Overview

1. Computational environment
   - Unix
   - MatLab
2. Basic programming elements
   - Variables and assignments
   - Expressions
3. Basic programming elements
   - Arrays, vectors and matrices
4. Summary

Unix

You will be working on the Lab computers, which all run Linux.
You will need to get used to some features of Unix/Linux systems

- Nearly all big computers used for scientific computing run under the Unix operating system
- There are many "dialects" of Unix; GNU/Linux is widely used on PCs
- Most Linux software comes from the GNU free software project, which dates back a long time
- Unix used to be really user-unfriendly, but there are now nice window manager interfaces like Gnome or KDE
- It is a lot more secure and stable than Microsoft Windows!
- It is also more versatile, but to really exploit the computational capabilities you need to use the command line

The command line

Once you log in to the computer you will see a menu button, a terminal button and a browser button in the bottom left hand corner of your screen.

The command line

Once you log in to the computer you will see a menu button, a terminal button and a browser button in the bottom left hand corner of your screen.

Click on the terminal button. This brings up a terminal window.
From here you can issue any command you like!

The command line

Once you log in to the computer you will see a menu button, a terminal button and a browser button in the bottom left hand corner of your screen.

Click on the terminal button. This brings up a terminal window.
From here you can issue any command you like!

First of all you need to change your password
Do this by typing `yppasswd` in the terminal window

The command line

Once you log in to the computer you will see a menu button, a terminal button and a browser button in the bottom left hand corner of your screen.

Click on the terminal button. This brings up a terminal window.
From here you can issue any command you like!

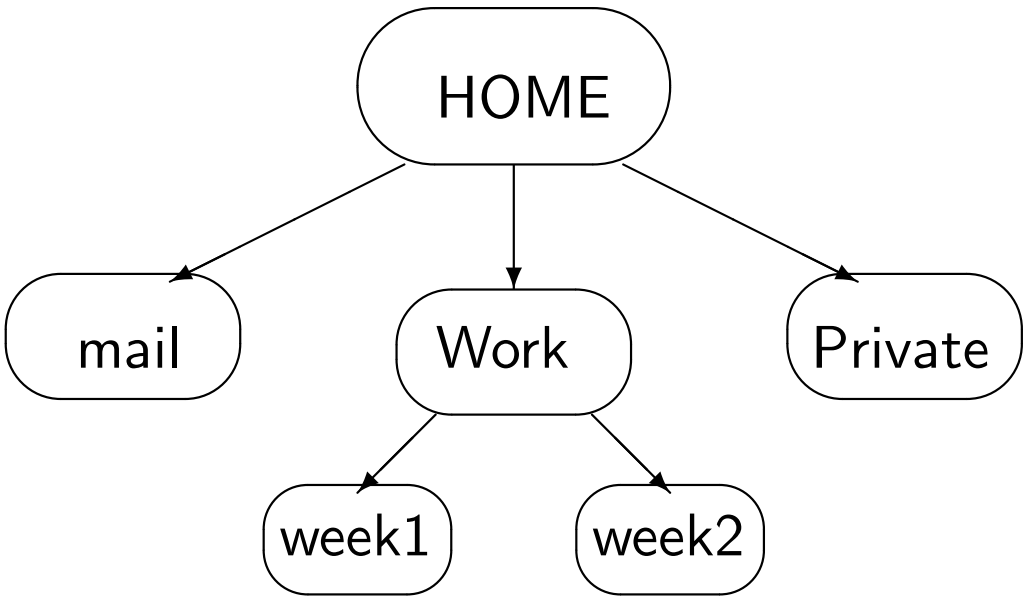
First of all you need to change your password
Do this by typing `yppasswd` in the terminal window
Choose your password carefully:
use a mix of upper and lower case, numbers and punctuation/symbols

Files and directories

Once you log in, you are in your Home directory
You can create new directories ("folders") under your home directory

Files and directories

Once you log in, you are in your Home directory
You can create new directories ("folders") under your home directory
It is very useful to organise your work and files into directories, for example:



Files and directories

Once you log in, you are in your Home directory
You can create new directories ("folders") under your home directory

Rules for file and directory names

- upper and lower case are different
- directory and file names should not contain spaces or the special characters $*?&‘“” /
- no restrictions on length!
- filenames do not need to end in .XYZ

Useful commands

`pwd`	Show your current directory [present working directory]
`mkdir` *dir*	Make a new directory
`rmdir` *dir*	Remove a directory
`cd`	Move to a different directory [change directory]
`cd` *dir*	Change to directory *dir*
`cd ..`	Move up one directory
`cd`	Move to your home directory

