Scilab

Vachan Potluri
vachanpotluri@iitb.ac.in

February 6, 2023

Introduction

What is Scilab?

A free alternative to MATLAB

Introduction

What is Scilab?

A free alternative to MATLAB

What can it do?

1. Advanced calculator
2. Programming
3. Plotting, visualisation

Simple calculations

Try out these and see if they give expected results

```
2+3-4
4^2
4**4
6/4
2+(2^2-(1/2))
1e-3 + 1d-2
```

Simple calculations

Try out these and see if they give expected results

```
2+3-4
4^2
4**4
6/4
2+(2^2-(1/2))
1e-3 + 1d-2
```

See what happens when you add a semicolon

```
6/4;
```

Variables

All calculations are stored by default in `ans`

```
6/4;
ans
```

Variables

All calculations are stored by default in `ans`

```
6/4;
ans
```

You can specify a variable to store the value instead

```
pi_approx = 22/7;
```

And see its value later

```
pi_approx
disp(pi_approx)
```

More on variables

Some useful pre-defined variables

```
%pi
%e
%i
%t
%f
%inf
%nan
%eps
```

Pre-defined functions

See if the outputs of these lines are as expected

```
1  abs(-2)
2  min(3,4,5)
3  max(-2,-3,-4)
4  sin(%pi/2)
5  cos(%pi)
6  tan(%pi/4)
7  asin(1)/(%pi/2)
8  exp(2)/%e^2
9  log10(100)
10 log(%e)
```

Auto-completion: hit `TAB`

Other Scilab windows

- Variable Browser
  - Only lists user-defined variables
  - To list all variables:

```
whos
```

Other Scilab windows

- Variable Browser
  - Only lists user-defined variables
  - To list all variables:

```
whos
```

  - You can delete all or specific user-defined variables

```
pi_approx = 22/7;
disp(pi_approx)
clear pi_approx
disp(pi_approx)
```

Other Scilab windows

- Variable Browser
  - Only lists user-defined variables
  - To list all variables:

```
whos
```

  - You can delete all or specific user-defined variables

```
pi_approx = 22/7;
disp(pi_approx)
clear pi_approx
disp(pi_approx)
```

- Command History
  - Execute an old command by double clicking
  - Can also navigate using ↑ and ↓ keys
  - Clear screen using `clc`

Other Scilab windows

- Variable Browser
  - Only lists user-defined variables
  - To list all variables:

```
whos
```

  - You can delete all or specific user-defined variables

```
pi_approx = 22/7;
disp(pi_approx)
clear pi_approx
disp(pi_approx)
```

- Command History
  - Execute an old command by double clicking
  - Can also navigate using ↑ and ↓ keys
  - Clear screen using `clc`
- File Browser
  - Useful when working with multiple files

Basic matrix creation

Wrap inside `[]`, use `,` and `;` to separate columns and rows

```
x = [1,2,3]
y = [4;5;6;7]
A = [1,0;0,1]
```

Basic matrix creation

Wrap inside `[]`, use `,` and `;` to separate columns and rows

```
x = [1,2,3]
y = [4;5;6;7]
A = [1,0;0,1]
```

Scilab will warn you if the dimensions are inconsistent

```
B = [1,2,3;4,5]
```

Basic matrix creation

Wrap inside `[]`, use `,` and `;` to separate columns and rows

```
x = [1,2,3]
y = [4;5;6;7]
A = [1,0;0,1]
```

Scilab will warn you if the dimensions are inconsistent

```
B = [1,2,3;4,5]
```

Adding `'` will transpose the matrix

```
B = [1,2,3;4,5,6];
B'
```

Basic matrix creation

Wrap inside `[]`, use `,` and `;` to separate columns and rows

```
x = [1,2,3]
y = [4;5;6;7]
A = [1,0;0,1]
```

Scilab will warn you if the dimensions are inconsistent

```
B = [1,2,3;4,5]
```

Adding `'` will transpose the matrix

```
B = [1,2,3;4,5,6];
B'
```

You can fill matrices with pre-existing matrices

```
row1 = [1,2,3,4];
row2 = [5,6,7,8];
M = [row1;row2]
```

Special functions for matrix creation

Creating ranges

```
i = 1:10
j = 1:2:10
x = 0:0.1:1
y = linspace(0,1,25)
```

Special functions for matrix creation

Creating ranges

```
i = 1:10
j = 1:2:10
x = 0:0.1:1
y = linspace(0,1,25)
```

Some useful commands for creating dummy matrices of required size

```
A = zeros(2,2)
B = ones(3,2)
M = eye(3,3)
```

