Scilab: basic/matrix operations

Dr. Madhu N. Belur

Control & Computing group
Department of Electrical Engineering
Indian Institute of Technology Bombay
Email: belur@ee.iitb.ac.in

Outline

Today's focus

- Scilab is free.
- Matrix/loops syntax is same as for Matlab.
- Scilab provides all basic and many advanced tools.
- Signal processing, Control systems, block-diagram-simulation (xcos), Electrical, networks, op-amps, device simulation, ++
- Toolboxes: image/video processing, wavelets, neural networks, filter design, hardware-interfacing for real time control, ++
- French National Space Agency (CNES) (like India ISRO) uses only Scilab (search google Martin CNES Scilab)
- Teaching Matlab is like MBBS colleges teaching to-be-doctors how to use only very expensive medicines
- Today: basic matrix operations and loops syntax, plotting
- Also: signal processing: DSP,

Today's focus

- Scilab is free.
- Matrix/loops syntax is same as for Matlab.
- Scilab provides all basic and many advanced tools.
- Signal processing, Control systems, block-diagram-simulation (xcos), Electrical, networks, op-amps, device simulation, ++
- Toolboxes: image/video processing, wavelets, neural networks, filter design, hardware-interfacing for real time control, ++
- French National Space Agency (CNES) (like India ISRO) uses only Scilab (search google Martin CNES Scilab)
- Teaching Matlab is like MBBS colleges teaching to-be-doctors how to use only very expensive medicines
- Today: basic matrix operations and loops syntax, plotting
- Also: signal processing: DSP,

Today's focus

- Scilab is free.
- Matrix/loops syntax is same as for Matlab.
- Scilab provides all basic and many advanced tools.
- Signal processing, Control systems, block-diagram-simulation (xcos), Electrical, networks, op-amps, device simulation, ++
- Toolboxes: image/video processing, wavelets, neural networks, filter design, hardware-interfacing for real time control, ++
- French National Space Agency (CNES) (like India ISRO) uses only Scilab (search google Martin CNES Scilab)
- Teaching Matlab is like MBBS colleges teaching to-be-doctors how to use only very expensive medicines
- Today: basic matrix operations and loops syntax, plotting
- Also: signal processing: DSP,

Today's focus

- Scilab is free.
- Matrix/loops syntax is same as for Matlab.
- Scilab provides all basic and many advanced tools.
- Signal processing, Control systems, block-diagram-simulation (xcos), Electrical, networks, op-amps, device simulation, ++
- Toolboxes: image/video processing, wavelets, neural networks, filter design, hardware-interfacing for real time control, ++
- French National Space Agency (CNES) (like India ISRO) uses only Scilab (search google Martin CNES Scilab)
- Teaching Matlab is like MBBS colleges teaching to-be-doctors how to use only very expensive medicines
- Today: basic matrix operations and loops syntax, plotting
- Also: signal processing: DSP,

Today's focus

- Scilab is free.
- Matrix/loops syntax is same as for Matlab.
- Scilab provides all basic and many advanced tools.
- Signal processing, Control systems, block-diagram-simulation (xcos), Electrical, networks, op-amps, device simulation, ++
- Toolboxes: image/video processing, wavelets, neural networks, filter design, hardware-interfacing for real time control, ++
- French National Space Agency (CNES) (like India ISRO) uses only Scilab (search google Martin CNES Scilab)
- Teaching Matlab is like MBBS colleges teaching to-be-doctors how to use only very expensive medicines
- Today: basic matrix operations and loops syntax, plotting
- Also: signal processing: DSP,

Today's focus

- Scilab is free.
- Matrix/loops syntax is same as for Matlab.
- Scilab provides all basic and many advanced tools.
- Signal processing, Control systems, block-diagram-simulation (xcos), Electrical, networks, op-amps, device simulation, ++
- Toolboxes: image/video processing, wavelets, neural networks, filter design, hardware-interfacing for real time control, ++
- French National Space Agency (CNES) (like India ISRO) uses only Scilab (search google Martin CNES Scilab)
- Teaching Matlab is like MBBS colleges teaching to-be-doctors how to use only very expensive medicines
- Today: basic matrix operations and loops syntax, plotting
- Also: signal processing: DSP,