Useful commands

`pwd`	Show your current directory [present working directory]
`mkdir` *dir*	Make a new directory
`rmdir` *dir*	Remove a directory
`cd`	Move to a different directory [change directory]
`cd` *dir*	Change to directory *dir*
`cd ..`	Move up one directory
`cd`	Move to your home directory
`ls` *dir*	List the content of a directory
`ls -l`	Show more information about each file
`ls -F`	Make it easier to distinguish between directories and other files
`ls -a`	List all files, including those beginning with . (dot)

Useful commands

`pwd`	Show your current directory [present working directory]
`mkdir` *dir*	Make a new directory
`rmdir` *dir*	Remove a directory
`cd`	Move to a different directory [change directory]
`cd` *dir*	Change to directory *dir*
`cd ..`	Move up one directory
`cd`	Move to your home directory
`ls` *dir*	List the content of a directory
`ls -l`	Show more information about each file
`ls -F`	Make it easier to distinguish between directories and other files
`ls -a`	List all files, including those beginning with . (dot)
`cp` *file1 file2*	Copy a file
`mv` *file1 file2*	Move (rename) a file
`rm` *file*	Remove a file forever!

More useful commands

`cat` *file*	Show the contents of a (short) text file
`less` *file*	View the contents of a file
`grep` *string file*	Find occurrences of a string of characters in a file
`diff` *file1 file2*	Show the differences between two files

More useful commands

`cat` *file*	Show the contents of a (short) text file
`less` *file*	View the contents of a file
`grep` *string file*	Find occurrences of a string of characters in a file
`diff` *file1 file2*	Show the differences between two files
`emacs`	A powerful text editor
`mutt`	A versatile mail program which can be used anywhere

More useful commands

`cat` *file*	Show the contents of a (short) text file
`less` *file*	View the contents of a file
`grep` *string file*	Find occurrences of a string of characters in a file
`diff` *file1 file2*	Show the differences between two files
`emacs`	A powerful text editor
`mutt`	A versatile mail program which can be used anywhere
`chmod` *opt file*	Change the permissions of your files to prevent (or allow) others to tamper with them
`man` *command*	Get help (manual pages) on a command

More useful commands

`cat` *file*	Show the contents of a (short) text file
`less` *file*	View the contents of a file
`grep` *string file*	Find occurrences of a string of characters in a file
`diff` *file1 file2*	Show the differences between two files
`emacs`	A powerful text editor
`mutt`	A versatile mail program which can be used anywhere
`chmod` *opt file*	Change the permissions of your files to prevent (or allow) others to tamper with them
`man` *command*	Get help (manual pages) on a command

You can also pipe the output of one command into another one, for example

```
grep "stuff" file.txt | less
```

MatLab

We will be using MatLab to perform computation
— implementing and exploring the methods we encounter
At the end you will be using MatLab for a serious physics project

MatLab

We will be using MatLab to perform computation
— implementing and exploring the methods we encounter
At the end you will be using MatLab for a serious physics project

What is MatLab?

- a powerful package with lots of numerical tools
- a computing environment where you can get answers quickly
- a good training ground for programming and exploring numerical methods
- a very good plotting tool

MatLab

We will be using MatLab to perform computation
— implementing and exploring the methods we encounter
At the end you will be using MatLab for a serious physics project

What is MatLab?

- a powerful package with lots of numerical tools
- a computing environment where you can get answers quickly
- a good training ground for programming and exploring numerical methods
- a very good plotting tool

What is MatLab not?

- free software (you have to pay!)
- a programming language
- a tool for high-performance computing

Getting started

Type `matlab &` in a terminal window to start MatLab
This brings up the Matlab window

Getting started

Type `matlab &` in a terminal window to start MatLab
This brings up the Matlab window
You can use it as a calculator:

```
>> a=3

a =

     3

>> b=sin(a)

b =
     0.1411

>>
```

Getting started

Type `matlab &` in a terminal window to start MatLab
This brings up the Matlab window
You can use it as a calculator:

```
>> a=3

a =

     3

>> b=sin(a)

b =
     0.1411

>>
```

Over the next few weeks we will explore many more features!

Variables

Variables

A variable is anything with a name and a value.

In MatLab they come into being when given a value.

[In other languages you may have to declare them upfront]

On the computer, each variable is put in a dedicated box of storage space.