Special functions for matrix creation

Creating ranges

```
i = 1:10
j = 1:2:10
x = 0:0.1:1
y = linspace(0,1,25)
```

Some useful commands for creating dummy matrices of required size

```
A = zeros(2,2)
B = ones(3,2)
M = eye(3,3)
```

Can you make sense of this result?

```
M = [[zeros(1,2);ones(1,2);eye(2,2)],ones(4,1)]
```

Matrix operations

Scalar operations affect all elements of matrices

```
A = eye(3,3);
A*2
A/4
A+5
```

Matrix operations

Scalar operations affect all elements of matrices

```
A = eye(3,3);
A*2
A/4
A+5
```

Scilab automatically figures out matrix operations too

```
B = 2*ones(3,3)
A+B
A*B
B^2
```

Matrix operations

Scalar operations affect all elements of matrices

```
A = eye(3,3);
A*2
A/4
A+5
```

Scilab automatically figures out matrix operations too

```
B = 2*ones(3,3)
A+B
A*B
B^2
```

Special element wise operations

```
A.+B
A.*B
A.^B
A./B
A.^2
```

How is `A^2` different from `A.^2`?

Matrix functions

Most Scilab functions can operate element-wise on matrices

```
A = %pi/2*[0,1;2,3];
sin(A)
```

Matrix functions

Most Scilab functions can operate element-wise on matrices

```
A = %pi/2*[0,1;2,3];
sin(A)
```

Some special functions for matrices

```
length(A)
size(A)
sum(A)
det(A)
inv(A)
trace(A)
```

Matrix indexing

Access elements using `(row,col)`

```
A = eye(3,3);
A(1,2) = 2;
A
```

Matrix indexing

Access elements using `(row,col)`

```
A = eye(3,3);
A(1,2) = 2;
A
```

A single index can also be used: increments column-wise

```
A(4)
```

Matrix indexing

Access elements using `(row,col)`

```
A = eye(3,3);
A(1,2) = 2;
A
```

A single index can also be used: increments column-wise

```
A(4)
```

Extract rows and columns using `:`

```
A(:,2)
A(1,:)
```

Matrix indexing

Access elements using `(row,col)`

```
A = eye(3,3);
A(1,2) = 2;
A
```

A single index can also be used: increments column-wise

```
A(4)
```

Extract rows and columns using `:`

```
A(:,2)
A(1,:)
```

Special symbol `$`

```
A($,3)
```

Matrix indexing

Access elements using `(row,col)`

```
A = eye(3,3);
A(1,2) = 2;
A
```

A single index can also be used: increments column-wise

```
A(4)
```

Extract rows and columns using `:`

```
A(:,2)
A(1,:)
```

Special symbol `$`

```
A($,3)
```

Arrays can also be used to access and modify

```
A([1,2],2)
A(4,:) = [10,20,30]
```

Matrix indexing

Access elements using `(row,col)`

```
A = eye(3,3);
A(1,2) = 2;
A
```

A single index can also be used: increments column-wise

```
A(4)
```

Extract rows and columns using `:`

```
A(:,2)
A(1,:)
```

Special symbol `$`

```
A($,3)
```

Arrays can also be used to access and modify

```
A([1,2],2)
A(4,:) = [10,20,30]
```

See if this makes sense

```
A = eye(4,4);
j = [2,4];
A(1,j) = j
A([7,8]) = 50
A($,$) = -1
B = [9,10;j];
A(B) = 100
```

Strings

Wrap in `""` or `''`

```
fname = "Vachan";
lname = 'Potluri';
fname + lname
```

Strings

Wrap in `""` or `''`

```
fname = "Vachan";
lname = 'Potluri';
fname + lname
```

Function `string` converts variables to strings

```
A = eye(2,2)
string(A)
```

Saving and loading data

Scilab has a working directory

```
pwd
```

Working directory can be changed from File Browser (and also using `cd` or `chdir`)

Saving and loading data

Scilab has a working directory

```
pwd
```

Working directory can be changed from File Browser (and also using `cd` or `chdir`)

Function `save` saves user-defined variables to a file in working directory

```
x = 1.5;
A = [1,2;3,4]
save("data.dat")
```

Saving and loading data

Scilab has a working directory

```
pwd
```

Working directory can be changed from File Browser (and also using `cd` or `chdir`)

Function `save` saves user-defined variables to a file in working directory

```
x = 1.5;
A = [1,2;3,4]
save("data.dat")
```

These variables can be loaded for use later

```
listvarinfile("data.dat")
load("data.dat")
```

Accessing help

Scilab's built-in help functionality is very useful

```
help
help save
```

Exercises[1]

Exercise 1

The pressure drop Δp required for a flow rate Q in a pipe of diameter D is

$$\Delta p = 4.52 \frac{Q^{1.85}}{C^{1.7} D^{4.87}}$$

Find Δp for these combinations of flow rates and diameters:

- $Q = 50,\ 100,\ 200,$ 400 and 1000
- $D = 0.5,\ 1,\ 1,\ 2$ and 4

[1] Amos Gilat. *MATLAB: An Introduction with Applications*. 6th ed. Wiley, 2017.

