Matlab/Octave basics

Ágnes Baran, Csaba Noszály

Vectors

The `row` and `column` vectors are `different`!

creating row-vectors:

The vector $a = (-1.2, 3.1, 4.7, 1.9)$ can be created by listing its elements enclosed by square brackets, separating them by `comma`:

```
a = [-1.2, 3.1, 4.7, 1.9]
```

or by `space`:

```
a = [-1.2 3.1 4.7 1.9]
```

Vectors

1. The indexing starts at `1`.
2. the i-th element of the vector is `a(i)`.
3. `length(a)` returns the number of elements in a.
4. the empty vector is `[]`

Vectors as regular sequences

the comma operator:

It is the most useful syntax for defining vectors.

- for the vector $b = (1, 2, 3, 4, 5)$ simply use: `b = 1:5`
- for $c = (5, 4, 3, 2, 1)$: `c = 5:-1:1`
- for $d = (2, 2.2, 2.4, 2.6, 2.8, 3)$: `d = 2:0.2:3`

the comma syntax:

```
x = first:stepsize:last
```

where the default value of stepsize is `one`, that is

```
x = first:last
```

is the shorter form of

```
x = first:1:last
```

the `linspace` function:

- the vector $e = (1, 1.2, 1.4, 1.6, 1.8, 2)$ can be created by:

```
e = linspace(1,2,6)
```

- for a 100 element vector f:

```
f = linspace(1,2)
```

`linspace` syntax:

```
x = linspace(first, last, numberOfElements)
```

The elements in the resulting vector are `equally spaced`. The `default` number of elements is 100:

```
x = linspace(first, last)
```

is the shorter form of

```
x = linspace(first, last, 100)
```

Column-vectors

creation:

- by listing the
  elements separated by `semicolon`, enclosing them in square brackets:

```
m = [-3; 0; 7]
```

- transposing a row-vector:

```
n=[1 -2 4 -1]'
```

Note that: the operator `apostrophe` is the conjugate-transpose operator, so the result is not the expected one for complex valued vectors. For "transposing only" complex vectors use the function `transpose`

Column-vectors

The usage of `x(i)` and `length(x)` is the same as for row-vectors.
The call `size(x)` returns the vector

`[numOfRows, numOfColumns]`

where `numOfRows=1` for row-vectors and `numOfColumns=1` for column-vectors. In Octave/Matlab the vectors are `2`-dimensional objects.

Common ways of defining vectors

- `[a, b]`: extending along the 2nd-dimension, the sizes of the 1st dimension must agree
- `[m;n]`: extending along the 1st-dimension, the sizes of the 2nd dimension must agree
- `[-4 a 3 -1]`: extending along the 2nd dimension by listing the new elements
- `[1;m;-3]`: extending along the 1st dimension by listing the new elements
- `h(2:4)`: extract a subvector by specifying the indices of the desired elements
- `h([1 4 5])`: extract a subvector by specifying the indices of the desired elements
- `h(2)=[]`: clear individual elements
- `h([2 4])=[]`: clear elements specified by an index vector

Important! For `a=[-1 3 2]` the result of the command `a(6)=4` is the vector `a=[-1 3 2 0 0 4]`. Not that on could expect. The undefined

Some useful functions

- `min(x)` and `max(x)` returns the smallest and the largest element of x
- `sort(x)` returns the sorted instance of x (the default order: increasing)
- `sort(x,'descend')` returns the reversely sorted instance of x
- `flip(x)` returns a new vector (matrix) with elements of x in reverse order (for matrices the elements are the rows by default)
- `length(x)` number of elements
- `sum(x)` sum of the elements
- `prod(x)` product of the elements
- `mean(x)` arithmetic mean of the elements of x
- `x(end)` in vector (matrix) context the `end` has special meaning: the last element of the object

Vector operations

For the same shaped vectors `a` and `b`

- `a+b` and `a-b` is the elementwise sum and difference
- `x=a+1` for convenience the basic operations with a scalar defined elementwise
- `x=a.^2` see above
- `x=a.*b` it is the elementwise product, results in a new vector $a_i b_i$
- `x=a./b` see above
- `x=1./a` shorter version of `ones(size(a))./a`
- `dot(a,b)` the scalar, inner product, the sum-product of the vectors

Fontos! The dot before the operator results elementwise operations.

The builtin functions `sin`, `cos`, `tan`, `exp`, `log`, `sqrt`, `abs`, can have vector and matrix parameters, resulting the function called elementwise.