Variables

Variables

A variable is anything with a name and a value.
In MatLab they come into being when given a value.
[In other languages you may have to declare them upfront]
On the computer, each variable is put in a dedicated box of storage space.

Assignments

You assign a value to a variable with the statement

```
variable = value
```

Variables

Variables

A variable is anything with a name and a value.
In MatLab they come into being when given a value.
[In other languages you may have to declare them upfront]
On the computer, each variable is put in a dedicated box of storage space.

Assignments

You assign a value to a variable with the statement
`variable = value`
For example, the statment
```
>> a=2
```
gives the value 2 to the variable a.
Note that
```
>> 2=a
```
Will not work, since it will try to give the value of a to the number 2.

Expressions

The right hand side of an assignment can be an expression involving variables and/or numbers:

```
>> b = 2*a + 3;
>> c = sin(b)/(2+a);
>> vol = 4*6*(2+7.1);
```

Expressions

The right hand side of an assignment can be an expression involving variables and/or numbers:

```
>> b = 2*a + 3;
>> c = sin(b)/(2+a);
>> vol = 4*6*(2+7.1);
```

Operators and functions

- Arithmetic operators are `+ - * / \ ^`
- Remember precedence rules: use (brackets) when unsure!

Expressions

The right hand side of an assignment can be an expression involving variables and/or numbers:

```
>> b = 2*a + 3;
>> c = sin(b)/(2+a);
>> vol = 4*6*(2+7.1);
```

Operators and functions

- Arithmetic operators are `+ - * / \ ^`
- Remember precedence rules: use (brackets) when unsure!
- MatLab (and many programming languages) knows about many mathematical functions:
  `sin(x), cos(x), acos(x), sinh(x), log(x), exp(x),...`
- Remember the (brackets) around the argument!

Expressions

Computers are stupid!

You have to tell them exactly what you want them to compute.

Expressions

Computers are stupid!

You have to tell them exactly what you want them to compute.
When we write $2a$, this means '2 multiplied by a'.
So we need to tell the computer this!

Expressions

Computers are stupid!

You have to tell them exactly what you want them to compute.
When we write $2a$, this means '2 multiplied by a'.
So we need to tell the computer this!

Exercise 1

How would you write

$$x = 3ab$$

on the computer?

Exercise 2

How would you write

$$\xi = \frac{2\sin x}{3 + 2z} \quad ?$$

Numbers in MatLab

MATLAB is designed to calculate, so assumes everything is a double-precision floating-point unless you tell it otherwise:

Numbers in MatLab

MATLAB is designed to calculate, so assumes everything is a double-precision floating-point unless you tell it otherwise:

```
>> clear
>> a=2;
>> whos
  Name        Size                Bytes  Class     Attributes

  a           1x1                     8  double
```

Numbers in MatLab

MATLAB is designed to calculate, so assumes everything is a double-precision floating-point unless you tell it otherwise:

```
>> clear
>> a=2;
>> whos
  Name        Size                Bytes  Class     Attributes

  a           1x1                     8  double
```

Usually this is ok ... but beware!

Rounding errors

- There is no point trying to get answers to greater precision than machine precision

Rounding errors

- There is no point trying to get answers to greater precision than machine precision
- Critical when subtracting large numbers!

Rounding errors

- There is no point trying to get answers to greater precision than machine precision
- Critical when subtracting large numbers!

Example

Solution of $ax^2 + bx + c = 0$,

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

can be useless if $b^2 \gg ac$

Rounding errors

- There is no point trying to get answers to greater precision than machine precision
- Critical when subtracting large numbers!

Example

Solution of $ax^2 + bx + c = 0$,

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

can be useless if $b^2 \gg ac$

- We often use increased precision when summing up lots of numbers

Arrays, vectors and matrices

In physics applications we are often interested in arrays (rows) of numbers, for example

- the position of an object as a function of time
- the value of the electric field as a function of x and y
- the temperature, pressure and wind speed in different places
- the wave function of a quantum mechanical particle

Arrays, vectors and matrices

In physics applications we are often interested in arrays (rows) of numbers, for example

- the position of an object as a function of time
- the value of the electric field as a function of x and y
- the temperature, pressure and wind speed in different places
- the wave function of a quantum mechanical particle

Discretisation

Computers know nothing about the continuum

Arrays, vectors and matrices

In physics applications we are often interested in arrays (rows) of numbers, for example