Exercises[1]

Exercise 3

The pressure drop Δp required for a flow rate Q in a pipe of diameter D is

$$\Delta p = 4.52 \frac{Q^{1.85}}{C^{1.7} D^{4.87}}$$

Find Δp for these combinations of flow rates and diameters:

- $Q = 50,\ 100,\ 200,$ 400 and 1000
- $D = 0.5,\ 1,\ 1,\ 2$ and 4

Exercise 4

A magic square is a matrix in which all rows, columns and diagonals sum to same number.

1. Generate a magic square of size 10
2. Verify its properties

Hint: search `Scilab` help for the function `testmatrix`

[1] Amos Gilat. *MATLAB: An Introduction with Applications*. 6th ed. Wiley, 2017.

SciNotes: built-in editor

- Console is only useful for short calculations
- A single file containing all commands is useful for large calculations

SciNotes: built-in editor

- Console is only useful for short calculations
- A single file containing all commands is useful for large calculations
- Scilab can do this through “scripts” or “executables”

SciNotes: built-in editor

- Console is only useful for short calculations
- A single file containing all commands is useful for large calculations
- Scilab can do this through “scripts” or “executables”
- SciNotes is Scilab’s builtin-in GUI for handling scripts

SciNotes: built-in editor

- Console is only useful for short calculations
- A single file containing all commands is useful for large calculations
- Scilab can do this through "scripts" or "executables"
- SciNotes is Scilab's builtin-in GUI for handling scripts
- Customary to save such files with `.sce` extension

Conditional statements

Can you make sense of this?

```
x=6;
reminder = modulo(x,3);

if reminder==0 then
    disp("3 divides x")
elseif reminder==1 then
    disp("x leaves reminder 1 when divided by 3")
else
    disp("x leaves reminder 2 when divided by 3")
end
```

Hint: look at help for function `modulo`

Conditional statements

Can you make sense of this?

```
x=6;
reminder = modulo(x,3);

if reminder==0 then
    disp("3 divides x")
elseif reminder==1 then
    disp("x leaves reminder 1 when divided by 3")
else
    disp("x leaves reminder 2 when divided by 3")
end
```

Hint: look at help for function `modulo`

Logical expressions generally use

`==, ~=, <, <=, >, >=, &&, ||, %t, %f`

Loops

```
1 array = 1:10;
2 value = 5;
3
4 for a=array
5     if value==a then
6         disp("Value
    ↪exists in array");
7         break;
8     end
9 end
```

What does `break` statement do?

Loops

```
1 array = 1:10;
2 value = 5;
3
4 for a=array
5     if value==a then
6         disp("Value
    ↪exists in array");
7         break;
8     end
9 end
```

Scilab always loops over columns

What does `break` statement do?

Loops

```
array = 1:10;
value = 5;

for a=array
    if value==a then
        disp("Value
    ↪exists in array");
        break;
    end
end
```

What does `break` statement do?

Scilab always loops over columns

```
array=[1;2;3]
i=1;
for a=array
    disp("Element " +
    ↪string(i) + ":")
    disp(a)
    i = i+1;
end
```

Functions

```
function [Tf,Tk] = centigradeToFarenhietKelvin(Tc)
    Tf = Tc*9/5 + 32;
    Tk = Tc + 273;
endfunction

[Tf,Tk] = centigradeToFarenhietKelvin(37);
disp(Tf)
disp(Tk)
```

Here `Tf` and `Tk` are the "return" values; `Tc` is the parameter

Functions

```
function [Tf,Tk] = centigradeToFarenhietKelvin(Tc)
    Tf = Tc*9/5 + 32;
    Tk = Tc + 273;
endfunction

[Tf,Tk] = centigradeToFarenhietKelvin(37);
disp(Tf)
disp(Tk)
```

Here `Tf` and `Tk` are the "return" values; `Tc` is the parameter

Can also have multiple parameters

```
function s = sum(a,b)
    s = a+b;
endfunction
disp(sum(1,2));
```