Today's focus

- Scilab is free.
- Matrix/loops syntax is same as for Matlab.
- Scilab provides all basic and many advanced tools.
- Signal processing, Control systems, block-diagram-simulation (xcos), Electrical, networks, op-amps, device simulation, ++
- Toolboxes: image/video processing, wavelets, neural networks, filter design, hardware-interfacing for real time control, ++
- French National Space Agency (CNES) (like India ISRO) uses only Scilab (search google Martin CNES Scilab)
- Teaching Matlab is like MBBS colleges teaching to-be-doctors how to use only very expensive medicines
- Today: basic matrix operations and loops syntax, plotting
- Also: signal processing: DSP,

Defining a matrix

- `A=[1 3 4 6]` and `A=[1 3 4 6];`
- `B=[1 3 4 6;5 6 7 8] // next row by ;`
- `length(A), ones(A), zeros(B), zeros(3,5)`
- `size(A), size(A,'c'), size(A,'r')`
- `B(1,3)`
- `B(1,3) = -45`
- `quote ( ' ) for transpose of A`

Defining a matrix

- `A=[1 3 4 6]` and `A=[1 3 4 6];`
- `B=[1 3 4 6;5 6 7 8] // next row by ;`
- `length(A), ones(A), zeros(B), zeros(3,5)`
- `size(A), size(A,'c'), size(A,'r')`
- `B(1,3)`
- `B(1,3) = -45`
- `quote ( ' ) for transpose of A`

Defining a matrix

- `A=[1 3 4 6]` and `A=[1 3 4 6];`
- `B=[1 3 4 6;5 6 7 8] // next row by ;`
- `length(A), ones(A), zeros(B), zeros(3,5)`
- `size(A), size(A,'c'), size(A,'r')`
- `B(1,3)`
- `B(1,3) = -45`
- `quote ( ' ) for transpose of A`

Defining a matrix

- `A=[1 3 4 6]` and `A=[1 3 4 6];`
- `B=[1 3 4 6;5 6 7 8] // next row by ;`
- `length(A), ones(A), zeros(B), zeros(3,5)`
- `size(A), size(A,'c'), size(A,'r')`
- `B(1,3)`
- `B(1,3) = -45`
- `quote ( ' ) for transpose of A`

Random numbers/matrices

rand(9) generates a 1×1 random number (uniformly distributed between 0 and 1, etc: see help).

If A is an $n \times p$ matrix, then

B = rand(A) // defines a random matrix B of the size of A.

(A is not overwritten. The matrix B is defined.)

Random numbers/matrices

rand(9) generates a 1×1 random number (uniformly distributed between 0 and 1, etc: see help).
If A is an $n \times p$ matrix, then
B = rand(A) // defines a random matrix B of the size of A.
(A is not overwritten. The matrix B is defined.)

Random numbers/matrices

rand(9) generates a 1×1 random number (uniformly distributed between 0 and 1, etc: see help).
If A is an $n \times p$ matrix, then
B = rand(A) // defines a random matrix B of the size of A.
(A is not overwritten. The matrix B is defined.)

Random numbers/matrices

rand(9) generates a 1×1 random number (uniformly distributed between 0 and 1, etc: see help).
If A is an $n \times p$ matrix, then
B = rand(A) // defines a random matrix B of the size of A.
(A is not overwritten. The matrix B is defined.)

Multiplication and addition

(Vectors are also matrices: with number of rows/columns = 1).
Suppose A and B are matrices of 3 rows and 4 columns.
A+B // same sizes of A and B required
A*B // Not defined. size(A,'c') $\neq$ size(B,'r')
A*B' // dimensions allow multiplication
A+4 // understood as elementwise addition
A*4 and 4*A // understood as elementwise multiplication
A.*B // understood as elementwise multiplication of matrices

Multiplication and addition

(Vectors are also matrices: with number of rows/columns = 1).
Suppose A and B are matrices of 3 rows and 4 columns.
A+B // same sizes of A and B required
A*B // Not defined. size(A,'c') $\neq$ size(B,'r')
A*B' // dimensions allow multiplication
A+4 // understood as elementwise addition
A*4 and 4*A // understood as elementwise multiplication
A.*B // understood as elementwise multiplication of matrices

Multiplication and addition

(Vectors are also matrices: with number of rows/columns = 1).

Suppose A and B are matrices of 3 rows and 4 columns.