- the position of an object as a function of time
- the value of the electric field as a function of x and y
- the temperature, pressure and wind speed in different places
- the wave function of a quantum mechanical particle

Discretisation

Computers know nothing about the continuum

— we must tell it to compute values at discrete positions or times x_i, t_i

$$\vec{r}(t) \to \vec{r}(t_i) \qquad \to (x_i, y_i, z_i)$$

$$\vec{E}(x, y) \to \vec{E}(x_i, y_j) \qquad \to (E_x^{ij}, E_y^{ij}, E_z^{ij})$$

etc

Vectors in MatLab

We want to operate on vectors, matrices and higher-dimensional arrays
Many computer languages are not very good at this

Vectors in MatLab

We want to operate on vectors, matrices and higher-dimensional arrays
Many computer languages are not very good at this
But this is exactly that MatLab is designed for!

Vectors in MatLab

We want to operate on vectors, matrices and higher-dimensional arrays
Many computer languages are not very good at this
But this is exactly that MatLab is designed for!
You can create a vector by just typing in a row of numbers

```
>> v = [0 2 5 6]
v =
     0 2 5 6
```

Vectors in MatLab

We want to operate on vectors, matrices and higher-dimensional arrays
Many computer languages are not very good at this
But this is exactly that MatLab is designed for!
You can create a vector by just typing in a row of numbers

```
>> v = [0 2 5 6]
v =
     0 2 5 6
```

or, more usefully, by supplying start, end and increment:

```
>> w = [0:0.1:1]
w =
  Columns 1 through 7
   0    0.1000   0.2000   0.3000   0.4000   0.5000   0.6000
  Columns 8 through 11
   0.7000   0.8000   0.9000   1.0000
```

This discretises the interval [0,1] into points with separation 0.1

Vector operations

The i'th element of a vector `v` is addressed as `v(i)`

```
>> v(1)
ans =
        0
>> v(3)
ans =
        5
```

Vector operations

The i'th element of a vector `v` is addressed as `v(i)`

```
>> v(1)
ans =
        0
>> v(3)
ans =
        5
```

MatLab also allows you to access a subset of a vector, or extend it:

```
>> w(2:4)
ans =
      0.1000    0.2000    0.3000
>> v = [v 8]
v =
     0   2   5   6 8
```

Vector arithmetic

MatLab allows you to do arithmetic directly on vectors:

```
>> u = 2*v
u =
     0   4   10   12   16
>> y = sin(w)
y =
  Columns 1 through 7
   0    0.0998   0.1987   0.2955   0.3894   0.4794   0.5646
  Columns 8 through 11
   0.6442   0.7174   0.7833   0.8415
```

Vector arithmetic

MatLab allows you to do arithmetic directly on vectors:

```
>> u = 2*v
u =
     0   4   10   12   16
>> y = sin(w)
y =
  Columns 1 through 7
   0    0.0998   0.1987   0.2955   0.3894   0.4794   0.5646
  Columns 8 through 11
   0.6442   0.7174   0.7833   0.8415
```

BUT ...

Matrices

MatLab is really at MATrix LABoratory!

Matrices

MatLab is really at MATrix LABoratory!
— its default mode is to apply the rules of linear algebra to everyting
It treats a vector as a $1 \times n$ matrix, so will try to use rules of matrix multiplication if you tell it to multiply two vectors.

Matrices

MatLab is really at MATrix LABoratory!
— its default mode is to apply the rules of linear algebra to everyting
It treats a vector as a $1 \times n$ matrix, so will try to use rules of matrix multiplication if you tell it to multiply two vectors.

Element-by-element operations

Perform element-by-element multiplication, division or exponentiation by preceding the operator with a . (dot): `.*`, `./`, `.^`

Matrices

MatLab is really at MATrix LABoratory!
— its default mode is to apply the rules of linear algebra to everyting
It treats a vector as a $1 \times n$ matrix, so will try to use rules of matrix multiplication if you tell it to multiply two vectors.

Element-by-element operations

Perform element-by-element multiplication, division or exponentiation by preceding the operator with a . (dot): `.*`, `./`, `.^`

Matrix operations

transpose	A'
inverse	`inv(A)`
solve `Ax=b`	`x=A\b`
solve `yA=c`	`y=c/A`
eigenvalues	`eig(A)`

And more!!

Summary

- Variables are labels pointing to 'boxes' in the computer's memory
- Vectors and arrays lie at the heart of computational physics
  - Elements are represented by a name and one or more indices
  - MatLab is well suited to operating on vectors and matrices!