857143 !

c=a*b
```

Результат:

```
c =
!   1.          0.          1.110E-16 !
! - 8.882E-16   1.          0.        !
!   4.441E-16 - 8.882E-16   1.        !
```

То есть практически получили единичную матрицу.

Пример 2.
Получение обратной матрицы для матрицы, заданной в алгебраической (символьной форме).

```
r=[poly([1 2 3],"x","c"),1;poly([1 -2],"x","c"),2]
```

Результат :

```
r =
! 1 + 2x + 3x^2   1 !
!                  !
! 1 - 2x          2 !
```

```
inv(r)
```

Результат:

```
ans =
!      2              - 1          !
! ------------   --------------- !
! 1 + 6x + 6x^2    1 + 6x + 6x^2   !
!                                  !
!  - 1 + 2x       1 + 2x + 3x^2    !
! ------------   --------------- !
! 1 + 6x + 6x^2    1 + 6x + 6x^2   !
```

Замечание: Если обращаемая матрица вырожденна (singular) или почти вырожденна (определитель матрицы равен или близок к нулю) (например, `a=[1 2 3;11 22 33; 110 220 330]`), то в результате выполнения команды `b=inv(a)` получим сообщение:

```
!--error 19
singular matrix
```

Способ 2.
Для обращения матриц в алгебраических виде существует специальная команда **invr**.
Синтаксис
F = invr(H)
Параметры
H : полиномиальная или рациональная матрица
F : полиномиальная или рациональная матрица
В результате работы команды вычисляется $F=H^{-1}$ с помощью алгоритма Леверье (Leverrier).

Способ 3.
Для полиномиальных матриц с помощью команды **coffg**.
Параметры
Fs : квадратная полиномиальная матрица.

Команда **coffg** вычисляет $\mathbf{Fs}^{-1}$ методом ко-факторов (co-factors method).
В результате выполнения команды будут получены два числа, полностью характеризующие обратную матрицу:
Ns = числитель (является полиномиальной матрицей) и **d** = общий знаменатель.

Матрица, обратная данной матрице **Fs**, будет вычисляться по формуле **Ns/d**.

Замечание: Для больших матриц результат может оказаться неудовлетворительным.

Пример.
```
r=[poly([1 2 3],"x","c"),1;poly([1 -2],"x","c"),2]
r =
!1 + 2x + 3x²   1!
!                !
!1 - 2x         2!