A+B // same sizes of A and B required

A*B // Not defined. size(A,'c') ≠ size(B,'r')

A*B' // dimensions allow multiplication

A+4 // understood as elementwise addition

A*4 and 4*A // understood as elementwise multiplication

A.*B // understood as elementwise multiplication of matrices

Multiplication and addition

(Vectors are also matrices: with number of rows/columns = 1).
Suppose A and B are matrices of 3 rows and 4 columns.
A+B // same sizes of A and B required
A*B // Not defined. size(A,'c') $\neq$ size(B,'r')
A*B' // dimensions allow multiplication
A+4 // understood as elementwise addition
A*4 and 4*A // understood as elementwise multiplication
A.*B // understood as elementwise multiplication of matrices

Multiplication and addition

(Vectors are also matrices: with number of rows/columns = 1).
Suppose A and B are matrices of 3 rows and 4 columns.
A+B // same sizes of A and B required
A*B // Not defined. size(A,'c') $\neq$ size(B,'r')
A*B' // dimensions allow multiplication
A+4 // understood as elementwise addition
A*4 and 4*A // understood as elementwise multiplication
A.*B // understood as elementwise multiplication of matrices

for and if-then-end

- v = -3:10 // vector from -3 to 20 (default increment 1)
- v=1:2.3:10 // vector of 'suitable length'
- for i=1:10, // (or for i=v,)
  disp(i)
  end
- i==4 // check this for undefined i, for i=5 and i=4
- if (true or false) then
  do something
  end
- (else, elseif-else, etc possible)

for and if-then-end

- v = -3:10 // vector from -3 to 20 (default increment 1)
- v=1:2.3:10 // vector of 'suitable length'
- for i=1:10, // (or for i=v,)
  disp(i)
  end
- i==4 // check this for undefined i, for i=5 and i=4
- if (true or false) then
  do something
  end
- (else, elseif-else, etc possible)

for and if-then-end

- v = -3:10 // vector from -3 to 20 (default increment 1)
- v=1:2.3:10 // vector of 'suitable length'
- for i=1:10, // (or for i=v,)

  ```
          disp(i)
  end
  ```
- i==4 // check this for undefined i, for i=5 and i=4
- if (true or false) then

  ```
          do something
  end
  ```
- (else, elseif-else, etc possible)

commands into a file

Write all commands, etc in a file (using Notepad or Scilab-editor)
.sce extension recommended
exec filename or exec('filename',2) // does not echo your file
filename could be a 'function' : very important

commands into a file

Write all commands, etc in a file (using Notepad or Scilab-editor)
.sce extension recommended
exec filename or exec('filename',2) // does not echo your file
filename could be a 'function' : very important

function

Recommended .sci extension

file1.sci (can have multiple functions with different function names)

These names should not already exist!

```
// —- file1.sci begins blah-blah
function total = addition(a,b)
total = a+b
endfunction
// — file1.sci ends
```

Scilab prompt:

```
-> exec('file1.sci',2)
-> q = addition(4,5)
```

eigenvalues (spectrum), trace

- `det(A), spec(A), trace(A)`
- `sum prod`
- `disp(i) (display i) or disp('Text to be displayed')`
- `rank, svd`
- `find`

eigenvalues (spectrum), trace

- `det(A), spec(A), trace(A)`
- `sum prod`
- `disp(i) (display i)` or `disp('Text to be displayed')`
- `rank, svd`
- `find`

eigenvalues (spectrum), trace

- `det(A), spec(A), trace(A)`
- `sum prod`
- `disp(i) (display i) or disp('Text to be displayed')`
- `rank, svd`
- `find`

plot and plot2d

- x = 0:0.3:3
- y = x.^2 // elementwise squaring
- plot(x) and plot(x,y) // independent, dependent-variable(s)
- plot(x',y') // Now 2$^{\mathrm{nd}}$ argument can have many columns
- x = 0:0.3:3'; y2 = x.^2; y3 = x.^3;
- plot(x, [y2 y3]),
- plot2d(x, [y2 y3], [2, -4]) // help plot and plot2d

plot and plot2d

- x = 0:0.3:3
- y = x.^2 // elementwise squaring
- plot(x) and plot(x,y) // independent, dependent-variable(s)
- plot(x',y') // Now 2$^{\text{nd}}$ argument can have many columns
- x = 0:0.3:3'; y2 = x.^2; y3 = x.^3;
- plot(x, [y2 y3]),
- plot2d(x, [y2 y3], [2, -4]) // help plot and plot2d

Defining polynomials

Polynomials play a very central role in control theory: the transfer function is a ratio of two polynomials.
Key commands:

- `s=poly(0,'s')`
- `s=poly(0,'s','roots')`
- `p=s^2+3*s+2`
- `p=poly([2 3 1],'s','coeff')`
- `roots(p)`
- `horner(p,5)`
- $a = [1\ \ 2\ \ 3]$
- `horner(p,a)`
- `horner(p,a')`
- `w=poly(0,'w')`
- `horner(p,%i*w)`