`NaN` : Not a Number is a result of undefined operation. For example 0/0, `Inf/Inf`)

Exercise 1

(a) Without typing the elements, create the following vectors:

(1) $a = (0, 1, \cdots, 30)$
(2) $b = (2, 4, 6, \ldots, 100)$,
(3) $c = (2, 1.9, 1.8, \cdots, 0.1, 0)$
(4) $d = (0, 3, 6, \ldots, 27, 30, -100, 30, 27, \cdots, 6, 3, 0)$
(5) $e = \left(\frac{1}{2}, \frac{1}{3}, \cdots, \frac{1}{20}\right)$
(6) $f = \left(\frac{1}{2}, \frac{2}{3}, \cdots, \frac{19}{20}\right)$

(b) You are given the vector $x = 1 : 100$. Define the vector

(1) with elements of x, but in the reverse order
(2) of the first 5 elements of x
(3) with elements of x, except the 4th one.
(4) with elements of x, except the 5.,72 and 93. ones.
(5) of the odd indexed elements of x
(6) of the 2., 5., 17. and 81. elements of x

Exercise 2

Your are given the vector $x = randi(10, 1, 10)$. Without using any loop construct, define the vector y for which

(1) $y(i) = x(i) + 2$

(2) $y(i) = x(i)^2$

(3) $y(i) = 1/x(i)$

(4) $y(i) = \sin(x(i)^3 - 1)$

(5) $y(i) = x(i) - i$

Defining matrices by listing

$A = [1, 2, 3; 4, 5, 6; 7, 8, 9]$ or $A = [1 \quad 2 \quad 3; \quad 4 \quad 5 \quad 6; \quad 7 \quad 8 \quad 9]$ results in:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

The `comma` separates the elements of the row, the `semicolon` separates the rows

The indexing for matrices is `one` based, as for vectors.

`A(i,j)` is the (i, j)-th element of A

Defining matrices from vectors

For vectors `a=[1 -2 0]`; `b=[2 -11 7]`; `m=[-3;0;7]`; `n=[1; -2; 0]`;
the result of
`B=[a;b]`:

$$B = \begin{pmatrix} 1 & -2 & 0 \\ 2 & -11 & 7 \end{pmatrix}$$

`C=[a'  b']`:

$$C = \begin{pmatrix} 1 & 2 \\ -2 & -11 \\ 0 & 7 \end{pmatrix} \qquad D = \begin{pmatrix} -3 & 1 \\ 0 & -2 \\ 7 & 0 \end{pmatrix}$$

Extending matrices

For matrices and vectors defined above the result of `E=[A;a]` (or `E=[A;[1,-2,0]]`):

$$E = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 1 & -2 & 0 \end{pmatrix}$$

The result of `F=[A m]` (or `F=[A, m]`):

$$F = \begin{pmatrix} 1 & 2 & 3 & -3 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 7 \end{pmatrix}$$

Extending matrices

For `G=[C D]` and `H=[C;D]` we get:

$$G = \begin{pmatrix} 1 & 2 & -3 & 1 \\ -2 & -11 & 0 & -2 \\ 0 & 7 & 7 & 0 \end{pmatrix} \qquad H = \begin{pmatrix} 1 & 2 \\ -2 & -11 \\ 0 & 7 \\ -3 & 1 \\ 0 & -2 \\ 7 & 0 \end{pmatrix}$$

For `C(4,5)=9`:

$$C = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ -2 & -11 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9 \end{pmatrix}$$

The size of matrix has changed without warning!

Accessing elements, rows, columns and submatrices

- `size(A)`: the number of rows and columns in row-vector
- `length(A)` = `max(size(A))`
- `numel(A)` the number of elements in A
- `A(i,j)` the i, j-th element
- `A(i,:)` the i-th row-vector
- `A(:,j)` the j-th column
- `A(2:3,:)` the 2-nd and 3-rd rows
- `A([1 2 4],:)` the 1., 2. and 4. rows
- `A(:,[1 3])` the 1. and 3. columns
- `A(2:3,[1 3])` the 1. and 3. elements of the 2. and 3. rows

Manipulating matrices

Deleting rows and columns

- `A(i,:)=[]` will delete the *i*-th row
- `A(:,j)=[]` will delete the *j*-th column
- `A([1 3],:)=[]` will delete the 1-st and 3-rd rows
- `A(:,[1 3])=[]` will delete the 1-st and 3-rd columns

Exchanging rows, columns

Swapping the *i*-th and *j*-th column:
`A([i,j],:)=A([j,i],:)`, and `A(:,[i,j])=A(:,[j,i])` respectively.