[a,d]=coffg(r)
```
Результат:
```
d =
1 + 6x + 6x²

a =
!  2       - 1            !
!                         !
!- 1 + 2x    1 + 2x + 3x² !
```
Матрица, `s=a/d` будет обратной к матрице `r`.

Как найти псевдообратную (pseudoinverse) матрицу?

С помощью команды псевдоинверсии **pinv**.

Синтаксис
pinv(A,[tol])
Параматры
A : действительная или комплексная матрица
tol : действительное число

Команда вычисляет такую матрицу **X** той же размерности, что и **A'**, чтобы выполнялось условие:
A*X*A = A, **X*A*X = X** и оба **A*X** и **X*A** были Эрмитовы (Hermitian) .
Вычисление базируется на методе сингулярного разложения SVD (Singular Value Decomposition) и все сингулярные значения меньшие, чем допуск (tolerance) очищаются до нуля. Этот допуск является необязательным параметром команды **pinv**.

Пример.
```
A=rand(5,2)*rand(2,4);
norm(A*pinv(A)*A-A,1) // определяется норма матрицы
```
С помощью команды **norm** мы определили, с какой погрешностью была произведена псевдоинверсия.

Как найти сингулярное разложение матрицы?

С помощью команды **svd** (=singular value decomposition).
Синтаксис
s=svd(X)

[U,S,V]=svd(X)

[U,S,V]=svd(X,0) (obsolete)

[U,S,V]=svd(X,"e")

[U,S,V,rk]=svd(X [,tol])

Параметры

X : действительная или комплексная матрица
s : действительный вектор (сингулярные значения)
S : действительная диагональная матрица (сингулярные значения)
U, **V** : ортогональная или унитарная квадратная матрица (сингулярные вектора).
tol : действительное число. Играет роль допуска.

Команда **svd** вычисляет три матрицы (**X**, **S** и **V**), обладающие следующими свойствами:
матрица **S** диагональна и имеет ту же размерность, что и матрица **X**. Ее диагональные элементы неотрицательны и расположены в порядке убывания;
матрицы **U** и **V** унитарны и выполняется условие: **X = U*S*V'** или производится разложение "экономичного размера".
Если матрица **X** имеет размер **m** на **n** и m > n, то вычисляются только первые **n** столбцов матрицы **U**, а матрица **S** будет иметь размер **n** на **n**.
В случае синтаксиса **s=svd(X)**, возвращает вектор **s**, содержащий сингулярные значения.
Конструкция **[U,S,V,rk]=svd(X [,tol])** возвращает дополнительный параметр **rk**, числовое значение ранга матрицы **X**, то есть число сингулярных значений, больших чем значение допуска **tol**.
Значение **tol** по умолчанию совпадает с рангом матрицы.

Пример.
```
X=rand(4,2)*rand(2,4)
svd(X)
sqrt(spec(X*X'))
```

Как найти собственные значения матрицы?

С помощью команды **spec**.
Подробно смотрите с помощью **help spec**.

Как определить норму матрицы?

С помощью команды **norm**.
Матричные нормы часто требуются при анализе матричных алгоритмов. Например, программа решения системы уравнений может давать некачественный результат, если матрица коэффициентов "почти вырожденная". Для количественной характеристики близости к вырожденности нам нужно уметь измерять расстояния в пространстве матриц. Такую возможность предоставляют матричные нормы.

Синтаксис команды **norm**.
[y]=norm(x [,flag])
Параметры
x : действительный или комплексный вектор или матрица (тип `full` или `sparse`)
flag : строка (тип нормы) (значение по умолчанию =2)

Для матриц:
norm(x) : эквивалентно **norm(x,2)** и является наибольшим сингулярным значением **x** (`max(svd(x))`).
norm(x,1) : наибольшая сумма по столбцу, то есть `maxi(sum(abs(x),'r'))`.

norm(x,'inf'),norm(x,%inf) : бесконечная норма (infinity norm) **x**. Является наибольшая суммой по строке: `maxi(sum((x),'c'))`.
norm(x,'fro') : Норма Фробениуса (Frobenius). Вычисляется по формуле `sqrt(sum(diag(x'*x)))`.

Для векторов эта команда определяет p-нормы.
norm(v,p) : l_p норма $(\text{sum}(v(i)^p))^{(1/p)}$.
norm(v) : равна =`norm(v,2)`, то есть норма при p=2.
norm(v,'inf') : `max(abs(v(i)))`.

Замечание:
Наиболее важными являются нормы при `p=1, 2, infinity`.

norm(v,1) является суммой модулей всех элементов вектора.
norm(v,1)=|x(1)|+|x(2)|+...+|x(n)|

norm(v,2) является квадратным корнем из суммы квадратов всех элементов вектора
norm(v,2)=(|x(1)|^2+|x(2)|^2+...+|x(n)|^2)$^{1/2}$

norm(v,'inf') равна модулю максимального элемента вектора по модулю

Пример.
```
v=[1,2,3];
norm(v,1) // Результат: 6=1+2+3
norm(v,'inf') // Результат: значение максимального элемента =3
A=[1,2;3,4]
norm(A,1) // Результат =6. Это максимальная из сумм элементов по столбцам
(2+4)
```

Как определить модуль матрицы?

С помощью команды **abs**.

Синтаксис
t=abs(x)
Параметры
x : действительный или комплексный вектор или матрица
t : действительный вектор или матрица

Команда возвращает модуль всех элементов **x**.
Если значения **x** комплексны, то команда **abs(x)** возвращает комплексный модуль всех элементов **x**.

Пример 1.
```
a=[1.1   -2.2   3;4.4  -5.4  0]
```

Результат:
```
a=
! 1.1  - 2.2   3. !
! 4.4  - 5.4   0. !
t=abs(a)
```
Результат:
```
! 1.1   2.2   3. !
! 4.4   5.4   0. !
```

Пример 2.
```
x=[1,%i,2;-1,-%i,1+%i] //%i - мнимая единица
```
Результат:
```
x =
!   1.    i    2.     !
! - 1.  - i    1. + i !
t=abs(x)
```
Результат:
```
ans =
! 1.   1.   2.     !
! 1.   1.   1.4142136 !
```

Как определить детерминант (определитель) матрицы?

Способ 1.
С помощью команды **det**.
Замечание: Матрица должна быть квадратной.

Синтаксис команды **det**
det(X) [e,m]=det(X)
Параметры
X : действительная или комплексная квадратная матрица, полиномиальная или рациональная матрица
m : действительное или комплексное число, детерминант основанный на мантиссе 10.
e : целое число, степень числа

Представление детерминанта в виде двух чисел способствует повышению точности в численных вычислениях.
det(X)= m*10^{e}
Для полиномиальных матриц команда **det(X)** эквивалентна команде **determ(X)**.
Для рациональных матриц команда **det(X)** эквивалентна команде **detr(X)**.

Пример 1.
```
x=poly(0,'x');
a=[x,1+x;2-x,5]
```
Результат:
```
a=
!x        1 + x!
!              !
!2 - x    5    !

