Scilab

Vachan Potluri
vachanpotluri@iitb.ac.in

February 3, 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

As a calculator

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
```

As a calculator

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)
```

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

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`

Manual matrix creation

Wrap inside `[]`, use `,` and `;` to fill row and columns

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

Manual matrix creation

Wrap inside `[]`, use `,` and `;` to fill row and columns

```
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]
```

Manual matrix creation

Wrap inside `[]`, use `,` and `;` to fill row and columns

```
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'
```

Manual matrix creation

Wrap inside `[]`, use `,` and `;` to fill row and columns

```
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 output?

```
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.^2
A.^B
A./B
```

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)
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
```