Transforming to vector:

`A(:)` the elements of A listed in column major order:
`[A(:,1);...;A(:,end)]`

Predefined matrices

`eye(n)`	the identity matrix of size $n \times n$
`eye(n,m)`	the identity matrix of size $n \times m$
`ones(n)`	the $n \times n$ array of all ones
`ones(n,m)`	the $n \times m$ array of all ones
`zeros(n)`	the $n \times n$ array of all zeros
`zeros(n,m)`	the $n \times m$ array of all zeros

Matrix-vector operations

For matrices A and B and a scalar c , the operations

`A+B` `A-B` `c*A` `A*B` `A^2`

are performed in the usual way (as we learned in mathematics). The operation

`A+c`

is defined for convenience, it is a shorter form of `A+c*ones(size(A))`.
The result of

`A/B` and `A\B`

$A \cdot B^{-1}$ and $A^{-1} \cdot B$ respectively.

The `.` before the operator results in elementwise operation, so the ij-th element of

- `A.*B` is $a_{ij} * b_{ij}$,
- `A.^2` is a_{ij}^2,
- `A./B` is a_{ij}/b_{ij}.

Most of the built-in functions can be called with matrix argument, with elementwise evaluation.

Exercise 3

Let

$$x = [-1 \;\; 4 \;\; 0],\; y = [3 \;\; -2 \;\; 5] \text{ and } A = [-3 \;\; 1 \;\; -4; 6 \;\; 2 \;\; -5]$$

For the commands given below describe the result or explain why it cannot be performed!

(1) $z = [x, y]$
(2) $z = [x; y]$
(3) $z = [x', y']$
(4) $z = [x'; y']$
(5) $z = [A; x]$
(6) $z = [A, x]$
(7) $z = [x; A; y]$
(8) $z = [A'; x]$
(9) $z = [A', x]$
(10) $z = [A', x']$
(11) $x + y$
(12) $x + y'$
(13) $A + y$
(14) $A + 2$
(15) $x./y$
(16) $A\,\hat{}\,2$
(17) $A.\hat{}\,2$

Exercise 4

Let

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

Construct the matrix B, which is made by

(1) deleting the 1-st row of A
(2) deleting the 2. and 4. columns row of A
(3) deleting the last row and column row of A
(4) putting A next to itself.
(5) transposing A
(6) exchanging the 2. and 4. columns of A
(7) squaring the elements of A

(8) adding 3 to each element of A
(9) taking the square root of the elements of A
(10) taking the sine of the elements of A
(11) by setting a_{12} to -2
(12) by setting the second row of A to $[-1 \;\; 0 \;\; -2 \;\; 3]$

Exercise 5

- Define the matrix below:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 20 & 18 & 16 & 14 & 12 & 10 & 8 & 6 \\ 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 \end{pmatrix}$$

- Examine the results of the following commands:

(1) `sum(A)`
(2) `sum(A,2)`
(3) `reshape(A,6,4)`
(4) `max(A)`
(5) `max(A,[],2)`
(6) `max(A,2)`
(7) `flipud(A)`
(8) `fliplr(A)`
(9) `size(A)`
(10) `length(A)`

Some more builtins

- `det(A)`: determinant of A
- `inv(A)`: inverse of A
- `dot(a,b)`: scalar (inner) product of a and b
- `norm(A)` 2-norm of A (also for vectors)
- `norm(A,inf)` ∞-norm of A
- `norm(A,1)` 1-norm of A

Solving the system $Ax = b$:

```
x=A\b
```

Some more builtins

`diag`

- `diag(a)` gives a square matrix whose main diagonal is a
- `diag(a,k)` returns a square matrix whose k-th diagonal is a.
- `diag(A,k)` returns the k-th diagonal of A

Note

The main diagonal is the 0-th one. The diagonal above the k-th one is the $k+1$-th one.

Some more builtins

`tril` and `triu`

- `tril(A)`: returns the lower triangle part of A.
- `triu(A)`: returns the upper triangle part of A.
- `tril(A,k)`: return the matrix A with elements above the k-th diagonal set to `0`.
- `triu(A,k)`: return the matrix A with elements below the k-th diagonal set to `0`.

Defining functions

The structure (syntax) of Octave/Matlab functions:

```
function outVariables = funName( inVariables )
   % body of the function
end
```

Important! The name of file to which your function saved should be `funName.m`.