D=det(a)
```
Результат:
```
D =- 2 + 4x^2 + x
```

Пример 2.

```
s=[1,2.3;-4,5.2]
[e,m]=det(s)
D=det(s)
```

Результат:

```
D=14.4
m = 1.44
e = 1.
```

Мы проверили, что `det(s)=m*10^e`.

Способ 2.

С помощью команды **detr**.

Синтаксис

d=detr(h)

Параметры

h : полиномиальная или рациональная квадратная матрица

Метод вычислений основан на алгоритме Леверрье(Leverrier).

Способ 3.

С помощью команды **determ(X)**. Эта команда специально служит для вычисления детерминанта полиномиальных (то есть представленных в символьном виде) матриц. Вычисление с помощью этой команды детерминанта матриц, заданных численно, недопустимо.

Синтаксис

res=determ(W [,k])

Параметры

W : действительная квадратная полиномиальная матрица

k : целое число, служащее верхней границей в представлении детерминанта матрицы **W** в виде полинома.

Детерминант вычисляется с помощью метода FFT.

Пример.

```
s=poly(0,'s');
w=[s, 1+s^2;s-2, s+4]

w =
!       s    1 + s^2!
!                   !
! - 2 + s    4 + s !

determ(w)
```

Результат:

```
ans =
2 + 3s + 3s^2 - s^3
determ(w,2)
```

Результат:

```
ans =
5 + 2s
```

Замечание: Команда **determ** с незаданным параметром **k** даст тот же результат, что и команда **det(w)**.

Как извлечь из матрицы диагональные элементы?

С помошью команды **diag**.
Синтаксис
[y]=diag(vm, [k])
Параметры
vm : вектор или матрица (полная или разреженная)
k : целое число. По умолчанию принимает значение "0", то есть действие команд `diag(vm)` и `diag(vm,0)` эквивалентно.
y : вектор или матрица
Для матрицы **vm** команда **diag(vm,k)** возвращает вектор или матрицу из элементов, стоящих по **k** -той диагонали матрицы. Команда **diag(vm)** возвращает вектор из элементов главной диагонали и эквивалентна команде **diag(vm,0)**. Значения **k>0** соответствуют диагоналям, лежащим выше главной диагонали матрицы, а **k<0** соответственно ниже.

Пример.
```
A=[1,2;3,4]
A =
! 1.  2. !
! 3.  4. !
d_1=diag(A) // Главная диагональ
```

Результат:
```
d_1 =

! 1. !
! 4. !
diag(A,1)
```

Результат: `ans =2`
```
diag(A,-1)
```
Результат: `ans =4`

Замечание: Для создания диагональной линейной системы используйте команду **sysdiag**.

Как определить ранг матрицы?

С помощью команды **rank**.

Синтаксис
```
[i]=rank(X)
[i]=rank(X,tol)
```
Параметры
X : действительная или комплексная матрица
tol : неотрицательное действительное число

Вспомним определение:
Рангом матрицы называется наивысший порядок отличного от нуля минора этой матрицы. Фактически рангом матрицы является максимальным числом линейно независимых строк матрицы. Понятие ранга является важным при решении системы линейных уравнений. Необязательный параметр **tol** лимитирует точность вычислений.

Пример.
```
a=[1 2 3; -6 3 -2;10 20 30;49 -2 5]
```
Результат:
```
a =