Differentiation

- `p=poly([1 3 4 -3],'s','coeff')`
- `cfp=coeff(p) constant term first`
- `diffpcoff=cfp(2:length(cfp)).*[1:length(cfp)-1]`
- `diffp=poly(diffpcoff,'s','coeff')`
- degree(p) can be used instead of `length(cfp)-1`
- Of course, derivat(p)

Differentiation

- `p=poly([1 3 4 -3],'s','coeff')`
- `cfp=coeff(p)` constant term first
- `diffpcoeff=cfp(2:length(cfp)).*[1:length(cfp)-1]`
- `diffp=poly(diffpcoeff,'s','coeff')`
- degree(p) can be used instead of `length(cfp)-1`
- Of course, derivat(p)

Differentiation

- `p=poly([1 3 4 -3],'s','coeff')`
- `cfp=coeff(p) constant term first`
- `diffpcoeff=cfp(2:length(cfp)).*[1:length(cfp)-1]`
- `diffp=poly(diffpcoeff,'s','coeff')`
- degree(p) can be used instead of `length(cfp)-1`
- Of course, derivat(p)

Differentiation

- `p=poly([1 3 4 -3],'s','coeff')`
- `cfp=coeff(p)` constant term first
- `diffpcoeff=cfp(2:length(cfp)).*[1:length(cfp)-1]`
- `diffp=poly(diffpcoeff,'s','coeff')`
- degree(p) can be used instead of `length(cfp)-1`
- Of course, derivat(p)

Differentiation

- `p=poly([1 3 4 -3],'s','coeff')`
- `cfp=coeff(p) constant term first`
- `diffpcoff=cfp(2:length(cfp)).*[1:length(cfp)-1]`
- `diffp=poly(diffpcoff,'s','coeff')`
- degree(p) can be used instead of `length(cfp)-1`
- Of course, derivat(p)

Differentiation

- `p=poly([1 3 4 -3],'s','coeff')`
- `cfp=coeff(p) constant term first`
- `diffpcoeff=cfp(2:length(cfp)).*[1:length(cfp)-1]`
- `diffp=poly(diffpcoeff,'s','coeff')`
- degree(p) can be used instead of `length(cfp)-1`
- Of course, derivat(p)

csv: comma separated values file

write_csv : writes data into a csv file.

read_csv : reads data from a csv file.

horner find inv and pinv

dft ($+$ and -1: options compulsory!)

Useful inbuilt variables:

%i, %s, %z, %e, %eps

a=5×10^{-3} : a = 5D-3

also a = 5d-3 & 5e-3 & 5E-3

Commands/variables are case sensitive!

csv: comma separated values file

write_csv : writes data into a csv file.

read_csv : reads data from a csv file.

horner find inv and pinv

dft (+ and − 1: options compulsory!)

Useful inbuilt variables:

%i, %s, %z, %e, %eps

a=5×10^{-3} : a = 5D-3

also a = 5d-3 & 5e-3 & 5E-3

Commands/variables are case sensitive!

csv: comma separated values file
write_csv : writes data into a csv file.
read_csv : reads data from a csv file.
horner find inv and pinv
dft (+ and − 1: options compulsory!)
Useful inbuilt variables:
%i, %s, %z, %e, %eps
a=5×10^{-3} : a = 5D-3
also a = 5d-3 & 5e-3 & 5E-3
Commands/variables are case sensitive!

csv: comma separated values file
write_csv : writes data into a csv file.
read_csv : reads data from a csv file.
horner find inv and pinv
dft ($+$ and $-$ 1: options compulsory!)
Useful inbuilt variables:
%i, %s, %z, %e, %eps
a=5×10^{-3} : a = 5D-3
also a = 5d-3 & 5e-3 & 5E-3
Commands/variables are case sensitive!

The command 'find'

find([%T %F %T %F %T %F %F])
gives 'indices' of TRUE's.
x=[4 5 6 7]
$x < 5.5$
$true\_indices\_of\_x$=find($x < 5.5$)
y=[5 6 7 8 9 0 -1]
y($true\_indices\_of\_x$)

Conclusions

- Matrices and polynomials provide rich source of problems
- Due to good (and cheap/free) computational tools available currently, the future lies in computational techniques
- Scilab provides powerful tools
- We saw: `for` `horner` `poly` `coeff` `roots` `find`