Defining functions

Example

```
function y=quad(x)
    y=2*x.^2-3*x+5;
end
```

The result of the command `y=masodf(x)` is the value of $2x^2 - 3x + 5$, where x can be a vector or matrix, because we used elementwise operations in the definition.

Logical operators, functions

- $<$, $<=$, $>$, $>=$, $==$, $\sim=$ (for vectors,matrices the comparison is done elementwise)
- `A&B`, `A|B`, `~A`, `xor(A,B)` (elementwise evaluationt)
- `all(x)`: is true if all elements of v is nonzero.
- `all(A)`: is a row vector `[all(A(:,1),...,all(A(:,end)))]`, that is `all` is applied column-wise.
- `all(A,2)` `all` is applied row-wise.
- `any(x)`: is true if some element of v is nonzero.
- `any(A)`: is a row vector `[any(A(:,1),...,any(A(:,end)))]`, that is `any` is applied column-wise.
- `any(A,2)` `any` is applied row-wise.

Logikai függvények

- `ind=find(A)` returns the indices of nonzero elements of A (column-wise ordering is used).
- `ind=find(A,n)` returns the indices of the first n nonzero elements of A (column-wise ordering is used).
- `ind=find(A,n,last)` returns the indices of the last n nonzero elements of A (column-wise ordering is used).
- `[rowI,colI]=find(A)` returns the row-column indices of nonzero elements of A.
- `[rowI,colI,elem]=find(A)` same as the previous, except that it catches also the nonzero elements.

Logical functions

The `find` can be called with logical vector, matrix argument:

- `find(A<=B)`
- `find(A==5)`
- `find(A>5,4)`
- `find(A>5,4,last)`
- `find(A<5 & A>4)`
- `find(abs(A-2)<=0.01)`

`logical(A)` converts A to a logical array, the nonzeros map to the logical 1.

Branching, if-else

```
if logicalExpression
   % body
else
   % body
end
```

Example

```
N=input('Type an integer: ');
if mod(N,3)==0
    disp('divisible by 3');
else
    disp(sprintf('the remainder after dividing by 3 is: %d', mod(N,3) ))
end
```

Useful structures, functions

- `break` : immediately exits the execution of the containing for or while `loop`
- `continue` : stops the execution of the body's statements, and then continues with the next iteration
- `pause` : waits for press a key
- `pause(n)` : waits for *n* seconds
- `return` : stops the execution of the script (or function)
- `error('message')` : stops the execution of the script (or function) displaying `'message'`

Exercise 6

Let $x = [0 \quad -1 \quad 2 \quad 0 \quad 4 \quad 4]$ and $y = [-1 \quad -2 \quad 3 \quad 1 \quad 0 \quad 4]$. What is the result of the evaluation of the following expressions?

(1) $x == y$
(2) $x <= y$
(3) $x > y$
(4) $x > 0$
(5) $y <= 3$
(6) $x|y$
(7) $x\&y$

Exercise 7

For the vectors of the previous exercise, what is the result of:

(1) `find`$(x == y)$
(2) `find`$(x <= y)$

Exercise 8

Let `a=rand(1,20)` Create the vector b containing the elements of with $a_i > 0.5$!

Exercise 9

Find an appropriate loopless expression that creates the following matrices in a given size:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 3 & 0 & 3 & 0 & 0 \\ 0 & 6 & 0 & 4 & 0 \\ 0 & 0 & 9 & 0 & 5 \end{pmatrix}, \quad B = \begin{pmatrix} -2 & 2 & 2 & 2 & 2 \\ 0 & -2 & 2 & 2 & 2 \\ 0 & 0 & -2 & 2 & 2 \\ 0 & 0 & 0 & -2 & 2 \\ 0 & 0 & 0 & 0 & -2 \end{pmatrix}$$

Exercise 10

Find an expression, which extends (at the end) the rows of a given matrix by the `mean` of the numbers in the row.

Exercise 10(b)

Find an expression, which extends the columns (at the end) of a given matrix by the `sum` of the numbers in the row.

Exercise 11(benchmark)

Examine the following snippets:

```
clear all
N=100000; # a huge vector
x=rand(1,N);

disp('a naive way:')
tic
for i=1:N
   y(i)=x(i)+i;
end
toc
```

```
disp('a little bit smarter:')
tic
yy=zeros(1,N);
for i=1:N
  yy(i)=x(i)+i;
end
toc

disp('the expected way:')
tic
yyy=x+1;
toc
```

`Lesson:` Avoid using of loops if possible and use the built-in features instead!