!   1.    2.   3.  !
!  -6.    3.  -2.  !
!  10.   20.  30.  !
!  49.   -2.   5.  !

t=rank(a)
```
Результат:
```
t =
3.
```

Как найти ядро (нуль-пространство) матрицы?

С помощью команды **kernel**.
Вспомним определение ядра (нуль-пространством) матрицы:
Ядро матрицы `A` – это множество векторов `x` таких, произведение матрицы `A` на которые равно нулевому вектору. Поиск ядра матрицы А эквивалентен решению системы линейных однородных уравнений.
`Ядром матрицы A являются все решения уравнения AW=0.`

Синтаксис команды **kernel**
W=kernel(A [,tol,[,flag])
Параметры
A : действительная или комплексная матрица. В случае, если матрица представлена в разреженном виде (тип sparse), то только действительная **A**.
flag : строка, имеющая значение **'svd'** (по умолчанию) или **'qr'**
tol : действительное число
W : полная матрицы

Команда **kernel** возвращает ортономальный базис нуль-пространства матрицы **A**.
W=kernel(A) возвращает ядро матрицы **A**. Если матрица **W** будет непустой, будет выполняться:
A*W=0 .

Параметры **flag** и **tol** являются необязательными: **flag = 'qr'** или **'svd'** (по умолчанию принимает значение **'svd'**). Указывают на используемый алгоритм вычисления.

tol = параметр допуска. В качестве значения **tol** по умолчанию принимается величина порядка `%eps(~2.220E-16)`.

Пример.
```
A=[2 -3 5 7;4 -6 2 3;2 -3 -11 -15]

A =
! 2. - 3.    5.    7. !
! 4. - 6.    2.    3. !
! 2. - 3. - 11. - 15. !

W=kernel(A)
```

```
W =
!   .3026009   .7751569 !
!   .2244353   .5075518 !
! - .7491452   .3042427 !
!   .5448329 - .2212674 !

s=A*W
```

Результат:
```
s =

1.0E-14 *

! .0444089 .0222045 !
! .0666134 .0666134 !
! .1776357 .0888178 !
```

Видно, что `s` является матрицей с практически нулевыми элементами.

Солвер для решения линейных уравнений

Для решения линейных уравнений служит команда **linsolve**.

Синтаксис
[x0,kerA]=linsolve(A,b [,x0])
Параметры
A : действительная матрица размера **na** x **ma** Возможно представление матрицы с помощью типа "разреженная" (sparse).

b :вектор, размерность которого совпадает с числом строк в матрице A (то есть **na**
x0 : действительный вектор
kerA : действительная матрица размером **ma** x **k**

Команда **linsolve** вычисляет все решения уравнения **A*x+b=0**.

x0 частное решение (если существует) и **kerA** равно ядру матрицы **A**. Если **x=x0+kerA*w** с произвольным значением **w**, удовлетворяющим условию **A*x+b=0**.

Пример.
Решим систему уравнений:
```
   x1-x2+1=0
-2*x1-x2+3=0

a=[1 -1;-2 -1]
b=[1 3]'
[x,kerA]=linsolve(a,b)
```
Результат:
```
x =
! .6666667 !
! 1.6666667 !
```

Для проверки выполним:
```
r=a*x+b
```

Результат:

```
r =
1.0E-15 *
! .3330669 !
! .4440892 !
```

Замечание:
Для "хороших" линейных уравнений `kerA=[]`. Если детерминант матрица `A` близок к нулю, то в результате выполнения команды **linsolve** получим не пустое значение `kerA`. Если детерминант матрицы `A` недопустимо (с точки зрения Scilab) близок к нулю, то вместо результата мы получим следующее сообщение:

```
WARNING:Conflicting linear constraints!
kerA =
[]
x =
[]
```